IMT School for Advanced Studies, Lucca

Lucca, Italy

Parallel methods for solving stochastic optimal control problems: control of drinking water networks

PhD Program in Computer Science and Engineering

XXVIII Cycle

By

Ajay Kumar Sampathirao

November, 2016

The dissertation of Ajay Kumar Sampathirao is approved.

Program Coordinator: Prof. Alberto Bemporad, IMT School for Advanced Studies, Lucca

Supervisor: Prof. Alberto Bemporad, IMT School for Advanced Studies, Lucca

Supervisor: Dr. Pantelis Sopasakis, IMT School for Advanced Studies Lucca

The dissertation of Ajay Kumar Sampathirao has been reviewed by:

Prof. Colin Jones, École Polytechnique Fédérale de Lausanne

Prof. Gabriele Pannocchia, Università di Pisa

IMT School for Advanced Studies, Lucca

November, 2016

To my parents and my sister

Contents

List of Figures

List of Tables

List of Algorithms

Acknowledgements

Completing this thesis wouldn't have been possible without those who supported me one way or other in this amazing journey. I would like to express my sincere gratitude to all.

First and foremost, I am extremely grateful to my supervisor, Prof. Alberto Bemporad, for providing me the opportunity of this work. His guidance and support has an indispensable role in this research. He always shown keen interest in my work and provided the platform to interact with other researchers to explore new ideas. I am also grateful to my second supervisor and also an esteemed friend, Dr. Pantelis Sopasakis. My relationship with him is more than professional and I am indebted to him for the support during my personal family problem. I have to thank him for the effort he put on me during my early days of PhD. The discussions we had were insightful and helped in shaping this research. What I appreciate is his meticulous and systematic approach in problem solving which I happened to learned during his companionship. I am also grateful to Prof. Panagiotis Patrinos who taught me convex optimisation which is the central theme of my thesis. His advice and support are invaluable in this work.

I would like to thank all the professors, staff of IMT who are helpful through-out my time in Italy. I want to thank the EFFINET project consortium for providing the data and models for the thesis. I want to thank my aunt Manjula for proof-reading and correcting my English.

On a personal level, I am very thankful to all my friends who made my Italian experience memorable. I thank Marina, Lawrence, Andreas, Puya, Rafael, Sah, Marietta and many others for the great moments we shared. Marina is the one with whom I share my daily routine and freely talk about my worries. I cherish the conversations with her during our walks and they have personal impact on me. She took me to a Cerova which I would never forget. Lawrence, Sah, Pantelis and Marietta made the boring life of Lucca exciting with our small adventurous and it is great to go out with you guys. The ping-pong games, the wine-tastings and the barbecues we had were a great experiences and I thank Rafael and Yasim for them. Andreas, Puya, Stella, Vihang and Manas made the corridor lively with nerd guns, music and Zmart talk.

Finally, I want to thank my parents Janakiramaiah and Sailaja whose blessings and unconditional love is a source of inspiration in my dark times; I cannot express what they means to me but at this time I want remind of it. My deepest love to my sister Prathyusha. I also like to express my gratitude to my uncle Madhu who encouraged to purse my PhD

and my friends Naveen, Sudheer and other back in India for their support. Finally I would dedicate this thesis to my parents.

List of Publications

1. A.K. Sampathirao, P. Sopasakis, A. Bemporad and P. Patrinos, "Proximal quasi-Newton methods for scenario-based stochastic optimal control", IFAC 2017 World Congress (submitted).
2. A.K. Sampathirao, P. Sopasakis, A. Bemporad and P. Patrinos, "Fast parallelizable scenario-based stochastic optimization", 4th European Conference on Computational Optimization, Leuven, Belgium, Sept. 2016.
3. A.K. Sampathirao, P. Sopasakis, A. Bemporad and P. Patrinos, "Stochastic predictive control of drinking water networks: large-scale optimisation and GPUs", (provincially accepted for publication) IEEE Transactions on Control System Technology.
4. A.K. Sampathirao, P. Sopasakis, A. Bemporad and P. Patrinos, "Distributed solution of stochastic optimal control problems on GPUs", in Proc. 54th IEEE Conf. on Decision and Control, Osaka, Japan, 2015, pp. 7183–7188.
5. A.K. Sampathirao, P. Sopasakis and A. Bemporad, "Decentralised hierarchical multi-rate control of large-scale drinking water networks". In: 9th International Conference on Critical Information Infrastructures Security, Oct. 13-15 2014, Limassol, Cyprus pp. 1–12. (2014)
6. A.K. Sampathirao, J.M Grosso, P. Sopasakis, C. Ocampo-Martínez, A. Bemporad, and V. Puig, " Water demand forecasting for the optimal operation of large-scale drinking water networks: The Barcelona case study", in Proc. 19th IFAC World Congress, Cape Town, South Africa, 2014.

Abstract

This thesis is concerned with the development of optimisation methods to solve stochastic Model Predictive Control (MPC) problem and employ them in the management of Drinking Water Networks (DWNs). DWNs are large-scale, complex both in topology and dynamics, energy-intensive systems subjected to irregular demands. Managing these networks play a crucial role in the economic sustainability of urban cities. The main challenge associated with such infrastructures is to minimise the energy required for pumping water while simultaneously maintaining uninterrupted water supply. State-of-the-art control methodologies as well as the current engineering practices use predictive models to forecast upcoming water demands but do not take into consideration the inevitable forecasting error. This way, the water network is operated in a deterministic fashion disregarding its inherent stochastic behaviour which accrues from the volatility of water demand and, often, electricity prices. In this thesis, we address two challenges namely: optimisation methods for solving stochastic MPC problems and closed-loop feedback control for the management of drinking water networks.

MPC is an advanced control technology that copes

with complex control problem by repeatedly solving a finite horizon constrained optimal control problem; uses only the first decision as input and discards the rest of the sequence. This methodology decides the control action based on present state of the system and thus provides an implicit feedback to the system. Instead of historical demand profile, time-series models were developed to forecast the future water demand. The economic and the social aspects involved in operation of the DWN were captured in a cost function. Now the MPC controller combined with online forecaster minimise the cost function across a prediction horizon of 1 day with sampling time equal to 1 hour and thus the closed-loop strategy for DWN management is devised.

The forecasts are just nominal demands and differ from the actual demands. There exist several approaches when it comes to working with uncertain forecasts: (i) to assume that forecast errors are negligible and disregard them, (ii) to assume knowledge of their worst-case values (maximum errors), (iii) to assume knowledge of probabilistic information. These three approaches lead to the three principal flavours of MPC: the certainty-equivalent (CE), the worst-case robust and the stochastic MPC. CE-MPC is simple but not realistic (because the errors are not negligible), worst-case MPC is more meaningful but it is too conservative (because it is highly improbable that the errors admit their worst-case values)

and then we have stochastic MPC which is the approach pursued in this thesis.

A stochastic MPC allows a systematic framework as trade-off performance against constraint violation by modelling the uncertainty as stochastic process and quantifying its influence. However, this formulation is an infinite dimensional optimisation problem and its corresponding discrete approximation is deemed to be a large-scale problem with millions of decision variables. Therefore, the applicability of stochastic MPC in control applications is limited due to the unavailability of algorithms that can solve them efficiently and within the sampling time of the controlled system.

Here we developed optimisation algorithms that solve stochastic MPC problem by exploiting their structure and using parallelisation. These algorithms are (i) accelerated proximal gradient algorithm also known as forward-backward splitting and (ii) LBFGS method for forward-backward envelope (FBE) function. Both these algorithms employ decomposition to solve the Fenchel dual and make them suitable for parallel implementation. Graphics processing units (GPUs) are capable of perform parallel computation and are therefore perfect hardware to solve the stochastic MPC problem with the accelerated proximal gradient method.

The water network of the city Barcelona is considered to study the validity of the proposed algorithm. The GPU implementation is found to be 10 times

faster than commercial solvers like Gurobi running in multi-core environment and made the problem computationally tractable in the sampling time. The efficiency of the stochastic MPC to manage the DWN is quantified in terms of key performance indicators like economic utility, network utility and quality of service.

The forward-backward splitting is a first-order method and has slow convergence for ill-conditioned problems. We constructed a continuously differentiable real-valued forward-backward envelope function that has the same set of minimisers as the actual problem. Then we use quasi-Newton method, in particular LBFGS method, that utilises second-order information to solve the FBE. The computations with this algorithm are also parallelisable and it demonstrated fast convergence compared to accelerated dual proximal gradient algorithm.

Chapter 1

Introduction

1.1 Background and Motivation

1.1.1 Motivation

There is an increasing interest for efficient water management policies from academia, from industry and also from policy makers due to the looming environmental crisis. Water is a renewable natural resource that lies at the heart of everything essential for human life: in agriculture, in transport, in industries and in biosphere. Recent study predicts double in the utility of water demand by 2030 in every sector (Uni12). On the other hand, recent changes in climate is causing a shift in the pattern of rains resulting in floods and droughts. Even though, water is renewable, unsustainable human activities and water management policies are aiding more in depletion of the water resources than their conservation. These issues make water management an increasingly important environmental and socio-economic subject globally that needs serious attention.

Cities are the economic power-houses of a country and would become the residence for 5 billion population by 2030 (Uni15). The same report noted that there would be 41 mega-cities with more than 10 million inhabitants. Drinking Water Network (DWN) will be a critical infrastructure supplying water from the sources to the resident regions in these cities. This is vital for the survival of urban life, for maintaining a healthy level of economic development, and for the continuous operation of factories and hospitals. Management of this network plays a crucial role in economic growth and in fighting the water security challenge.

The water network is composed of a large number of interconnected pipes, tanks, pumps, valves and other hydraulic elements. The demands of certain region are lumped together and modelled as a single demand and referred as a demand unit. This network can be represented as a hierarchical system with interacting layers with each layer having different control objectives:

- In the *treatment* layer water is collected from various sources like bores, rivers, springs and processed into portable water with a specified quality by means of advanced process control methods. Sometimes the source of water is far off from the treatment plant so the water needs to be transported for long distance before the treatment plants. The control problem in this layer is to minimise cost (both in terms of energy and cost of chemicals) of the treatment plant.
- The *transportation* layer is an intermediate step involving water transfer from the treatment plants to the demand units via overhead tanks. Here, we should main-

tain enough water in the tanks to meet the requirement of the demand units. The control problem in this layer is energy efficient pumping of water while guaranteeing the needs of the consumers. A standard practice is to maintain a safety volume of water to counter emergency situations, like fire accidents, during network maintenance. Here the quality of water is also guaranteed by mixing water from various sources.

- The *service* layer is the final stage of supplying water to the individual consumers. The optimal control of this layer covers pressure zone monitoring and control by booster pumps and reduction valves to guarantee minimum pressure to all consumers and reduce leakage by maximum pressure reduction.

In 2014, the IEEE Control Systems Society identified the above aspects of the management of complex water networks as emerging future research directions (HTS14).

In the transportation layer, water from the treatment plants is transported to the overhead tanks and from these tanks to the demand units. Generally pumping consumes energy and the objective is to minimise the economic cost of pumping while guaranteeing uninterrupted water. A naive control policy is to fill the tanks completely during the off-peak hours and use this water to meet the demands during peak hours. But this policy is unsustainable and the water resources are exhausted excessively. The actuators and the tanks are needed to operate within their are physical limits. This policy could not guarantee satisfaction of these constraints.

In the previous decades, this policy is improved by considering forecast of the water demand for the next 24 hours. These

demand forecasts are obtained from the historical data. Subsequently the optimal control policy is the one that minimise the economic cost for the next while meeting the demand for the next 24 hours. The physical constraints are included in the optimisation and the control policy satisfies them. A generalised version of this policy also includes dynamic pricing of the electricity, i.e., use predictions of the electricity prices. The core optimisation problem is solved off-line and hence this is an off-line control policy. With this policy, timely intervention by the operator is required to facilitate uninterrupted water supply and guarantee constraint satisfaction.

These state-of-art control practices fail to address a major source of uncertainty – *demand uncertainty*. In practice, the off-line policy cannot meet the demand requirements and does not satisfy the constraints in closed-loop, so the operator resorts to some heuristic rules to satisfy them. The consequence of this is an ad hoc operation of the system. On the other hand, modern technologies like high-performance computation, accurate demand predictions, and advanced control techniques can enable to improve the operation of the drinking water network.

Model Predictive Control (MPC) is a modern control technology capable of handling constraints and has demonstrated enormous success in industrial process control. At each sampling time, the system dynamics over the future is predicted and a control policy that minimises the cost function over the future is calculated. The first element in the control policy is applied and the rest are discarded. This process is repeated at the next sampling time. Thus, MPC avoids solving the infinite horizon optimal control problem by repeatedly solving a finite horizon optimal every time. This also provides an 'im-

plicit' feedback action that can cope with uncertainties. The main reasons behind the success of MPC are (i) its ability to handle constraints and (ii) the recent increase in computational power of the hardware.

Besides handling constraints, another main advantage of the MPC for drinking water network is its ability to incorporate on-line demand forecasts. We develop a time-series model that can forecast the demand for the next 24 hours at every sampling. Based on these demand forecasts, the control policy is calculated by the MPC controller. However, the use of demand forecasts is not straightforward as they are subject to uncertainties. Neglecting these uncertainties is not a good practice and including them could reduce the process costs. In MPC, uncertainties are handled either with a robust formulation or stochastic formulation. A stochastic formulation includes the probabilistic information about the uncertainty and allows a trade-off between performance and satisfaction of constraints. Apart from this, water demands are stochastic by nature and therefore, we formulate a stochastic model predictive control which is investigated in the thesis.

Another aspect of the distribution layer in water network is the hierarchical nature in its operation. The economic optimisation provides set-points for the actuators which are achieved with local controllers (like PID or LQR controller). These controllers operate at lower sampling time. Due to historical reasons these networks have a complex topology and designing a centralised controller is cumbersome. It is time to call for decentralised strategies based on spatial and temporal decomposition of the overall dynamics so as to leverage the high computational cost of a centralized MPC.

1.2 Scope and Contributions of this thesis

In this thesis, we identify some critical and limiting challenges in the control of water networks: (i) need for closed-loop control policies (ii) uncertainty in demand forecast (iii) meeting water demands uninterruptedly and (iv) computational burden for a centralised controller. First we employ a MPC integrated with online demand forecast to manage the DWN. This strategy has a feedback from the network and guarantees to provide water that can meet the actual demands at every time instance. In order to address the uncertainty in the demand forecast, we extend the MPC to stochastic version that takes into consideration the randomness of the demand. This problem is of the form of an infinite dimensional optimisation problem, so we approximate this to a finite dimensional large-scale problem and employ convex optimisation to solve the stochastic MPC problem efficiently.

Publications

This work lead to the following publications

1. A.K. Sampathirao, P. Sopasakis, A. Bemporad and P. Patrinos, "Proximal quasi-Newton methods for scenario -based stochastic optimal control", IFAC 2017, submitted.
2. A.K. Sampathirao, P. Sopasakis, A. Bemporad and P. Patrinos, "Stochastic predictive control of drinking water networks: large-scale optimisation and GPUs", (provisionally accepted) IEEE Control Systems Technology, preprint: `http://arxiv.org/abs/1604.01074`.
3. A.K. Sampathirao, P. Sopasakis, A. Bemporad and P. Patrinos, "Proximal quasi-Newton methods for scenario-based stochastic optimal control", EUCCO 2016, Leuven, Belgium.

4. A.K. Sampathirao, P. Sopasakis, A. Bemporad and P. Patrinos, "Distributed solution of stochastic optimal control problems on GPUs", in Proc. 54th IEEE Conf. on Decision and Control, Osaka, Japan, 2015, pp. 7183–7188, https://eprints.imtlucca.it/2779/1/APG_GPU.pdf.
5. A.K. Sampathirao, P. Sopasakis and A. Bemporad, "Decentralised hierarchical multi-rate control of large-scale drinking water networks". In: 9th International Conference on Critical Information Infrastructures Security, Oct. 13-15 2014, Limassol, Cyprus, pp. 1-12, 2014. http://link.springer.com/chapter/10.1007%2F978-3-319-31664-2_6
6. A.K. Sampathirao, J.M. Grosso, P. Sopasakis, C. Ocampo-Martínez, A. Bemporad, and V. Puig, "Water demand forecasting for the optimal operation of large-scale drinking water networks: The Barcelona case study", in Proc. 19th IFAC World Congress, Cape Town, South Africa, 2014.
   http://www.ifac-papersonline.net/Detailed/68585.html

1.2.1 Working papers

1. A.K. Sampathirao, P. Sopasakis, A. Bemporad and P. Patrinos, "Parallelizable Quasi-Newton Methods for Stochastic Optimal Control", working paper.

Software

The algorithms developed during this thesis are available on public repositories for the interested reader to experiment with the developments presented in this thesis.

1. The matlab version for the dual proximal gradient algorithm to solve the stochastic optimal control is available at
   https://github.com/ajaykumarsampath/APG--stochastic-optimal-control
2. The CUDA implementation of the dual proximal gradient algorithm is available at
   https://bitbucket.org/ajaykumarsampath/tb-gpad

3. The matlab version for the LBFGS-FBE algorithm to solve the stochastic optimal control discussed in Chapter 6 is available at `https://github.com/ajaykumarsampath/L-BFGS-update`

The implementation of dual proximal method in CUDA for the DWN and the closed-loop simulations will be provided upon request.

Other reports

The case study for this thesis is the Barcelona city network which is taken as a part of the EU FP7 research project EFFINET "Efficient Integrated Real-time monitoring and Control of Drinking Water Networks". During the course of this project I co-authored four deliverable reports (SSG$^+$13; OMPG$^+$14; SSG$^+$14; SFS$^+$14) which have been successfully approved by the European Commission.

1.3 Structure

Chapter 2 — Worst-case control of DWN

Chapter 2 introduces the main challenges in management of Drinking Water Networks (DWNs) and derive its dynamic model using the mass-balance equation. We identified water demands as the main source of uncertainty in the network. Instead of using historical demand data, real-time forecasts improve the operation of the DWN. The demand time series are governed by multiple seasonalities and we propose various time series models to perform forecasts — namely, seasonal ARIMA, BATS and SVM. These models— along with a stochastic component which describes their uncertainty — are later used in Chapter 4.

The DWN of Barcelona city is selected as our case study. The time-series models developed for this system are analysed in terms of their predictive ability taking also into consideration the complexity of the model. Taking into account the worst-case prediction errors we formulate a robust worst-case model predictive control problem and we propose a computationally feasible formulation.

Chapter 3 — SMPC & Numerical Solution

In the first part of Chapter 3 we introduce the concept of stochastic model predictive control (SMPC). In SMPC, the uncertainty is seen as a stochastic process and incorporates the information about its probability distribution in the problem formulation. However, typical SMPC formulations — unless under restrictive assumption — this lead to infinite dimensional optimisation problems which, in most cases, are computationally intractable. We recast this as finite-dimensional problem using a discrete representation of the involved random process — scenario trees. The outcome of this is a rather large-scale yet well-structured optimisation problem.

In the second part we formulate the *Fenchel dual* optimisation problem on which we apply the accelerated proximal gradient algorithm elaborating on how the involved operations can be parallelised to a great extent. This parallel algorithm is implemented on graphics processing units (GPUs) and computational performance is investigated for a numerical example.

Chapter 4 — SMPC for DWNs

Whereas in Chapter 3 we formulate a generic algorithmic framework for solving stochastic optimal control problems, in Chapter 4 we apply it to controlling a real drinking water network — that of the city of Barcelona — using the models and concepts laid out in Chapter 2.

We analyse the closed-loop performance with a stochastic MPC controller for various scenario tree structures and we discuss the advantages of the proposed control scheme *vis-à-vis* state-of-the-art methods.

Chapter 5 — Spatial & Temporal Decomposition

In Chapter 5, we address the problem of the *spatial decomposition of networks of dynamical systems (partitioning into subsystems) and their temporal decomposition (multiple time scales) across two control layers*. Indeed, DWNs have a multi-scale hierarchical control architecture — an upper control layer that provide the set-point based on economic considerations and a lower layer that track the set-point. In this chapter we explore decentralised control techniques for systems with multi-rate hi-

erarchical control architecture under hard state-input constraints. The large-scale dynamical system is partitioned into interconnected subsystems with state and input constraint and the worst-case interactions between subsystems are modelled and accounted for in a robust manner.

Chapter 6 — Quasi-Newton methods

First-order methods like the proximal-gradient methods we present in Chapters 4 and 3 are suitable for medium precision. First-order methods are rather sensitive to bad conditioning which is likely to compromise their convergence speed and accuracy, often despite preconditioning. A remedy to these pathologies would be the introduction of second-order information in the smooth part of the cost, as in Sequential Quadratic Programming (SQP) aiming at a Q-superlinear convergence rate, but this approach would destroy the separability properties of the method and would call for an inner iterative algorithm.

Quasi-Newton methods such as the celebrated limited-memory BFGS (L-BFGS) method are successfully used for large-scale smooth unconstrained problems and enjoy good convergence rate properties. In this presentation we will show a different approach based on the *forward backward envelope* (FBE) function, which — under certain assumptions — is real-valued, continuously differentiable with Lipschitz gradient and (strongly) convex. The minimisers of the FBE are the minimisers of the original optimisation problem, so we may apply methods for smooth unconstrained minimisation such as L-BFGS. Additionally, at every iteration the proposed algorithm incurs a computational cost which is similar to that of a forward- backward step.

Overall, the proposed algorithm is parallelisable and to some extent we reuse parallelisation concepts from Chapters 4 and 3.

Chapter 2

Drinking water networks: modelling and control

Drinking Water Networks (DWN) are large-scale multiple-input multiple-output systems with uncertain disturbances (such as the water demand from the consumers) and involve components of linear, nonlinear and switching nature. Operating, safety and quality constraints deem it important for the state and the input of such systems to be constrained into a given domain. Moreover, DWNs' operation is driven by time-varying demands and involves a considerable consumption of electric energy and the exploitation of limited water resources. Forecasts about the demands provides a means to increase the reliability of the network. Traditional operators rely on historical demands and are not integrated in real-time operation of the network.

This chapter explores various state-of-the-art methods for demand forecasting, such as Seasonal ARIMA, BATS and Support Vector Machine, and presents a set of statistics to validate these models. These models are integrated with a Model Predictive Control (MPC) strategy to generate an online control policy. The constraints in this MPC formulation are shrunk to accommodate the worst case forecast er-

rors. This strategy allows for an online optimal flow management of a DWN. These results presented in this chapter have appeared in (SGS+14).

2.1 Introduction

2.1.1 Motivation

Drinking Water Networks (DWN) are large-scale, multiple-input multiple-output systems whose operation is liable to be subjected to a set of operating, safety and quality-of-service constraints while at the same time their dynamics is affected by disturbances of stochastic nature (see (BU94; GOMP13)). All these characteristics render their control a challenging problem. The optimal management of DWNs is a complex task with outstanding socio-economic and environmental implications and has received considerable attention by the scientific community.

One popular way to minimise the operational cost is to formulate the problem as optimal scheduling of pumps best known in literature as *pump scheduling problem*. Such open-loop approaches are known since the 80's and a review of the early optimisation methods for pump scheduling is given in (OL94). Various techniques are proposed in the literature to solve the pump scheduling problem; dynamic programming (ZS89), linear programming (SS97), mixed integer nonlinear programming (BBMJ+13) and evolutionary algorithm based methods (MP04). Using the 24 hour demand forecast, the optimal pump scheduling is solved for a small system using dynamic programming in (ZS89) which cannot be recreated for a large system. Sherali *et al.* (SS97) uses the linearisation based technique to construct a tight linear programming relaxation of the pump scheduling problem and then uses a branch-and-bound algorithm to obtain global optimal solutions. In (BBMJ+13) the problem was formulated as a mixed-integer nonlinear program to account for the on/off operation of the pumps. Heuristic approaches using evolutionary algorithms, genetic algorithms, and simulated annealing have also appeared in the literature (MP04).

The control policy obtained from all the above methods is an "open-loop policy" as it is based on an offline optimisation and does not include the feedback from the network during its operations. In most

cases these problem cannot be solved online and not suitable for real-time control. It is pointed in (SO96) that feedback to the decision maker will improve the decision making in DWNs.

Model Predictive Control (MPC) (RM09; QB03) is an advanced control technology that solves a finite-horizon constrained optimal control problem online at every time instance and implicitly induce a feedback in the system. It is known that MPC is well suited for large-scale MIMO (multiple-input multiple-output) systems and is also amenable to structural changes of the dynamical model of the considered system. Based on the MPC technology strategies are developed for real-time management of the drinking water networks in (OMPC$^+$09; BOMPB10). Additional constraints for water quality can be included with MPC controller (BMLL$^+$04).

However these methods did not account for a crucial source of uncertainty–*demand uncertainty*. The control policy is based on the demand forecast generated from the historical data. Using such forecast is limiting as they cannot adjust with the previous observations. So we propose to build a time-series model that forecast the future demands using the current and the previous observations. In general, water usage can vary in both the long-term and the short-term, usually exhibiting time-based patterns for different areas. Hence, a better understanding of the characteristics of the time-series is necessary so as to perform accurate forecasts of water demand and have an optimal closed-loop performance. The overall objective of DWNs managers is to provide a reliable water supply in the cheapest way, guaranteeing availability and continuity of the service with a certain probability and without delay under some operating conditions, specific environments and uncertain events.

Therefore, the operation of a DWN is strongly conditioned by the uncertain water demand, which follows a non-stationary dynamics (see (QPC$^+$06)). In this chapter, three well-established time series modelling methodologies are employed to capture the dynamics of water demand, namely a Seasonal Auto-Regressive Integrated Moving-Average (sARIMA) model, a Box-Cox transformation (BATS) model developed by (LHS11), and a Support Vector Machine (SVM) model. All these models are statistically validated and are accompanied by an estimation of their prediction error. All these approaches proved to be adequate for the modelling of the demand.

Afterwards, Model Predictive Control (MPC) is used for computing optimal decisions regarding the operation of a DWN taking into account the management criteria and the various operating, safety and physical constraints of the problem. Moreover, a performance index related to water and energy costs is optimised leading to a suitable operation of the water network. The forecasting of demands and their variances are then used to propagate uncertainty in the open-loop prediction within the MPC strategy to assure better handling of constraints and proper fulfilment of the control objectives (management criteria).

2.1.2 Outline

Section 2.2 explains how the the dynamic model of the drinking water network is obtained from mass-balance equations. Afterwards, in Section 2.3 three different modelling approaches are presented to forecast water demand using historical data. Finally, Section 2.4 explains how these forecasts are integrated in the management of the DWN and tested on the DWN of Barcelona. The dynamic model and the forecasting models developed here are used later in Chapter 4.

2.1.3 Notation

In this chapter, $\mathbb{N}_{[k_1,k_2]}$ denotes the set of natural numbers between k_1. Let k_2, $\mathcal{P} \subset \mathbb{R}^n$ be a polytope and $A \in \mathbb{R}^{m\times n}$ be a matrix; then it is defined $A \cdot \mathcal{P} := \{y \in \mathbb{R}^m : y = Ax; x \in \mathcal{P}\}$. Let $\mathcal{P}$, $\mathcal{Q}$ be two polytopes in $\mathbb{R}^n$; the Pontryagin difference of these polytopes is the polytope $\mathcal{P} \ominus \mathcal{Q} := \{z \in \mathbb{R}^n, \exists q \in \mathcal{Q}, \text{ such that } z+q \in \mathcal{P}\}$. The Minkowski sum of $\mathcal{P}$ and $\mathcal{Q}$ is the polytope $\mathcal{P} \oplus \mathcal{Q} := \{z = p + q : p \in \mathcal{P}, q \in \mathcal{Q}\}$.

2.2 Modelling

A DWN generally involves a number of actuators; namely valves and pumps (the latter being grouped into pumping stations). Let $u \in \mathbb{R}^{n_u}$ denote the vector of all these inputs. The state of the DWN can be fully observed by measuring the reservoir volumes. This will be denoted by $x \in \mathbb{R}^{n_x}$. The water demand from the various outlets of the distribution network will be considered to be a stochastic variable

$d \in \mathbb{R}^{n_d}$. At each time instant t, the current demand $d(t) = d(t \mid t)$ is measured while subsequent demands $d(t+j \mid t)$ can be forecast (as random variables) through appropriate demand prediction models.

The mass balance equations of the network lead to a continuous-time dynamical model in the form of a stochastic differential equation:

$$\mathrm{d}x(t) = f(x(t), u(t))\mathrm{d}t + g(d(t))\mathrm{d}B_t, \tag{2.1}$$

where $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$, $g : \mathbb{R}^{n_d} \to \mathbb{R}^{n_x}$ and B is a Wiener stochastic process. In this equation, a deterministic part and a stochastic (additive) component is identified. The aforementioned model is linear, i.e. $f(x,u) = Ax + Bu$ and $g(d) = G_d d$ where A, B and G are matrices of appropriate dimensions. In the following sections it will be explain how such a model is derived. The exact linearisation of (2.1) with sampling period T_s yields the following discrete-time stochastic dynamical model:

$$x_{k+1} = Ax_k + Bu_k + G_d d_k \tag{2.2}$$

for $k \in \mathbb{N}$ so that $x(kT_s) = x_k$ and $u(t) = u_k$ for $t \in [kT_s, (k+1)T_s)$. In the sequel it is explained how such a model can be derived based on principles of mass transfer.

2.2.1 Hydraulic modelling

The dynamics of DWNs have been studied in depth in the last two decades (see (BU94; OMPC$^+$09)). The hydraulic model of the DWN consists of the mass-balance equations for the water in every reservoir $i = 1, \ldots, N_r$ and distribution node $j = 1, \ldots, N_n$. In simple words, the mass-balance equation of a single tank is the rate of change of the water level in the tank is the difference between inflow and outflow. This equation is based on mass conservation and it neglects pressure-head required to achieve the flow into the tank. It is always assumed that there is a pressure combination that can meet the flow requirement.

Let $V_i(t)$ be the volume of water inside the tank i and let $q_{i,p}^{out}(t)$ for $p = 1, \ldots, M_i^{out}$ and $q_{i,q}^{in}(t)$ for $q = 1, \ldots, M_i^{in}$ be the influx and outflux streams from and to tank i. It then follows that:

$$\frac{\mathrm{d}V_i(t)}{\mathrm{d}t} = \sum_{p=1}^{M_i^{in}} q_{i,p}^{in}(t) - \sum_{l=1}^{M_i^{out}} q_{i,l}^{out}(t). \tag{2.3}$$

At every tank it should hold that:

$$V_i^{min} \leq V_i \leq V_i^{max}, \tag{2.4}$$

where V_i^{min} and V_i^{max} are respectively the lower and the upper tank volume capacities. The upper limit is imposed so that the tank does not overflow and should in all cases be treated as a hard constraint. In most cases, the flows $q_{i,p}^{in}$ and $q_{i,l}^{out}$ are not driven by gravity, but are instead controlled by a set of pumps which come with certain technical limitations which give rise to the constraints:

$$0 \leq q_i(t) \leq q_i^{max}. \tag{2.5}$$

In a DWN, a *node* is a meeting point of three or more pipes. There, the mass balance yields the static equality constraint:

$$\sum_{p=1}^{M_j^{in}} q_{j,p}^{in}(t) = \sum_{l=1}^{M_j^{out}} q_{j,l}^{out}(t) \tag{2.6}$$

for $j = 1, \ldots, N_n$ where N_n is the number of nodes.It is assumed here, that all flows in the network can be modelled with a set of unidirectional positive flows, which translates to the constraint $q_i(t) \geq 0$, for every flow. Certain outgoing flows q_i are actual demands from the supply network and, as such, they are stochastic variables. When the above mentioned mass balance equations put together to arrive at the following expression in discrete-time:

$$x_{k+1} = A_d x_k + B_d u_k + G_d d_k, \tag{2.7}$$

where $x \in \mathbb{R}^{n_x}$ is the state vector corresponding to the volumes of water in the storage tanks, $u \in \mathbb{R}^{n_u}$ is the vector of manipulated inputs and $d \in \mathbb{R}^{n_d}$ is the vector of uncertain demands. Mass preservation gives rise to the following input-disturbance coupling equation:

$$Eu_k + E_d d_k = 0, \tag{2.8}$$

where E and E_d are matrices of proper dimensions.

In this context, the state and input constraints can be rewritten as:

$$u_k \in \mathcal{U} := \{u \in \mathbb{R}^{n_u} \mid u^{min} \leq u \leq u^{max}\}, \ \forall k \in \mathbb{N} \tag{2.9a}$$

$$x_k \in \mathcal{X} := \{x \in \mathbb{R}^{n_x} \mid x^{min} \leq x \leq x^{max}\}, \ \forall k \in \mathbb{N} \tag{2.9b}$$

Both $\mathcal{X}$ and $\mathcal{U}$ are compact polytopes. The flow model that results from this analysis is a Linear Time-Invariant (LTI) discrete-time dynamical model with linear constraints which perfectly fits into the control framework of linear MPC.

The above formulation has been widely used in the formulation of model predictive control problems for DWNs (GOMPJ14; OMFBP10).

2.3 Demand forecasting models

The reliable modelling and the ability to predict the upcoming water demands from every output node of the network is an essential task for the design of proper controllers for the DWN. The non-stationary of the demand time series along with the presence of multiple seasonal patterns calls for state-of-the art modelling approached that can capture such complex dynamics.

In this section, three different modelling approaches are presented to model the water demand from a DWN case study. These models are trained using the same dataset of 2700 demand measurements from the DWN of Barcelona, out of 8760 data points that are available. The rest is used as an external test-set for validating these models. These data were provided by AGBAR (Aguas de Barcelona, s.a.), which is the company that manages the Barcelona DWN)[1]. A quick glance at the historical demands will revile that demands have a daily and a weekly pattern and suggests to use forecast with seasonal models.

2.3.1 Seasonal ARIMA time-series models

ARIMA models are widely used as they can capture complex linear dynamics of stationary processes or processes that become stationary after one applies the difference operator finitely many times. ARIMA models put more emphasis on the recent past of the time series they intercept, so, they are considered to be suitable for short-term forecasting (see (BJR94)). Here the Seasonal ARIMA models are considered

[1]The patterns of the water demand which are used here were synthesised from real values measured at the Barcelona DWN in 2007. The data were obtained within the FP7-funded EU project EFFINET. See `http://effinet.eu`.

and seasonally integrated with seasonal AR (Auto-Regressive) terms which describe well time series that follow a periodic pattern.

In order to derive ARIMA models, the following notation is introduced. For a time series d_k, let $Ld_k = d_{k-1}$ be the *backward shift operator*. For $i \geq 2$, define $L^i = LL^{i-1}$, then, $L^i d_k = d_{k-i}$. Denoting by 1 the identity operator $1z_t = z_t$ and $\nabla = 1 - L$, it follows that $\nabla d_t = d_t - d_{t-1}$. Let α_t be a white noise process, i.e., a time series such that $\mathbb{E}\alpha_t = 0$, $\alpha_t \sim N(0, \sigma_\alpha^2)$ and $\mathrm{cov}(a_t, a_{t+k}) = 0$ for all $k \neq 0$.

Let ϕ be a polynomial of L of order p with unitary constant term, symbolically $\phi \in \mathrm{K}_p^1[L]$, i.e., $\phi(L) = 1 - \phi_1 L - \phi_2 L^2 - \ldots - \phi_p L^p$ and let $\psi \in \mathrm{K}_q[L]$ be a polynomial with of the form $\psi(L) = 1 + \psi_1 L + \ldots + \psi_q L^q$. Then, an $\mathrm{ARIMA}(p, d, q)$ model has the form:

$$\phi(L)(1 - L)^d \bar{d}_t = \psi(L)\alpha_t, \tag{2.10}$$

where $\bar{d}_t = d_t - \mu$ where $\mu = \mathbb{E}d_t$ is the (known) expected value of d_t. In certain cases, ϕ and ψ are assumed to have some of their coefficients fixed to 0. Then, if χ_p and χ_q are the sets of non-zero coefficients of ϕ and ψ respectively (with $\max \chi_p = p$ and $\max \chi_q = q$), the respective ARIMA model is denoted by $\mathrm{ARIMA}(\chi_p, d, \chi_q)$.

The water demand exhibits also a periodic variation which in time series analysis is referred to as *seasonality*; this seasonality follows some calendar trend such as daily, weekly or monthly. An ARIMA process may have a single or multiple seasonalities. This periodicity is captured by a term of the form $1 - L^s$ (that acts on $\bar{d}_t$) where s is the seasonality of the process and the respective model is denoted as $\mathrm{SARIMA}(\chi_p, d, \chi_q; s)$. An ARIMA process with single seasonality s is denoted by $\mathrm{ARIMA}(p, d, q) \times (P, D, Q)_s$. A single-season seasonal ARIMA model has the following representation:

$$\phi(L)(1 - L)^d \phi_s(L^s)(1 - L^s)^D \bar{d}_t = \psi(L)\psi_s(L^s)\alpha_t, \tag{2.11}$$

where the order of the polynomial ϕ_s is P and the order of ψ_s is Q and they have the following form:

$$\phi_s(L^s) = 1 - \beta_1 L^s - \beta_2 L^{2s} - \ldots - \beta_P L^{Ps} \tag{2.12a}$$

$$\psi_s(L^s) = 1 + \zeta_1 L^s + \zeta_2 L^{2s} + \ldots + \zeta_Q L^{Qs} \tag{2.12b}$$

Such seasonal ARIMA models as (2.11) can be combined with seasonal AR and seasonal MA elements giving rise to a more flexible setting. A multi-seasonal MA polynomial operator $\Phi(L) = 1 + \Phi_{s_1} L^{s_1} +$

$\ldots + \Phi_{s_P} L^{s_P}$ whose exponents form the set $\chi_P^{\Phi} = \{s_1, s_2, \ldots, s_P\}$ where $s_i < s_{i+1}$ for all $i = 1, \ldots, P-1$. The elements of χ_P^{Φ} are called MA seasons and they are not assumed to be multiples of some elementary season. A multi-seasonal AR component represented by the polynomial $\Psi(L) = 1 - \Psi_{t_1} L^{t_1} - \ldots - \Psi_{t_Q} L^{t_Q}$ for which the set of seasonal AR lags χ_Q^{Ψ} is defined accordingly.

An s-seasonal ARIMA model seasonally integrated with a χ_P^{Φ}-seasonal MA and a χ_Q^{Ψ}-seasonal AR component will be hereinafter denoted by:

$$\mathrm{SARIMA}(\chi_p, d, \chi_q; s) \times \mathrm{SAR}(\chi_P^{\Phi}) \times \mathrm{SMA}(\chi_Q^{\Psi}).$$

The problem of parameter estimation of ARIMA models is formulated either as a Least Squares problem or as a (nonlinear) Maximum Likelihood problem. Model parameter estimates are always accompanied by their variance and their t-statistic; what one optimally expects is low variance and high statistical significance justifying the choice of the underlying model. The Log-Likelihood as well as the Root Mean Square (RMS) value of the residuals will be used as a measure of the goodness-of-fit of the model. Here the estimation of model parameters was carried out using the function `estimate` of MATLAB's econometrics toolbox.

One of the principle factor to be considered in selecting the model is *parsimony*; model with smallest number of coefficients to fit the availability data produce better forecasts. Other criteria useful in evaluating the model are quality of the coefficients (statistical significance), predictive power, and stationarity. Keeping these features in mind, a set of models was created for each of the 88 demands of the DWN of Barcelona. Indicatively, the following model for the demand form the output node `c450BEG` of the Barcelona DWN:

$$\mathrm{ARIMA}(\{1:4, 6:9\}, 1, \{1:13, 15, 17\}; 168) \times \mathrm{SAR}(\{168, 336\}), \tag{2.13}$$

which was trained using 2700 samples and was formulated as a maximum likelihood problem. All the parameters of this model were determined with high statistical significance (based on their t-statistic).

For the evaluation of the predictive power of each model, the water demand forecast throughout a horizon of $H_p = 24\mathrm{h}$ ahead is considered to demonstrate how each model performs in the long run. Let

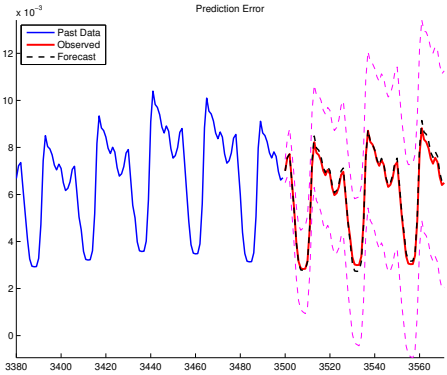

Figure 1: Stochastic forecasts using the SARIMA model (2.13). The purple dashed thin lines represent the 99% upper and lower bounds computed by a Monte Carlo simulation using 10^5 seeds.

$\hat{d}_{k+j|k}$ be the predicted expected demand of water for the future time instant $k+j$ performed at time k. Let d_{k+j} be the actual value of the demand. The total prediction error along the period $[k, k+H_p]$ is quantified by the prediction mean squared error (PMSE), defined as:

$$\mathrm{PMSE}_{H_p,k} = \frac{1}{H_p}\sum_{i=1}^{H_p}(\hat{d}_{k+i|k} - d_{k+i})^2, \tag{2.14}$$

and its square root is the PRMSE, i.e., $\mathrm{PRMSE}_{H_p,k} = \sqrt{\mathrm{PMSE}_{H_p,k}}$. Model (2.13) was found to have $\mathrm{PMSE}_{24} = 0.0158$ (with st. dev. 0.0049) and a $\mathrm{PRMSE}_{24} = 0.1311$ on average[2].

If the fitted SARIMA model is adequate, then the residuals should follow approximately a distribution with zero mean and should be uncorrelated. A special statistical test, namely the Ljung-Box test, has been devised to test this hypothesis. This model (2.13) passed the Ljung-Box test for uncorrelated residuals with a p-value of 0.2908, the

[2]Based on external data and using 150 samples. The computation was carried out against an external test-set, not available to the model.

value of the Ljung-Box statistic being 22.96 with critical value 31.41. This model was selected among a set of many other models using the Akaike Information Criterion given by $\mathrm{AIC} = \ln(\hat{\sigma}_\alpha^2) + 2k/T$, where $\hat{\sigma}_\alpha^2$ is the statistical estimate of σ_α^2, k is the number of parameters of the model and T is the number of observations used for the estimation of the model. This criteria penalise the estimated variance of the residuals and the complexity of the model aiming to choosing the simplest model that achieves good fit. For the model in (2.13) there was obtained an $\mathrm{AIC} = -8.5044$.

Conclusively, model (2.13) interpolates very well the demand time series, it has high predictive ability, its residuals are uncorrelated, it is determined with high statistical significance and good predictive power.

2.3.2 Box-Cox transformation, ARMA Errors, trends and seasonality (BATS) Modelling

BATS models were introduced by De Livera et al. (LHS11) and have proven to be well-suited for modelling time series with multiple seasonal patterns and complex dynamics.

Let d_k, $k \in \mathbb{N}$, denote an observed time series of any water demand, and $d_k^{(\omega)}$ its Box-Cox transformation with the parameter ω. The transformed series is then decomposed into an *irregular* component h_k, a *level* component l_k, a *growth* component b_k and possible seasonal ones $s_k^{(i)}$ with seasonal frequencies m_i, for $i = 1, \ldots, P$, where P is the total number of seasonal patterns in the series. The irregular component h_k is described by an $\mathrm{ARMA}(p, q)$ process with parameters ϕ_i for $i = 1, \ldots, p$ and θ_i for $i = 1, \ldots, q$, and an error term ε_k which is assumed to be a Gaussian white noise process with zero mean and constant variance σ^2. The smoothing parameters, given by α_d, β_d, $\gamma_{d,i}$ for $i = 1, \ldots, P$, determine the extent of the effect of the irregular component on the states l_k, b_k, $s_k^{(i)}$ respectively. The equations of the

model are:

$$d_k^{(\omega)} = \begin{cases} \frac{d_k^{(\omega)}-1}{\omega}, & \omega \neq 0, \\ \log(d_k), & \omega = 0, \end{cases} \tag{2.15a}$$

$$d_k^{(\omega)} = l_{k-1} + \phi b_{k-1} + \sum_{i=1}^{P} s_{k-m_i}^{(i)} + h_k, \tag{2.15b}$$

$$l_k = l_{k-1} + \phi b_{k-1} + \alpha_d h_k, \tag{2.15c}$$

$$b_k = \phi b_{k-1} + \beta_d h_k, \tag{2.15d}$$

$$s_k^{(i)} = s_{k-m_i}^{(i)} + \gamma_{d,i} h_k, \tag{2.15e}$$

$$h_k = \sum_{i=1}^{p} \varphi_i h_{k-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{k-i} + \varepsilon_k. \tag{2.15f}$$

The decomposition of the time series into such components and the use of them for modelling and forecasting offers also a qualitative insight into the dynamics of the process.

A set of BATS models was trained for the demand time series by maximising their log-likelihood according to (DLHS10) using two seasonal patterns: one daily and one weekly. The best BATS model for each demand time series was chosen on the basis of the Akaike Information Criterion given by $\mathrm{AIC} = L^\star + 2k$, where $L^\star$ is the maximum log-likelihood of the model, and k is the number of tunable parameters of the model.

2.3.3 RBF-based support vector machine

Other modelling approaches involving non-linear functions have been proposed in the literature and can be employed for the modelling of the demand time series. Neural Networks (GRP07), Discrete Wavelet Transform methods (Sol02), Support Vector Machine methods (TvBdW$^+$03) are only some of the methods available in the literature. In this section we propose the use of a time series modelling approach using a Support Vector Machine (SVM) model with a Radial Basis Function (RBF) with $\gamma = 0.015$.

The problem was formulated as an ϵ-SVM (see (CV95)) with $\epsilon = 10^{-5}$ and was solved using the celebrated library libSVM by (CCCJ11). The parameter C of the cost function of the problem was set to 1000.

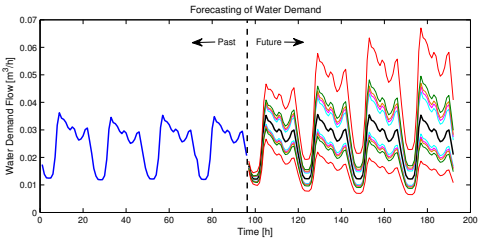

Figure 2: Forecasting of water demand using the proposed BATS model. (Black thick line) expected forecast, (surrounding lines) confidence intervals for 90%, 95%, 97%, 99% and 99.9999% confidence levels.

The explanatory variables used were 200 past values of the demand and a set of binary calendar variables as follows: Let m_i be 1 if the day when the measurement was taken was the i-th day of the week (for $i = 0, \ldots, 6$) and 0 otherwise. Let h_j be the corresponding variable referring to the hour of the day for $j = 0, \ldots, 23$. In this way, the information about the seasonal pattern of the time series is encoded. All the features of the dataset were scaled to the interval $[0, 1]$.

A 10-fold cross-validation of the model produced a $q^2 = 0.9952$. It was found that $\mathrm{PMSE}_{24} = 0.0065$ (with st. dev. 0.0051) and $\mathrm{PRMSE}_{24} = 0.0743$ on average based on 150 samples. The performance of the SVM-based predictor and the stringency of the confidence intervals for its forecasts is illustrated in Figure 3.

2.3.4 Comparison and evaluation

The three proposed models are concisely compared in Table 1 in regard to their complexity and predictive ability against external data.

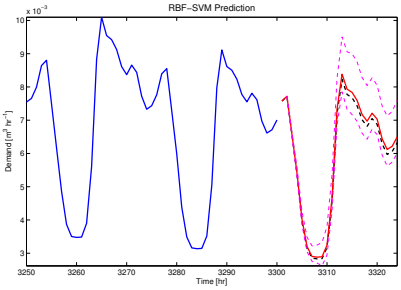

Figure 3: Forecasting of the demand time series using the proposed RBF-SVM model. (Blue line): past demand data, (Red line): Actual demand, (Black dashed line): SVM predictions, (Light magenta dashed lines): 99% confidence intervals calculated by Monte Carlo simulations using 5000 seeds.

Table 1: Comparison of the Predictive Models

Performance Index	sARIMA	BATS	RBF-SVM
Average PMSE_{24}	0.0158	0.0043	0.0065
Average PRMSE_{24}	0.1311	0.0584	0.0743
No. Parameters	25	26	229

2.4 Model predictive control for the operating management of DWN

Managing drinking water network is a complex task as it involves balancing various social and economical objectives. In this section, it is discussed how these objectives are translated into cost function and formulated a Model Predictive Control (MPC) problem.

2.4.1 Model predictive control

Model Predictive Control (MPC) also popular as Receding Horizon Control (RHC) is an optimisation-based control technique that has demonstrated phenomenal success in various industries. The reasons behind this success can be attributed to its conceptual simplicity, ability to cope with complex system dynamics, handle constraints on state and inputs and accommodate conflicting interests. The theory and applications have evolved a lot over the past three decades and several monographs are available on MPC (see Rawlings and Mayne (RM09), Camacho and Alba (CBA07) and Grüne and Pannek (GP11)).

The most vital components of MPC is the dynamics model of the system to predict the future behaviour. Using this model, MPC solves a finite-horizon open-loop optimal control problem to produce a finite control sequence given by $\pi = \{u_{k|k}, u_{k+1|k}, \dots u_{k+N-1|k}\}$ where $u_{k+i|k}$ is the control action predicted time $k+i$ from the time k. The first element of the control sequence is applied to the system discarding the rest of the sequence. The same computation is repeated at the next sampling time. This aspect of online solving an optimal control problem in MPC results in an *implicit* feedback action and offers resilience to system uncertainties. This procedure can be summarised in Algorithm 1.

2.4.2 Control objectives

The cost function used in MPC formulation reflects the control objective of the system. In drinking water networks, this cost comprised of production and transportation cost, deviation of the volume of water in storage tanks from the prescribed operating limits and delivering

Algorithm 1 Receding horizon control policy

1. Measure the state at x_k at time instance k.
2. Solve the finite-horizon optimal control problem to generate $\pi^\star = \{u^\star_{k|k}, u^\star_{k+1|k}, \dots u^\star_{k+N-1|k}\}$.
3. Apply $u^\star_{k|k}$ to the system and begin from step 1 for the next time instance.

smooth control actions. These costs were quantified following the approach reported by (BOMPB10).

Water production and transportation cost

This cost is associated with treatment of drinking water, regulations and electricity consumptions and is quantified by the cost function:

$$\ell^w(u_k, k) := W_\alpha(\alpha_1 + \alpha_{2,k})' u_k, \tag{2.16}$$

where α_1 is a vector related to the production costs of water according to the selected source (treatment plant, dwell, etc.), and $\alpha_{2,k}$ is associated with the pumping cost for the transportation of the water through certain paths. W_α is a proper scaling factor.

As the price of electricity is non-constant throughout the day, $\alpha_{2,k}$ is time-dependent. This price model depends on the agreement between the electricity provider(s) and the network operator. This agreement could be a fixed price for certain time or a price that follow a certain way whose pattern is known to the operator.

Safety storage

It is important that the stored volume of water in the tanks remain into certain prescribed levels. This safety volume is to compensate unexpected and unpredictable events and also to maintain pressure levels in the network. For this reason, a *safety-storage cost* is included. This cost is given by:

$$\ell^s(x_k) = s_k' W_x s_k, \tag{2.17}$$

where $s_k = \max\{0, x_s - x_k\}$ and $x_s = \beta x_{max}$ where $\beta \in (0, 1)$ is a factor that represents the safety volume with the storage capacity.

$W_x \in \mathbb{R}^{n_x \times n_x}$ is a positive semi-definite matrix with non-negative entries. This cost is non-quadratic and also replaced it with the term:

$$\ell^S(\xi_k) = \xi_k' W_x \xi_k, \tag{2.18}$$

where $\xi_k = \xi_k(x_k) \geq x_s - x_k$ accompanied by the convex constraint $\xi_k \geq 0$.

Smooth operation

In order to avoid intermittent, rapid oscillating actuation, which may lead to glitches in operation and mechanical failures, the variations of the control variables between consecutive time instants. This is the *smooth operation cost* is defined as:

$$\ell^\Delta(\Delta u_k) = (\Delta u_k)' W_u \Delta u_k, \tag{2.19}$$

where $W_u \in \mathbb{R}^{n_u \times n_u}$ is a square positive definite matrix, and $\Delta u_k = u_k - u_{k-1}$.

Total cost

The stage-cost at a time instance k incurred during the operation of DWN is the summation of the all the above costs and given as

$$\ell(x_k, u_k, u_{k-1}, k) = \ell^w(u_k, k) + \ell^\Delta(\Delta u_k) + \ell^S(\xi_k). \tag{2.20}$$

The total cost function and performance index is made up of the aforementioned costs; it is:

$$V(x_k, \mathbf{u}_k, \Xi_k, \Delta \mathbf{u}_k, k) = L^w(\mathbf{u}_k, k) + L^\Delta(\Delta \mathbf{u}_k) + L^S(\Xi_k), \tag{2.21a}$$

where L^w is the *total water production cost*, L^Δ is the *total smooth operation cost* and L^S is the *total safety volume cost* given by:

$$L^w(\mathbf{u}_k, k) = \sum_{i \in \mathbb{N}_{[0,N-1]}} \ell^w(u_{k+i|k}, k), \tag{2.21b}$$

$$L^\Delta(\Delta \mathbf{u}_k) = \sum_{i \in \mathbb{N}_{[0,N-2]}} \ell^\Delta(\Delta u_{k+i|k}), \tag{2.21c}$$

$$L^S(\Xi_k) = \sum_{i \in \mathbb{N}_{[0,N-1]}} \ell^S(\xi_{k+i|k}). \tag{2.21d}$$

2.4.3 Demand forecasts

The water demand is the main source of uncertainty that affects the dynamics of the model. Using the time-series forecaster that are discussed in the Section 2.3, a nominal forecast of the upcoming N stages of the demand is predicted at time k. Lets denoted nominal demand by $\hat{d}_{k+j|k}$. Then, the actual future demands d_{k+j} — which are unknown to the controller at time k — can be expressed as

$$d_{k+j}(\epsilon_j) = \hat{d}_{k+j|k} + \epsilon_j, \forall j \in \mathbb{N}_{[0,N]}. \tag{2.22}$$

In general, this prediction errors increases with the horizon and unbounded. Neglecting this forecasting errors, could result in violating the constraints in the closed-loop. One way to avoid this is by assuming the errors are bounded and robustifying the constraints in the MPC. Let's assume that prediction error ϵ_j draw values from a compact set $\mathcal{E}_j = \{\epsilon : \epsilon_j^{min} \leq \epsilon \leq \epsilon_j^{max}\}$ which contains the origin in its interior; in practice, such sets are constructed from the prediction error data by choosing confidence intervals of 99.9%(or higher).

2.4.4 Certainty-equivalence MPC problem

MPC algorithms recently started being used for the control of DWNs (see for example (OMBPB12a)) as they guarantee certain quality of service, satisfaction of the operating constraints of the plant and optimal operation (according to the prescribed performance indices). In this section, an MPC problem is formulated taking into account the nominal demand forecasts and the estimated bounds for the prediction error.

At a given time instance k, the controller has access to the current demand measurement d_k and to a set of N future demand estimates, namely $\hat{d}_{k+j|k}$ given by the equation (2.22). Henceforth, the index $k+j|k$ will refer to a prediction at time k for the future time instant $k+j$.

Hereinafter, we use the notation $\mathbf{u}_k := (u_k, u_{k+1|k}, \ldots, u_{k+N-1|k})$ for the sequence of control actions, $\Delta\mathbf{u}_k := (u_{k+1|k} - u_k, \ldots, u_{k+N-1|k} - u_{k+N-2|k})$ for the successive differences of $\mathbf{u}$, and $\Xi_k = (\xi_{k+1|k}, \ldots, \xi_{k+N|k})$. The vector $\mathbf{d}_k$ that appears in the formulation of the MPC problem is defined as $\mathbf{d}_k = (d_k, \hat{d}_{k+1|k}, \ldots, \hat{d}_{k+N-1|k})$ and is provided

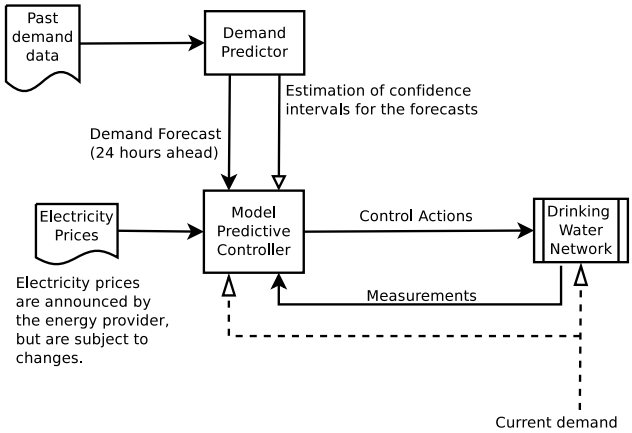

Figure 4: The closed loop system with the MPC controller and a demand estimator.

to the controller from the demand forecast module along with the vector of maximum/minimum estimated prediction errors:

$$\mathbf{e}_k = (\epsilon^{min}_{k+1|k}, \ldots, e^{min}_{k+N-1|k}, \epsilon^{max}_{k+1|k}, \ldots, e^{max}_{k+N-1|k}).$$

It is known that the prediction error $\epsilon_{k+j|k} \in \mathcal{E}_{k+j|k}$ and $\mathcal{E}_{k|k} = \{0\}$ because $d_{k|k} = d_k$ is measured. This implies that the predicted sequence of states is:

$$x_{k+j|k} = \hat{x}_{k+j|k} + \sum_{l=1}^{j} A^{l-1} G_d \epsilon_{k+l|k}, \tag{2.23}$$

where $\hat{x}_{k+j|k}$ stands for the nominal predicted state, whose dynamics, starting from $\hat{x}_{k|k} = x_k$ is described following (2.7), i.e.,

$$\hat{x}_{k+j+1|k} = A\hat{x}_{k+j|k} + Bu_k + G_d\hat{d}_{k+1|k}, \tag{2.24}$$

The MPC problem is then formulated with decision variable $\pi = \{u_{k|k}, u_{k+1|k}, \ldots u_{k+N-1|k}, \xi_{k+1|k}, \ldots, \xi_{k+N|k}\}$ as

$$V^\star(x_k, \mathbf{d}_k, k) = \min_{\mathbf{u}_k, \Xi_k} V(x_k, \mathbf{u}_k, \Xi_k, \Delta\mathbf{u}_k, k) \tag{2.25a}$$

subject to the constraints:

$$\hat{x}_{k+i|k} \in \mathcal{X} \ominus \bigoplus_{j=1}^{i} A^{j-1} G_d \mathcal{E}_{k+j|k}, \forall i \in \mathbb{N}_{[1,N]} \tag{2.25b}$$

$$u^{min} \leq u_{k+i|k} \leq u^{max}, \forall i \in \mathbb{N}_{[0,N-1]} \tag{2.25c}$$

$$\hat{x}_{k+i+1|k} = A\hat{x}_{k+i|k} + Bu_{k+i|k} + G_d \hat{d}_{k+i|k}, \quad \forall i \in \mathbb{N}_{[0,N-1]} \tag{2.25d}$$

$$Eu_{k+i|k} + E_d \hat{d}_{k+i|k} = 0, \forall i \in \mathbb{N}_{[0,N-1]} \tag{2.25e}$$

$$\xi_{k+i|k} \geq x^s - \hat{x}_{k+i|k}, \forall i \in \mathbb{N}_{[0,N]} \tag{2.25f}$$

$$\xi_{k+i|k} \geq 0, \forall i \in \mathbb{N}_{[0,H_p]} \tag{2.25g}$$

$$\hat{d}_{k|k} = d_k, \text{ and } \hat{x}_{k|k} = x_k \tag{2.25h}$$

In this formulation, only the nominal demand is used in the dynamic model of the system equation (2.25d) and popularly termed as *Certainty-Equivalence* MPC.

Here the equation (2.25b) implies that for all $i \in \mathbb{N}_{[1,N]}$, then $x_{k+i|k} \in \mathcal{X}$ as long as $\epsilon_{k+j|k} \in \mathcal{E}_{k+j|k}$ for all $j \in \mathbb{N}_{[1,i]}$. Normally, in order to perform the set operations in (2.25b), it is required to iterate over all vertices of the sets $\mathcal{E}_{k+j|k}$ (i.e., 2^{n_d} elements). However, in the context of water networks, G_d is a very sparse matrix (maximum 3 non-zero elements per row) so the complexity is such that allows the on-line implementation of the proposed algorithm. Notice that the (predicted) state is constrained in a set that is smaller than $\mathcal{X}$ and that the constraints are time-varying along the prediction horizon and are conditioned by the estimated prediction error.

2.4.5 Formulation as convex QP

The forecasts $\hat{d}_{k+i|k}$ are computed using any of the time series models described in the previous section and in the deterministic formulation only the nominal values of the forecasts are used (certainty equivalence approach), i.e., the forecasts are treated as accurate. The above problem can be formulated as a constrained quadratic problem using

the batch as follows:

$$V^{\star}(x_k, \mathbf{d}_k, k) = \min_{y} V(y) \tag{2.26a}$$

$$\text{subject to : } Gy = \gamma(\mathbf{d}_k) \tag{2.26b}$$

$$Fy \leq \phi(x_k, \mathbf{d}_k) \tag{2.26c}$$

$$y_l \leq y \leq y_h \tag{2.26d}$$

where $y \in \mathbb{R}^{n_u(H_u+1)+n_x H_p}$ is the vector $y = (\mathbf{u}_k, \Xi_k)$ and

$$V(y) = \frac{1}{2} y' H y + f' y. \tag{2.27}$$

Here, the formulas of the these matrices H,f, G, F, ϕ, γ, y_l and y_h is derived below. First the $\Delta \mathbf{u}_{k+i|k}$ is rewritten as follows:

$$\Delta \mathbf{u}_{k+i|k} = \tilde{\Delta} \cdot \mathbf{u}_{k+i|k} \tag{2.28}$$

where $\tilde{\Delta} \in \mathbb{R}^{n_u H_u \times n_u(H_u+1)}$ is the matrix:

$$\tilde{\Delta} = \begin{bmatrix} -I_{n_u} & I_{n_u} & & & \\ & -I_{n_u} & I_{n_u} & & \\ & & \ddots & \ddots & \\ & & & -I_{n_u} & I_{n_u} \end{bmatrix} \tag{2.29}$$

The total cost function $V(\cdot)$ of (2.21) in the following form:

$$V(y) = \tilde{W}_{\alpha} \mathbf{u}_k + \Xi_k' \tilde{W}_x \Xi_k + \mathbf{u}_k' \tilde{W}_u \mathbf{u}_k, \tag{2.30}$$

where

$$\tilde{W}_{\alpha} = W_{\alpha} \left(a_1 \otimes \mathbf{1}_N + \text{vec}_{i \in \mathbb{N}_{[0,N-1]}}(\mathbf{a}_{2,k+i}) \right)',$$

$\tilde{W}_x = I_N \otimes W_x$ and $\tilde{W}_u = \tilde{\Delta}'(I_{N-1} \otimes W_u)\tilde{\Delta}$. The notation $\text{vec}_{i \in \mathbb{N}_{[0,N-1]}}(\mathbf{a}_{2,k+i})$ should be interpreted as:

$$\text{vec}_{i \in \mathbb{N}_{[0,N-1]}}(\mathbf{a}_{2,k+i}) = \begin{bmatrix} \mathbf{a}_{2,k} \\ \mathbf{a}_{2,k+1} \\ \vdots \\ \mathbf{a}_{2,k+N-1} \end{bmatrix}$$

where $\mathbf{a}_{2,k+i}' \in \mathbb{R}^{n_u}$ for all $i \in \mathbb{N}_{[0,N-1]}$, therefore $\tilde{W}_{\alpha} \in \mathbb{R}^{1 \times n_u(N-1)}$.

As a result, the matrices H and f that appear in (2.26) are given by $H = 2\,\text{blockdiag}\{\tilde{W}_u, \tilde{W}_x\}$ and $f = \begin{bmatrix} \tilde{W}'_\alpha & 0'_{n_x N} \end{bmatrix}'$. Note that H is symmetric.

Let's denote $\mathbf{x}_k = (x_{k+1|k}, \ldots, x_{k+N|k}) \in \mathbb{R}^{n_x N}$. Now the dynamics of the system in given as:

$$\mathbf{x}_k = S^x x_k + S^u \mathbf{u}_k + S^d \mathbf{d}_k, \tag{2.31}$$

where $S^x \in \mathbb{R}^{n_x N \times n_x}$ and $S^u \in \mathbb{R}^{n_x (N-1) \times n_u (H_u+1)}$ are the matrices

$$S^x = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^{N-1} \end{bmatrix}, S^u = \begin{bmatrix} B & & \\ AB & B & \\ \vdots & \vdots & \ddots \\ A^{N-2}B & \cdots & B \end{bmatrix}, \tag{2.32a}$$

$$S^d = \begin{bmatrix} G_d & & \\ AG_d & G_d & \\ \vdots & \vdots & \ddots \\ A^{N-2}G_d & \cdots & G_d \end{bmatrix}, \tag{2.32b}$$

Then the inequality constraints of (2.25) are translated, in terms of $\mathbf{u}_k$, $\mathbf{d}_k$ and x_k into:

$$S^u \mathbf{u}_k \leq \mathbf{x}^{max} - S^x x_k - S^d \mathbf{d}_k \tag{2.33a}$$

$$-S^u \mathbf{u}_k \leq -\mathbf{x}^{min} + S^x x_k + S^d \mathbf{d}_k \tag{2.33b}$$

$$-S^u \mathbf{u}_k - \Xi_k \leq -\beta \mathbf{x}^{max} + S^x x_k + S^d \mathbf{d}_k \tag{2.33c}$$

$$\Xi_k \geq 0 \tag{2.33d}$$

$$\mathbf{u}^{min} \leq \mathbf{u}_k \leq \mathbf{u}^{max} \tag{2.33e}$$

where $\mathbf{x}^{max} = \mathbf{1}_N \otimes x^{max}$, $\mathbf{x}^{min} = \mathbf{1}_N \otimes x^{min}$, $\mathbf{u}^{max} = \mathbf{1}_{N-1} \otimes u^{max}$ and $\mathbf{u}^{min} = \mathbf{1}_{N-1} \otimes u^{min}$. Therefore, the matrix $F \in \mathbb{R}^{3n_x N \times n_x N + n_u (N+1)}$ and the vector $\phi \in \mathbb{R}^{3n_x N}$ in (2.26):

$$F = \left[\begin{array}{c|c} S^u & \\ -S^u & \\ -S^u & -I_{n_x H_p} \end{array}\right], \phi(x_k, \mathbf{d}_k) = \begin{bmatrix} \mathbf{x}^{max} - S^x x_k - S^d \mathbf{d}_k \\ -\mathbf{x}^{min} + S^x x_k + S^d \mathbf{d}_k \\ -\beta \mathbf{x}^{max} + S^x x_k + S^d \mathbf{d}_k \end{bmatrix} \tag{2.34}$$

and

$$y_l = \begin{bmatrix} \mathbf{u}^{min} \\ \hline 0 \end{bmatrix}, \; y_h = \begin{bmatrix} \mathbf{u}^{max} \\ \hline \infty \end{bmatrix}. \tag{2.35}$$

Let $E \in \mathbb{R}^{n_e \times n_u}$. The equality constraints of (2.25) can be written as $(I_N \otimes E)\mathbf{u}_k = (I_N \otimes E_d)\mathbf{d}_k$, hence:

$$\underbrace{\left[\begin{array}{c|c} I_N \otimes E & 0 \end{array}\right]}_{G \in \mathbb{R}^{n_e(N) \times n_u(N) + n_x N}} \cdot y = \underbrace{-(I_N \otimes E_d)\mathbf{d}_k^{\langle 1, N-1 \rangle}}_{\gamma(\mathbf{d}_k) \in \mathbb{R}^{n_e(N)}}. \tag{2.36}$$

Note that only γ and ϕ need to be updated at every iteration while all the other matrices are calculated off-line. This problem can be solved on-line, very efficiently using either gradient (PB12) or Newton method (PSS11a) or using a commercial solvers like Gurobi (Gur10) or CPLEX (ILO09) . The closed-loop system with certainty-equivalence MPC can be summarised in the Algorithm

Algorithm 2 MPC for DWN

Require: Current state information x_0 and previous control action u_{-1}.
Compute the error bounds of the forecast $\mathbf{e}_k$.
loop
1. Measure the state x_k at time instance k.
2. Predict the demand estimates $\mathbf{d}_k$ using the time-series model. If necessary update the prediction error $\mathbf{e}_k$.
3. Update γ and ϕ in the QP (2.26) for the control policy $\pi^\star = \{u^\star_{k+j|k}, \xi_{k+j+1|k}\}_{j=0}^{N-1}$.
4. Apply $u^\star_{k|k}$ to the system and begin from step 1 for the next time instance.
end loop

This approach allows for the decoupling of the predictor from the controller. The effect of the predictor's estimated error on the controller is clear from the formulation of the problem as in (2.25): the bigger the error is, the stricter the constraints and the more conservative the control policy will be.

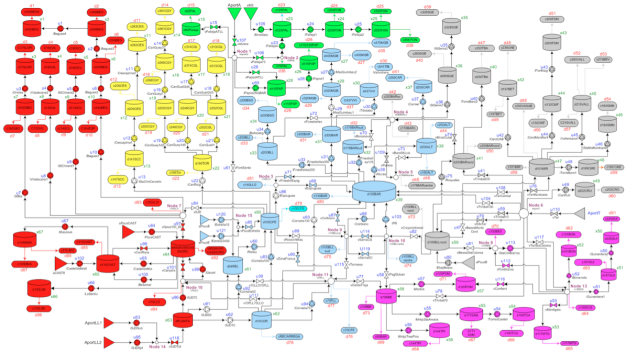

Figure 5: Structure of the DWN of Barcelona.

2.5 Case Study: Barcelona DWN

In this section, MPC with demand forecaster is investigated in the management of Barcelona drinking water network which has been reported by Ocampo *et al.* (OMBP11).

The drinking water network in Barcelona require to meet the water demands of the city and the metropolitan area. This is managed by the company Aguas de Barcelona (AGBAR) which provided the data and the network topology. The layout of the network is given in the Figure 5. The system is modelled following Section 2.2.1, using a linear time-invariant dynamical system with 63 state variables, 114 manipulated inputs, 88 disturbances and 17 flow intersection nodes. All other parameters and technical characteristics, such as x^{min}, x^{max}, x^s were specified by the network manager (AGBAR). This analysis is carried for a period of 7 days ($H_s = 168$) from 1$^{\text{st}}$ to 8$^{\text{th}}$ July 2007.

The time series models calibrated in Section 2.3 are used in a closed-loop setting with the MPC controller described above. The prediction horizon is $N = 24$ with sampling time of 1 hour. The weighting matrices in (2.16), (2.18) and (2.19) are chosen to be $W_\alpha = 2 \cdot 10^4$, $W_u = 10^5 \cdot I$, and $W_x = I$, respectively.

The optimisation problem (2.25) was solved online using CPLEX. In terms of complexity, it comprises 4248 decision variables, 1512 bound

constraints, 408 linear equations and 4536 linear inequality constraints. On average (based on 500 samples) the computational time for the solution of the optimisation problem was 1.85s (st. dev.: 0.055s) and the time needed for the its formulation was 0.018s (st. dev.: 0.005s).

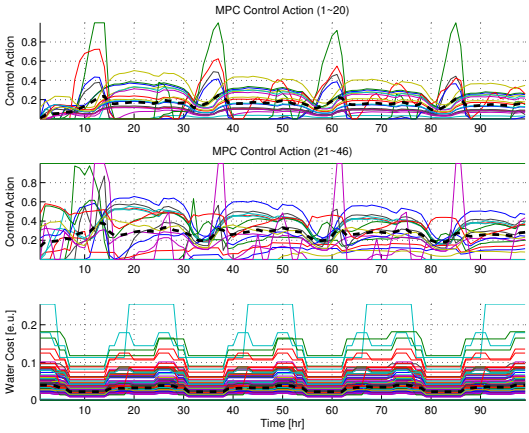

Figure 6: The MPC control action (scaled in the range $[0, 1]$) and the water production cost. The bold dashed lines represent the average values (average control action and average production cost). The input trajectories are split in two graphs for clarity. Only pumping actions are presented here.

As shown in Figure 6, the controller tends to operate the pumps when the electricity cost is low. Moreover, Figure 7 shows that the volume of water in the tanks remains always between the prescribed bounds and tends to stay over the safety storage limit.

In order to demonstrate how the prediction ability of the demand forecasting model effect the economical operations of the DWN, comparison between the case with *actual* demands and with the forecast demand. The SVM model developed is considered for forecast. The operational cost are measure in economical units (e.u) is calculated. It is found that SVM model incur an additional cost of 11.17% than exact future demand scenario. This comparison is shown in Figure 8.

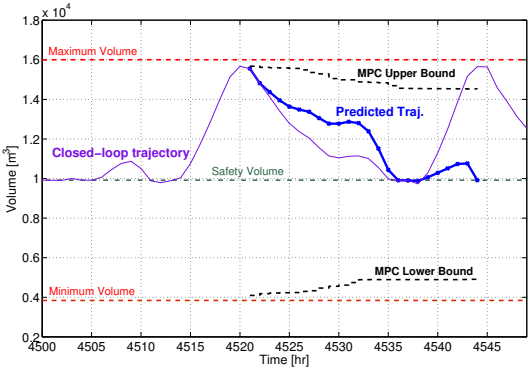

Figure 7: The controlled trajectory of the volume of water in the tank `d130BAR` of the Barcelona DWN.

2.6 Conclusions

Model predictive controller combine with demand forecaster is proposed for the management of drinking water networks. Water demand is identified as the main source of uncertainty and is predicted with three different time-series models – seasonal ARIMA, RBF-SVM and BATS. Finally, the controller is tested on the drinking water network of Barcelona.

In certainty-equivalence MPC, only the nominal forecast of the demand is considered and the constraints are shrunk to avoid closed-loop constraint violation. Here we utilise only information related to the worst-case prediction errors. This is a drawback with this formulation as it risks infeasibility and leads to a conservative control law. These limitations are addressed in a better way with stochastic model predictive control in the next chapter.

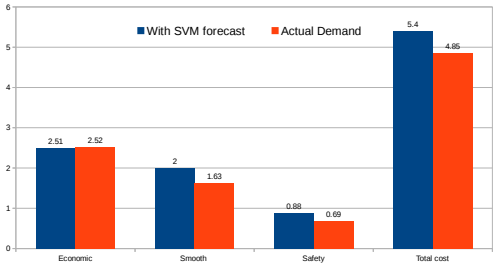

Figure 8: The operational cost for 1 week with perfect prediction and SVM model. This demonstrates that prediction errors lead to a greater cost (expressed in economic units (e.u.)).

Chapter 3

GPU-accelerated methods for stochastic optimal control

Model predictive controller combined with demand forecaster is proposed for real-time management of drinking water networks in Chapter 2. The forecasting errors are taken care by including the worst-case prediction errors and shrinking the constraints. This risks infeasibility and also provides a conservative control law. This encouraged for a stochastic optimal control formulation. In general, stochastic optimal control problems are deemed to be large-scale involving up to millions of decision variables. Their applicability in control applications is often limited by the availability of algorithms that can solve them efficiently and within the sampling time of the controlled system.

This motivated to develop optimisation algorithms that solve stochastic optimal control problems in parallel. Here we propose a dual accelerated proximal gradient algorithm which is amenable to parallelisation and demonstrate that its GPU implementation affords high speed-up values and greatly outperforms well-established commercial optimisers such as Gurobi. These results are published in (SSBP15).

3.1 Introduction

3.1.1 Background

Stochastic optimal control problems typically give rise to large-scale optimisation problems involving up to tens of millions of constrained decision variables. Such problems arise in stochastic model predictive control of Markovian switching systems (PSSB14), stochastic control of networked systems (PSS11b) and of large-scale uncertain systems (SGS$^+$14), portfolio optimisation under uncertainty (RK12), inventory management (Zip00), management of supply chain systems (Son13) and in many other applications of stochastic optimal control.

Despite their popularity, control engineering practice has taken little initiative towards adopting the theoretical results of stochastic optimal control theory, and this is mainly due to the prohibitive computational footprint that accompanies it. The increasing need for computational power gives the floor to graphics processing units (GPUs) and field programmable gate arrays (FPGAs) which are gaining momentum in control applications (LPBC14; MXAM12; RS13). Since one's ability to apply stochastic control methodologies is conditioned by the system's sampling rate, the need for algorithms and hardware that can solve such problems fast is becoming imperative.

Recently, Constantinides authored a tutorial paper in which he outlines the potential advantages of the use of FPGAs and GPUs for (deterministic) model predictive control (MPC) applications (Con09). Rogers and Slegers demonstrated the potential of *in situ* GPU-aided controllers for the guidance of a parafoil where Monte-Carlo simulations are performed in real-time to counteract the wind uncertainty (RS13). Although, there have been efforts to parallelise algorithms for the solution of MPC-related optimisation problems (see (GNDJ14; DBB13; PSA14)), there is no approach that exploits the structure of stochastic optimal control problems to achieve high computational throughput. For example, the approach of Di Cairano *et al.* (DBB13) treats the MPC optimisation problem as a general quadratic programming problem and, as a result, cannot scale up with the problem size. A GPU-based framework to solve large scale two-stage stochastic optimisation problems with applications to uncertain energy dispatch systems is presented in (PSA14).

In this chapter, a scalable parallelisable algorithm is proposed for

solving multi-stage stochastic optimal control problems. It uses accelerated proximal gradient method applied to the Fenchel dual optimisation problem and exploits the problem's structure to further accelerate computations. The dual gradient is calculated as the solution of an unconstrained minimisation problem which is solved by dynamic programming. This problem can be properly decomposed and solved in a parallelised way.

This boils down to an algorithm that requires only matrix-vector products, it is highly parallelisable and can be readily implemented on a GPU or FPGA architecture. The algorithm is well-suited the control for linear dynamical systems with additive and multiplicative uncertain components assuming that these are driven by a stochastic process that can be modelled as an evolution along a scenario tree. The proposed algorithm is division-free and therefore, as O'Donoghue *et al.* accentuate in (OSB13), it is suitable for embedded applications using fixed-point arithmetic. Moreover, it allows the cost function of the optimisation problem to have a nonsmooth part which can be used to encode hard or soft constraints, or terms involving $\|\cdot\|_1$.

3.1.2 Outline

In Section 3.2, we define some mathematical preliminaries that are used in this chapter. Section 3.3 introduces the stochastic optimal control problem and formulates for linear system with additive and multiplicative uncertainty. Later we state the computational limitations with this formulation and present a finite dimension approximation. This formulation is derived using a scenario-tree representation whose outcome is a large-scale optimisation problem. In the Section 3.4, the proximal-gradient algorithm and its accelerated variant are discussed for both primal and the dual problem. In the Section 3.5, the proximal gradient algorithm is applied to the stochastic optimal control problem where the structure of the problem is exploited leading to a parallisable algorithm. In Section 3.6 the performance of this algorithm is demonstrated on the spring-mass example with varying complexity of the scenario tree and increasing prediction horizon. Finally the conclusions are drawn in Section 3.7.

3.1.3 Notation

Let $\mathbb{R}$, $\mathbb{N}$, $\mathbb{R}^n$, $\mathbb{R}^{m\times n}$, $\mathbb{S}^n_+$, $\mathbb{S}^n_{++}$ denote the sets of real numbers, nonnegative integers, column real vectors of length n, real matrices of dimensions m-by-n, symmetric positive semidefinite and positive definite n-by-n matrices respectively. Let $\bar{\mathbb{R}} = \mathbb{R} \cup \{\pm\infty\}$ denote the set of extended-real numbers. The transpose of a matrix A is denoted by A'. The set of of nonnegative numbers $\{k_1, k_1 + 1, \ldots, k_2\}, k_2 \geq k_1$ is denoted by $\mathbb{N}_{[k_1,k_2]}$. We denote by $\|\cdot\|_1$ and $\|\cdot\|_2$ the 1-norm and Euclidean norm on $\mathbb{R}^n$. Unless otherwise stated $\|\cdot\|$ stands for $\|\cdot\|_2$.

3.2 Mathematical preliminaries

In this section we provide the definitions of some mathematical terms which will be used in this chapter.

3.2.1 Probability theory

In this section we provide a quick presentation of some useful definitions and facts from probability theory

Probability Space. A probability space is the triplet $(\Omega, \mathfrak{F}, \mathrm{P})$, where Ω is called the *sample space*, $\mathfrak{F}$ is the *event space* and P is a *probability measure*. In particular,

- *Sample space.* The set of all possible *outcomes* of an experiment is the sample space.
- *Event space.* The event space $\mathfrak{F}$ is a subset of the powerset of Ω — that is, it is a collection of subsets of Ω — which is a σ-algebra, that is
  1. $\varnothing \in \mathfrak{F}$,
  2. $A \in \mathfrak{F} \Rightarrow \Omega \backslash A \in \mathfrak{F}$,
  3. $A_1, A_2, \ldots \in \mathfrak{F} \implies \bigcup_{i=1}^{\infty} A_i \in \mathfrak{F}$.
- *Probability measure.* This is a function $\mathrm{P} : \mathfrak{F} \to [0, 1]$ such that
  1. $\mathrm{P}(\Omega) = 1$,
  2. If $A_1, A_2, \ldots \in \mathfrak{F}$ are disjoint events then $\mathrm{P}\left[\bigcup_{i=1}^{\infty} A_i\right] = \sum_{i=1}^{\infty} \mathrm{P}(A_i)$.

Measurable function. A function $\xi : (\Omega, \mathfrak{F}) \to (\Omega', \mathfrak{F}')$ between two measurable spaces is called measurable if $\xi^{-1}(B) \in \mathfrak{F}$ for every $B \in \mathfrak{F}'$.

Random variable. Let $(\Omega, \mathfrak{F}, \mathrm{P})$ be a probability space. A *measurable function* $\xi : \Omega \to \mathbb{R}$ is called a (real-valued) random variable over Ω — here we assume that $\mathbb{R}$ is equipped with the Borel σ-algebra. [1]

Stochastic process. Let $(\Omega, \mathfrak{F}, \mathrm{P})$ be a probability space. and let T be a parameter set — which is either $T = \mathbb{N}$ or a finite set $T = \{0, 1, \ldots, N\}$. Any collection (sequence) of random variables $\xi = \{\xi_t\}_{t \in T}$ defined on $(\Omega, \mathfrak{F}, \mathrm{P})$ is called a *stochastic process* with the index set T. The index t is often referred to as "time". For each $\omega \in \Omega$, let $\xi(\omega)$ denote the function $t \mapsto \xi_t(\omega)$ from T into $\mathbb{R}$; then $\xi(\omega)$ is a *sequence* — an element of $\mathbb{R}^T$. According to (Ç11, p. 53), the stochastic process ξ can be seen as a random variable $\xi(\omega)$ on the measurable space $(\Omega^T, \mathbb{R}^T)$.

Filtration. For a stochastic process $\xi = \{\xi_k\}_{k \in \mathbb{N}}$, a *filtration* is a construct which describes the information that is available to the random process up to given time k. Formally, for a probability space $(\Omega, \mathfrak{F}, \mathrm{P})$, the sequence of sub-$\sigma$-algebras $\{\mathfrak{F}_k\}_{k \in \mathbb{N}}$ is called a filtration if it satisfies $\mathfrak{F}_s \subset \mathfrak{F}_k$, for all $s < k$. A filtration is a formal way to represent a "stream of information". We can think $\mathfrak{F}_k$ as the information that is observed up to time k. As time progress, we observe the process and more information about it is gathered. This is represented with the nested sequence of σ-algebras of the filtration.

If the probability space $(\Omega, \mathfrak{F}, \mathrm{P})$ is endowed with a filtration $\{\mathfrak{F}_k\}_{k \in \mathbb{N}}$ is called a *filtered probability space* and denoted as $(\Omega, \mathfrak{F}, \{\mathfrak{F}_k\}_{k \in T}, \mathrm{P})$.

A stochastic process $\{\xi_t\}_{t \in T}$ is said to be adapted to a filtration $\{\mathfrak{F}_k\}_{k \in \mathbb{N}}$ if ξ_k is $\mathfrak{F}_k$-measurable for each $k \in \mathbb{N}$. It means the value of the random variable ξ_t can be determined by the information available at time k.

Expectation and Conditional expectation. For a probability space $(\Omega, \mathfrak{F}, \mathrm{P})$ the *expectation* of a random variable $\xi(\omega)$ is defined as

$$\mathbb{E}[\xi] = \int_\Omega \xi(\omega) \mathrm{P}(d\omega), \tag{3.1}$$

where the integral is the *Lebesque integral*. Now let $\mathfrak{G}$ be a sub-σ-algebra of $\mathfrak{F}$. Then a $\mathfrak{G}$-measurable random variable $\bar{\xi}$ is the condi-

[1] The Borel σ-algebra of $\mathbb{R}$ is the standard σ-algebra used with the set of real numbers. It is the smallest σ-algebra to contain the open intervals of $\mathbb{R}$.

tional expectation of C on $\mathfrak{G}$, denoted as $\bar{\xi} = \mathbb{E}[\xi|\mathfrak{G}]$, if it satisfies

$$\int_C \bar{\xi} d\mathrm{P} = \int_C \xi d\mathrm{P} \text{ for all } C \in \mathfrak{G}. \tag{3.2}$$

Note that the conditional expectation of a random variable is *itself a random variable*.

The conditional expectation satisfies $\mathbb{E}\big[\mathbb{E}[\xi|\mathfrak{G}]\big] = \mathbb{E}[\xi]$ whenever ξ is $\mathfrak{G}$-measurable and the well-known and widely-used *tower property*, that is, for two σ-algebras $\mathfrak{G}_1 \subseteq \mathfrak{G}_2$ it is $\mathbb{E}\left[\mathbb{E}\left[\xi \mid \mathfrak{G}_2\right] \mid \mathfrak{G}_1\right] = \mathbb{E}\left[\xi \mid \mathfrak{G}_1\right]$.

3.2.2 Convexity

In this section we provide a quick presentation of some useful definitions and facts from convex theory (BV04; RW09).

A set $\mathcal{C}$ is called convex if

$$x, y \in \mathcal{C}, \lambda[0,1], \text{ implies } \lambda x + (1-\lambda)y \in \mathcal{C}. \tag{3.3}$$

For a function $f : \mathcal{X} \to \bar{\mathbb{R}}$ we define

1. $\operatorname{epi} f = \{(x, \alpha) | x \in \mathcal{X}, \alpha \in \mathbb{R}, f(x) \leq \alpha\}$ which is called the *epigraph* of f, and
2. $\operatorname{dom} f = \{x \in \mathcal{X} | f(x) < \infty\}$ which is called the *domain* of f.

The function $f : \mathcal{X} \to \bar{\mathbb{R}}$ is said to be *convex* when $\operatorname{epi} f$ is a convex set. Another classical — and equivalent — definition of convexity requires that

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y) \tag{3.4}$$

holds for $\lambda \in [0,1]$ and all $x, y \in \operatorname{dom} f$ such that left-hand side is well defined. The function is called *strictly convex* if (3.4) holds strictly ($<$) for every $\lambda \in (0,1)$ and for all pairs $x, y \in \operatorname{dom} f$ with $x \neq y$.

A function $f : \mathcal{X} \to \bar{\mathbb{R}}$ is *σ-strongly convex* if $x \mapsto (x) - \frac{\sigma}{2}\|x\|_2^2$ is a convex function.

Proper function: A function $f : \mathcal{X}^n \to \bar{\mathbb{R}}$ is called *proper* if $f(x) > -\infty$ for some $x \in \operatorname{dom} f$ and its domain is non-empty.

Closed and Lower semicontinuous: A function $f : \mathcal{X} \to \bar{\mathbb{R}}$ is lower semicontinuous at $x \in \mathcal{X}$ if $f(x) \leq \liminf_{k\to\infty} f(x_k)$ for every sequence $\{x_k\} \subset \mathcal{X}$ with $x_k \to x$. A function $f : \mathcal{X} \to \bar{\mathbb{R}}$ is *closed* if

$\operatorname{epi} f$ is closed set. If the function is lower semicontinous on $\mathcal{X}$, then the epigraph of f is closed set in $\mathcal{X} \times \mathbb{R}$.

Properness, convexity, and lower semicontinuity are our standard assumptions for the functions occurring in the optimization problems within this thesis.

β-Lipschitz continuous. A mapping $F : \mathbb{R}^n \to \mathbb{R}^m$ is called β-Lipschitz continuous, with $\beta \geq 0$, if $\|F(x_1) - F(x_2)\|_* \leq \beta \|x_1 - x_2\|$ for every $x_1, x_2 \in \mathbb{R}^n$.

Euclidean projection & Euclidean distance. The *indicator function* of a set $C \subseteq \mathbb{R}^n$ is the extended-real valued function $\delta(\cdot|C) : \mathbb{R}^n \to \bar{\mathbb{R}}$ and for $x \in C$ it is $\delta(x|C) = 0$ and $\delta(x|C) = +\infty$ otherwise. Every non-empty closed convex set $C \subseteq \mathbb{R}^n$ defines the convex function $\operatorname{proj}(x|C) = \operatorname{argmin}_{c \in C} \|x - c\|_2$, which is called the *(Euclidean) projection* of x onto C. The Euclidean distance of a $x \in \mathbb{R}^n$ from C is defined as $d(x|C) = \min_{c \in C} \|x - c\|_2$.

Subgradient & Subdifferential. A vector $g \in \mathbb{R}^n$ is a *subgradient* of f at $x \in \operatorname{dom} f$ when

$$f(y) \geq f(x) + \langle g, y - x \rangle, \text{ for all } y \in \operatorname{dom} f \tag{3.5a}$$

The *subdifferential* of f at $x \in \operatorname{dom} f$ is the set of all the subgradients g of f at $x \in \operatorname{dom} f$ and denoted by $\partial f(x)$

$$\partial f(x) = \{g \mid f(y) \geq f(x) + \langle g, y - x \rangle, y \in \operatorname{dom} f\} \tag{3.5b}$$

In case of a smooth function, the subdifferential is a singleton $\partial f(x) = \{\nabla f(x)\}$.

Conjugate function. Let $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ closed, convex, proper extended real-valued, lower semi-continuous function. The *conjugate* function $f^* : \mathbb{R}^n \to \bar{\mathbb{R}}$ of a function is

$$f^*(y) = \sup_{x \in \mathbb{R}^n} \{\langle y, x \rangle - f(x)\}. \tag{3.6a}$$

This is the point-wise supremum of a family of affine function of y. This function is lower semi-continuous and convex (even if f is not convex).

The function f and its conjugate f^* satisfies the Young-Fenchel inequality

$$f(x) + f^*(y) \geq \langle y, x \rangle, \text{ for all } x, y \in \mathbb{R}^n \tag{3.6b}$$

This inequality follows from the supremum in the definition of the conjugate (3.6a) as

$$f^*(y) \geq \langle y, x\rangle - f(x), \implies f(x) + f^*(y) \geq \langle y, x\rangle, \text{ for all } x, y \in \mathbb{R}^n,$$

We use this relation in the next section to calculate the subgradient of the conjugate function.

Moreau envelope. The *Moreau envelope* or *Moreau-Yosida regularisation* of the function f with parameter γ is defined as

$$f^\gamma(x) = \inf_{v\in\mathbb{R}^n} \left\{ f(v) + \frac{1}{2\gamma}\|x - v\|^2 \right\}, \tag{3.7a}$$

for $\gamma{>}0$. This function can also be seen as the *infimal convolution*[2] of γf and the Euclidean distance function $\|\cdot\|^2$

$$f^\gamma(x) = (\gamma f) \,\square\, \tfrac{1}{2}\|\cdot\|^2 \tag{3.7b}$$

Let us define the function which is minimised inside the Moreau envelope as $L(v, x) := f(v) + \frac{1}{2\gamma}\|x - v\|^2$. This is strongly convex and has a unique minimiser. So $f^\gamma(x) = \inf_v L(v, x)$ is convex according to (RW09, Prop. 2.22) because it is written as an *inf-projection* and it has domain $\mathbb{R}^n$ because of (RW09, Thm. 1.17). The envelope function f^γ is continuously differentiable, even when f is nonsmooth (RW09, Theorem 1.25). The Moreau envelope f^γ is essentially a smooth or regularised version of f. It approximates f from below, that is $f^\gamma \leq f$, and has the same set of minimisers and minimisation of f^γ which is smooth is equivalent to minimising f (RW09). We discuss more details about Moreau envelope and its relationship with proximal algorithms in the Section 3.4.

3.3 Problem statement

This section introduces the general stochastic optimal control problem.

[2]The infimal convolution of two functions f_1 and f_2 is a function $(f_1 \,\square\, f_2)(v) = \inf_{x_1+x_2=v}(f_1(x_1) + f_2(x_2))$ which has two remarkable properties: (i) $\operatorname{dom}(f_1 \,\square\, f_2) = \operatorname{dom} f_1 + \operatorname{dom} f_2$ and (ii) $(f_1 \,\square\, f_2)^* = f_1^*(v) + f_2^*(v)$. The infimal convolution operator is, therefore, dual to addition and it is often termed *epi-addition*. More information can be found in (RW09, Sec. 1-H) and (PB14a, Sec. 3.1).

3.3.1 Stochastic optimal control

Consider the following discrete-time stochastic linear system

$$x_{k+1} = A_{\xi_k} x_k + B_{\xi_k} u_k + w_{\xi_k}, \tag{3.8}$$

where $x_k \in \mathbb{R}^{n_x}$ is the system state, $u_k \in \mathbb{R}^{n_u}$ is the input, and w_{ξ_k} is an additive disturbance term.

Let Ω_k be the sample space of the random variable ξ_k. In what follows we shall consider that each Ω_i, $i \in \mathbb{N}$ is a nonempty finite set. We define the product space $\Omega = \prod_{i\in\mathbb{N}} \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \Omega_i$ and the filtered probability space $(\Omega, \mathfrak{F}, \{\mathfrak{F}_k\}_{k\in\mathbb{N}}, \mathrm{P})$, where $\mathfrak{F}$ is the Borel σ-algebra on Ω and $\{\mathfrak{F}_k\}_{k\in\mathbb{N}}$ is a filtration on Ω where $\mathfrak{F}_k$ is the Borel σ-algebra on $\prod_{i=0}^{k} \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \Omega_i$. Finally, P is the product probability measure on Ω.

In a deterministic setting, the outcome of optimal control problem is a *sequence* of control actions, but that sequence is no longer optimal in presence of uncertainty. This challenge is addressed by minimising over control *laws* rather than sequences of control actions. Let us define $\pi = \{u_k(\cdot)\}_0^{N-1}$ as the *control policy* — the decision variable of the stochastic problem — where $u_k(\cdot)$ are the *control laws*. For a given time instant k the control laws are defined as

$$u_k(\cdot) = \psi(\boldsymbol{x}_k, \boldsymbol{u}_{k-1}, \boldsymbol{\xi}_{k-1})), \tag{3.9}$$

with $\boldsymbol{\xi}_k = (\xi_0, \ldots, \xi_k)$, $\boldsymbol{x}_k = (x_0, \ldots, x_k)$ and $\boldsymbol{u}_k = (u_0, \ldots, u_k)$. This means that decisions at time k are taken using only prior knowledge — in a *causal* fashion.

This is equivalent to saying that $u_k(\cdot)$ is $\mathfrak{F}_k$-measurable. *Markov* policies are a special case of this where the policy depends only on the present state $u_k = \psi(x_k)$ (BS78).

Let us define a stage-cost $\ell : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \Omega_k \to \bar{R}$ that is associated with x_k and u_k and depends on the random variable ξ_k. The stage cost function ℓ is written as

$$\ell(x, u, \xi) = \phi(x, u, \xi) + \bar{\phi}(F_\xi x + G_\xi u, \xi), \tag{3.10a}$$

where ϕ is real-valued, smooth in (x, u), and it is such that the following function of π

$$\mathbb{E}\left[\sum_{k=0}^{N-1} \phi(x_k, u_k, \xi_k)\right] \tag{3.10b}$$

is strongly convex over the manifold that defines the system dynamics [3], while $\bar{\phi}$ is an extended-real valued function, lower semicontinuous, proper, convex and possibly non-smooth.

The stage-by-stage description of the underlying decision-making process can be described as

- starting from $x_0 = p$, choose u_0, apply to the system to get x_1, $\xi_0 \in \Omega_0$ and pay $\ell(x_0, u_0, \xi_0)$,
- choose $u_1 = \psi(x_0, u_0, \xi_0, x_1)$, apply to the system to get x_2, $\xi_1 \in \Omega_1$ and pay $\ell(x_1, u_1, \xi_1)$,

$$\vdots$$

- choose $u_{N-1} = \psi(\boldsymbol{x}_{N-1}, \boldsymbol{u}_{N-2}, \boldsymbol{\xi}_{N-2})$, apply to the system to get x_N and $\xi_{N-1} \in \Omega_{N-1}$ and pay $\ell(x_{N-1}, u_{N-1}, \xi_{N-1})$ and $V_f(x_N, \xi_N)$

Here, $V_f(\cdot, \cdot) : \mathbb{R}^{n_x} \times \Omega_N \to \bar{\mathbb{R}}$ is the terminal cost function. This cost can also be decomposed as

$$V_f(x, \xi) = \phi_N(x, \xi) + \bar{\phi}_N(F_{N,\xi} x, \xi) \tag{3.10c}$$

The *stochastic optimal control* of horizon $N \in \mathbb{N}$ is formulated as:

$$V^\star(p) = \min_{\pi} \mathbb{E}\left[V_f(x_N, \xi_N) + \sum_{k=0}^{N-1} \ell(x_k, u_k, \xi_k) \right], \tag{3.11a}$$

subject to

$$x_{k+1} = A_{\xi_k} x_k + B_{\xi_k} u_k + w_{\xi_k}, \tag{3.11b}$$

$$x_0 = p, \tag{3.11c}$$

where $\mathbb{E}$ is the conditional expectation with respect to the above product measure P. Here the expectation $\mathbb{E}[\cdot]$ operation serves as a quantifier of the cost function which is a random variable.

This problem (3.11) encompasses a very wide class of optimization problems by allowing the stage cost ℓ and the terminal cost V_f to be extended-real valued functions. In what follows the stage cost

[3] This is the case when, for instance, $\phi(x, u, \xi) = x' Q_\xi x + u' R_\xi u$ and Q_ξ are symmetric positive semi-definite and R_ξ are symmetric positive definite.

can be readily taken to be a function of both k and $\boldsymbol{\xi}_k$, *i.e.*, it can be $\ell(x_k, u_k, \boldsymbol{\xi}_k, k)$, but for the sake of simplicity of presentation we will consider the simpler, yet quite general, case of Equation (3.11).

Function $\bar{\phi}$ can be chosen to be any nonsmooth function as dictated by the problem we need to solve. The function $\bar{\phi}$ is used to encode arbitrary hard constraints on states and inputs of the form $F_{\xi_k} x_k + G_{\xi_k} u_k \in Y_{\xi_k}$ by choosing

$$\bar{\phi}(\cdot, \xi_k) = \delta(\cdot | Y_{\xi_k}), \tag{3.12}$$

where Y_{ξ_k} are non-empty convex closed sets for which projections $\mathrm{proj}(\cdot | Y_{\xi_k})$ can be easily computed. This way we may impose polyhedral constraints by choosing Y_{ξ_k} to be boxes (hypercubes) or ellipsoidal constraints by choosing Y_{ξ_k} to be 2-norm-balls.

Soft constraints can be encoded by choosing

$$\bar{\phi}(\cdot, \xi_k) = \eta_{\xi_k} d(\cdot | Y_{\xi_k}), \tag{3.13}$$

where $\eta_{\xi_k} > 0$ is a scaling factor, while one may also choose

$$\bar{\phi}(\cdot, \xi_k) = \| \cdot \|_1 \tag{3.14}$$

to force the optimizer to be sparse.

The smooth part of the stage cost ℓ is here taken to be a quadratic function of the form $\phi(x_k, u_k, \xi_k) = x_k' Q_{\xi_k} x_k + u_k' R_{\xi_k} u_k$, where $R_{\xi_k} \in \mathbb{S}^{n_u}_{++}$ and $Q_{\xi_k} \mathbb{S}^{n_x}_{+}$. The smooth part of the terminal cost function V_f is a quadratic function $\phi_N(x_N, \xi_N) = x_N' P_{\xi_N} x_N$, with $P_{\xi_N} \in \mathbb{S}^{n_x}_{++}$. The function $\bar{\phi}_N$ can be selected in the same way as explained for $\bar{\phi}$, *e.g.*, terminal constraints of the form $F_{N,\xi} x_N \in X_f$ can be encoded using $\bar{\phi}_N(\cdot, \xi) = \delta(\cdot | X_f)$, where X_f is assumed to be such that $\mathrm{proj}(\cdot | X_f)$ can be easily evaluated computationally.

Challenges in stochastic optimal control

The main characteristics of the above stochastic optimal control problem (3.11) are (i) the decision variables are causal policies $u_k(\cdot)$ instead of single variables and (ii) the objective function is an expectational cost involving multi-dimensional integrals. Both these features make this an infinite dimensional optimisation problem. One has to consider either approximations of the original problem or draw restrictive

assumptions on the driving random process such as that ξ_{k_1} and ξ_{k_2} are independently distributed for $k_1 \neq k_2$ and that ξ_k are normally distributed.

A popular approximation is assuming the uncertainty is Gaussian with known mean and variance and restricting the set of admissible policies to affine feedback policies. Some popular works based on these assumptions are the stochastic tube formulation with predefined feedback policy (CKW09; KCRC10) and affine disturbance feedback policies (OJM08; CL15). Such assumptions make sense in some cases but fails to be realistic more often than not.

3.3.2 Finite dimension formulation

An alternative approach to solve the infinite dimensional is to approximate it with a finite number of *scenarios* bundled in the form of a *scenario tree* which is a discrete approximation of the probability distribution of the underlying random process. In this section we describe the stochastic optimal control formulation with the scenario-trees.

Scenario trees

A *scenario* is an outcome of the random process $\boldsymbol{\xi}_N$. A scenario tree is a structure dictated by the filtration of the probability space and describes the evolution of the process. This is illustrated in Figure 9. In this section, we discuss the structure and primitives through which we can describe the evolution of the scenario tree (SDR09).

The scenario tree is a finite set of nodes interconnected as shown in Figure 9. The node at stage $k = 0$ is called the *root* node. At the root node, there is no information about the stochastic process but the observation of x_0. At this stage, there is no uncertainty about the state of the system. The nodes at the next stage $k = 1$ correspond to the possible outcomes of the random variable ξ_0. In the tree shown in the Figure 9 there are three possible outcomes of ξ_1. Similarly the nodes at the last stage $k = N$ are the possible outcome of the random variable ξ_{N-1} — these are the *leaf nodes* of the tree.

Every node at stage $k = 1$ is a *child* of root node and the root node is the ancestor node. Similarly each node at stage $k = 2$ is a child of a node at stage $k = 1$ and nodes of stage $k = 1$ their ancestor. The leaf nodes of the tree have no children. A particular realisation of ξ_k is

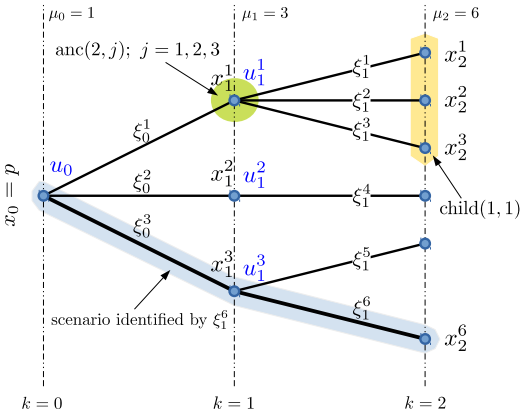

Figure 9: Example of a scenario tree structure. This figure represents the evolution of the system dynamics using a scenario tree structure.

denoted by ξ_k^i and it is associated with the i-th node at k-th stage. We will call the maximum number of children at a stage k the *branching factor* b_k of this stage.

The sample space Ω_k of stage k is the set of all the nodes at stage $k \in \mathbb{N}_{[1,N]}$. The cardinality of the set Ω_k is denoted by $\mu(k)$, while the total number of nodes of the tree is μ. Non-leaf nodes have a set of *children*; for $i \in \mathbb{N}_{[1,\mu(k)]}$ we denote the children nodes of the i-th node at stage k by $\text{child}(k, i) \subseteq \mathbb{N}_{[1,\mu(k+1)]}$. The ancestor of the node i at stage $k \in \mathbb{N}_{[1,N]}$ is denoted by $\text{anc}(k, i)$ and it gives a node at the $k-1$ stage. Finally, as it follows from the above discussion, a node of the tree is uniquely identified by the pair (k, i) with $k \in \mathbb{N}_{[0,N]}$ and $i \in \mathbb{N}_{[1,\mu(k)]}$.

A *scenario* is an admissible path from the root node to a leave nodes. Every node in the tree has a unique path from the root node which represents the history of the process. Let us denote $s(k, i)$ as the path to node i at stage k from the root node which can be recursively written as

$$s(k, i) = \{(k, i), (k-1, s(k-1, \text{anc}(k-1, i)))\}, \tag{3.15}$$

with $s(0,1) = \{(0,1)\}$.

The *number of scenarios* in the tree are equal to the number of leaf nodes $\mu := \mu(k)$. Let us denote with $\mathrm{scen}(k,i)$ as the set of scenarios that run through node i at stage k. This set is given by

$$\mathrm{scen}(k,i) = \{j \mid i \in s(N,j)\}. \tag{3.16}$$

The sets of children are disjoint, i.e., $\mathrm{child}(k,i) \cap \mathrm{child}(k,j) = \varnothing$, for $i,j \in \Omega_k, i \neq j, k \in \mathbb{N}_{[1,N-1]}$ and $\Omega_{k+1} = \cup_{i\in\Omega_k} \mathrm{child}(k,i)$. Shapiro *et al.* use this very structure to describe the scenario tree structure over the sample space Ω_N with σ-algebra $\mathfrak{F}_N = 2^{\Omega_N}$ and proceed to identifying Ω_k by a filtered space $(\Omega, \mathfrak{F}, \{\mathfrak{F}_k\}_{k=0}^N)$ where $\{\mathfrak{F}_k\}_{k=0}^N$ is induced by a successive regrouping of the leaf nodes according to their ancestors (SDR09; SB16). We shall discuss this further at the end of this section.

Different nodes of certain realisations of ξ_k can have the same numerical value but they may differ in their histories. Each node in the tree is associated with a probability mass which indicates the probability of its occurrence. The root node has probability 1 and the probability of moving from node i at stage k to a node $l \in \mathrm{child}(k,i)$ is p_k^{li}. Naturally $\sum_{l\in\mathrm{child}} p_{k+1}^{il} = 1$. These probabilities represent the conditional distribution of ξ_{k+1} given the path of the process $s(k,i)$. These probabilities are in one-to-one correspondence with the boughs of the tree.

Let us denote the probability of reaching a node i at stage k from root as p_k^i. This probability is the product of the conditional probabilities of the nodes encounter in its path from the root node, i.e.,

$$p_k^i = p_k^{i\,\mathrm{anc}(k,i)} p_{k-1}^{\mathrm{anc}(k,i)}$$

The summation of these probabilities at every stage gives

$$\sum_{i=1}^{\mu(k)} p_k^i = 1,$$

and for $k \in \mathbb{N}_{[0,N-1]}$ and $i \in \mathbb{N}_{[1,\mu(k)]}$ we have

$$\sum_{j\in\mathrm{child}(k,i)} p_{k+1}^j = p_k^i.$$

The tree structure can also be related with the filtration associated with the data process. Let us define $\mathfrak{F}_N$ as the sigma algebra associated with the set Ω_N. Now the set Ω_N can be represented as a union of disjoint sets $\text{child}(N-1,i)$, $i \in \Omega_{N-1}$ and the sigma algebra $\mathfrak{F}_N$ is the collection of these subsets. Let $\mathfrak{F}_{N-1}$ be the sub-algebra of $\mathfrak{F}_N$ generated by the sets $\text{child}(N-1,i)$, $i \in \Omega_{N-1}$. As these sets are disjoint, the resulting sets become the elementary events of $\mathfrak{F}_{N-1}$. By continuing in this way, a sequence of sigma algebra $\mathfrak{F}_0 \subset \mathfrak{F}_1 \ldots \subset \mathfrak{F}_N$ is constructed with $\mathfrak{F}_0 = \{\varnothing, \Omega_N\}$. This sequence of nested sigma algebras is called *filtration* and the corresponding scenario tree is a representation of this filtration. The elementary events of $\mathfrak{F}_k$ is a subset of Ω_N for which is corresponded to a node in Ω_k.

3.3.3 Stochastic optimal control based on scenario trees

The scenario tree approximate the stochastic process to finite dimension process. This section formulates the stochastic optimal control problem using the scenario tree. In practice, such scenario trees can be generated, *inter alia*, from real data by the algorithm proposed by Heitsch and Römisch (HR03).

Let us denote by x_k^i, u_k^i as the state and control action at the node i of stage k and let $\pi = \{\{x_k^i\}_{k,i}, \{u_k^i\}_{k,i}\}$. Let us also denote the states and control actions at k as $\mathbf{x}_k = \{x_k^i\}$,$\mathbf{u}_k = \{u_k^i\}$ respectively. This is shown in Figure 9.

At $k = 0$, the state $x_0^1 = p$ is known and the states at the next stages are predicted based on the scenario tree structure as:

$$x_{k+1}^j = A_k^j x_k^i + B_k^j u_k^i + w_{k+1}^j, j \in \text{child}(k,i). \tag{3.17}$$

At stage $k = 0$, the only information available is $x_0^1 = p$ and the action u_0 is decided. At stage $k = 1$, the state of the system is either of x_1^i with $i \in \Omega_1$ and the action u_1^i, $i \in \Omega_1$ is to be chosen. This action is based only on the past information x_0^1, u_0^1, w_1^i. Similarly, the action u_k^i at a state x_k^i, $i \in \Omega_k$ is chosen based on the history $s(k,i)$.

The expectation operation on the scenario tree is equivalent to summation across all the nodes of the tree weighted with the respective probability values. Here the decision variables are $\pi = \{x_k^i, u_k^i\}_{k,i}$.

Now the stochastic optimal control (3.11) can be rewritten as:

$$V^\star(p) = \min_{\pi} \sum_{k=0}^{N-1} \sum_{i=1}^{\mu(k)} p_k^i \ell(x_k^i, u_k^i, i) + \sum_{i=1}^{\mu(N)} p_N^i V_f^i(x_N^i, i) \tag{3.18a}$$

subject to

$$x_{k+1}^j = A_k^j x_k^i + B_k^j u_k^i + w_{k+1}^j, j \in \text{child}(k,i), \tag{3.18b}$$

$$x_0^1 = p. \tag{3.18c}$$

The number of decision variables depend on the number of scenarios which is given by

$$\sum_{k=0}^{N-1} \mu(k)(n_x + n_u) + \mu(N) n_x \tag{3.19}$$

So the number of decision variables depends on the nodes of the scenario tree. With a scenario tree, the stochastic optimal control becomes a large-scale finite-dimensional optimisation problem. Although the number of decision variables may be at the order of millions, the problem possesses a certain structure which can be exploited in the context of numerical optimisation methods to solve it very efficiently.

3.4 Proximal methods

Proximal algorithms are a class of numerical optimisation methods for solving nonsmooth, convex optimisation problems where (i) we identify a certain structure of the problem — a splitting of the cost function in two parts and (ii) we make use of the *proximal operator* — an operator of key importance in modern optimisation theory — to formulate such efficient solution algorithms.

3.4.1 Proximal operator & properties

In this section we introduce the proximal operator and we present some of its properties. A comprehensive overview of the main properties of the proximal operator are given in (PB14a).

Let us first define the *proximal* operator $\text{prox}_{\gamma f} : \mathbb{R}^n \to \mathbb{R}^n$ of f as follows

$$\text{prox}_{\gamma f}(v) = \underset{x \in \mathbb{R}^n}{\text{argmin}} \left\{ f(x) + \tfrac{1}{2\gamma} \|x - v\|_2^2 \right\}, \tag{3.20}$$

for $\gamma > 0$. The function inside the minimiser is strongly convex and not everywhere infinite. So this operation always results in a unique solution. The proximal operator generalises the Euclidean projection operation. When $f(x)$ is the indicator function $f(x) = \delta_{\mathcal{C}}(x)$, then the proximal operator is the projection operator

$$\text{prox}_f(v) = \text{proj}(x|\mathcal{C}) = \underset{x \in \mathbb{R}^n}{\text{argmin}} \|x - v\|_2^2.$$

The proximal operator gives the solution of minimisation that defines the Moreau envelope (3.7a). So the proximal operator satisfies

$$f^\gamma(x) = f(\text{prox}_{\gamma f}(x)) + \frac{1}{2\gamma} \| \text{prox}_{\gamma f}(x) - x\|^2 \tag{3.21a}$$

The gradient of the Moreau envelope , as shown in (RW09, Thm. 2.26), is given by

$$\text{prox}_{\gamma f}(x) = x - \gamma \nabla f^\gamma(x) \tag{3.21b}$$

At a minimiser of f, $x^\star \in \text{argmin}_x f(x)$, the gradient of the Moreau envelope vanishes, i.e., $\nabla f^\gamma(x^*) = 0$. So substituting this in the equation (3.21b), we have

$$x^\star = \text{prox}_f(x^\star) \tag{3.21c}$$

Equation (3.21c) can be seen as a simple optimality condition for the problem of minimising f. A simple fixed-point iteration on (3.21c) reads

$$x^{k+1} = \text{prox}_f(x^k), \tag{3.21d}$$

with x^0 being an initial guess. This is the well-known *proximal-point algorithm* applied for the minimisation of f.

The proximal-point algorithm can be viewed as a gradient step for the regularised function f^γ with a step-size γ.

This relationship allows to interpret the minimisation of f as equivalent to finding the fixed point for the proximal operator of f with parameter γ. For further details on fixed point theory and relation with proximal operator check (Roc76; BC11).

A property of major importance for the formulation of efficient algorithms is the *separable sum property*. The proximal operator of a separable sum reduces to evaluating the proximal operators of each of the summands, which can be done independently and concurrently. If $f(x) = \sum_{i=1}^{n} f_i(x_i)$, then

$$(\mathrm{prox}_f(v))_i = \mathrm{prox}_{f_i}(v_i). \tag{3.22}$$

The *Moreau decomposition* is a property that correlates the proximal operator of f and that of its conjugate f^*. In particular, it allows us to express proximal operators of a conjugate function in terms of the proximal operator of the original function. This can be stated as:

$$x = \mathrm{prox}_{\gamma f}(x) + \gamma\, \mathrm{prox}_{\gamma^{-1} f^*}(\gamma^{-1} x) \tag{3.23}$$

Substituting the Moreau envelope gradient equation (3.21b) in (3.23), we have

$$\mathrm{prox}_{\gamma^{-1} f^*}(\gamma^{-1} x) = \nabla f^{\gamma}(x). \tag{3.24}$$

As we will discover hereafter, this is a very useful property which enables the use of proximal algorithms for the solution of Fenchel-dual optimisation problems.

Computing the proximal operator

In proximal algorithms, the basic operation is the proximal operator, prox_f. Unlike the gradient, evaluating the proximal operator requires to solving a small optimisation problem. When f is smooth, then this function can be evaluated using the standard iterative methods based on gradient or Hessian methods. But for most standard functions, the proximal operator admits a closed-form or can be evaluated using the properties of proximal operator. For example for indicator function, $f(x) = \delta(x|C)$, it is $\mathrm{prox}_{\lambda g}(v) = \mathrm{proj}(v|C)$. The proximal operator of $d(\cdot|C)$ can also be easily computed provided that C is simple enough so that both $d(\cdot|C)$ and $\mathrm{proj}(\cdot|C)$ can be computed easily.

In Section 4.4.6, the proximal operator of the functions that are relevant for this thesis are evaluated and for complete list of proximal operators check Combettes *et al.* (Com11).

3.4.2 Proximal algorithms

This section introduces the algorithms with proximal operators of the objective terms. These algorithms can easily be generalised to non-smooth, extended real-valued functions. Like gradient methods, these methods are first-order and suitable for large-scale problems.

Proximal point algorithm

The simplest algorithm using proximal operator is *proximal point algorithm*

$$x^{\nu+1} = \text{prox}_{\lambda f}(x^{\nu}), \tag{3.25}$$

where x^{ν} is the ν-th iterate and f is a closed proper convex extended real-valued function. This is similar to Tikhonov regularisation where an quadratic penalty term is added to improve the condition number of the function and encourage next iterate $x^{\nu+1}$ to be close to the x^{ν}. As x^{ν} approaches the minimiser, the quadratic term becomes smaller and smaller.

Another perspective is that the proximal point is the gradient method for the Moreau envelope, that is

$$\begin{aligned} x^{\nu+1} &= \text{prox}_{\lambda f}(x^{\nu}) \\ &= x^{\nu} - \nabla f^{\gamma}(x^{\nu}) \end{aligned}$$

The convergence of this algorithm is $\mathcal{O}(1/\nu)$ given by Güler (Gül91) and the relationship with monotone operator is studied in Rockafellar (Roc76). In some cases, the proximal operation may not be available in a closed form; then we have to resort to an inexact evaluation of the proximal operator (HY12; Han03).

Proximal gradient algorithm

The objective function in most applications is not a single non-separable function but rather a composite function which can be written as a

summation of simpler functions. In the proximal point algorithm context we need to be able to evaluate the proximal operator of the whole function which is usually rather cumbersome to evaluate. Instead, we may be able to identify two summands in the cost function: one that is *continuously differentiable* and one that is *prox-friendly*, i.e., has a proximal operator that is easy to evaluate. In such cases, it is recommended to devise an algorithm that takes advantage of this very structure. Such a separation is not unique and different splittings would result in algorithms with different computational complexity and convergence rate.

Consider a composite function $F(x) = f(x) + g(x)$, where $f(x)$ is convex and differentiable with domain $\mathbb{R}^n$, $g(x)$ is convex but not necessarily differentiable. Assume that proximal operator of g is inexpensive. The optimisation problem is

$$\min_x F(x) = f(x) + g(x) \tag{3.26}$$

Consider a given iterate $x^{\nu-1}$. We replace the f-part of F with a local quadratic approximation about $x^{\nu-1}$, i.e.,

$$\begin{aligned}\hat{F}_\lambda(x, x^{\nu-1}) = f(x^{\nu-1}) + \langle x - x^{\nu-1}, \nabla f(x^{\nu-1})\rangle \\ + \tfrac{1}{2\lambda}\|x - x^{\nu-1}\|^2 + g(x),\end{aligned} \tag{3.27}$$

where $\lambda > 0$. This is a convex upper bound of $F(x)$ with $\hat{F}_\lambda(x, x) = F(x)$, i.e, $F(x) \leq \hat{F}_\lambda(x, x^{\nu-1})$ for all x. This quadratic approximation has the same minimisers as the original function, i.e.,

$$\operatorname*{argmin}_x F(x) = \operatorname*{argmin}_x \hat{F}_\lambda(x, x^{\nu-1}). \tag{3.28a}$$

Rearranging the terms in $\hat{F}_\lambda(x, x^{\nu-1})$, the minimisation is written as

$$\operatorname*{argmin}_x F(x) = \operatorname*{argmin}_x \left\{g(x) + \tfrac{1}{2\lambda}\|x - (x^{\nu-1} - \lambda\nabla f(x^{\nu-1}))\|^2\right\} \tag{3.28b}$$

This expansion is essentially the application of the proximal operator prox_g on the gradient update with respect to f. It can be written concisely as

$$x^{\nu} = \text{prox}_{\lambda g}(x^{\nu-1} - \lambda \nabla f(x^{\nu-1})) \tag{3.29}$$

Another approach to solving the above composite minimisation problem is to use the following optimality condition

$$x = \text{prox}_{\lambda g}(x - \lambda \nabla f(x)), \tag{3.30a}$$

which is equivalent to

$$0 \in \nabla f(x) + \partial g(x). \tag{3.30b}$$

A fixed-point iteration on (3.30a) is exactly the *proximal gradient algorithm* in (3.29) which consists in a *forward step* involving the gradient of f, that is $x - \lambda \nabla f(x)$, and a *backward step* which is the application of $\text{prox}_{\lambda g}$ to the forward update.

When $g = 0$, the proximal gradient algorithm (3.29) reduces to the gradient method and when $f = 0$, it reduces to the proximal point algorithm. When $g = \delta_{\mathcal{C}}(x)$, then the proximal gradient algorithm is the gradient projection algorithm.

When the gradient ∇f is Lipschitz continuous with constant $L > 0$, this method converges with rate $\mathcal{O}(1/\nu)$ (BT09a; Zhu95) with constant step size $\lambda = 1/L$. When the Lipschitz constant is not known or easily computable, then the step size can be calculated with an easy backtracking rule

$$F(x^{\nu}) \leq \hat{F}_{\lambda_\nu}(x^{\nu}, x^{\nu-1}). \tag{3.31}$$

As we will explain later in Section 3.4.3, a simple line search algorithm can be introduced to find a Lipschitz constant estimate L_ν — at iteration ν — so that $\lambda_\nu = 1/L_\nu$ satisfies condition (3.31).

Accelerated proximal gradient (APG) algorithm

The idea behind the accelerated version is to improve the convergence rate by using the previous iterate. Nesterov proposed accelerated version for the gradient-projection algorithm that can achieve a convergence rate of $\mathcal{O}(1/\nu^2)$ (Nes83). This method is extended to proximal-gradient framework by Beck *et al.* (BT09a). Here the proximal-gradient update is calculated at an extrapolated vector given by

$$v^{\nu} = x^{\nu} + \theta_{\nu}(\theta_{\nu-1}^{-1} - 1)(x^{\nu} - x^{\nu-1}).$$

The choice of θ_ν should satisfy the condition

$$\frac{1-\theta_\nu}{\theta_\nu^2} \leq \frac{1}{\theta_{\nu-1}^2}, \text{ for } \nu \geq 2.$$

This condition facilitates the decrease of θ_k and in ideal case, this decrease should be as fast as possible. An attractive choice is

$$\theta_k = \frac{2}{k+2}, \tag{3.32a}$$

or solve the quadratic equation with "=" to get

$$\theta_{k+1} = {}^1\!/_2(\sqrt{\theta_\nu^4 + 4\theta_\nu^2} - \theta_\nu^2), \tag{3.32b}$$

which tends to zero faster. For fast convergence, the $\{\theta_\nu\}$ should decrease as fast as possible and the later choice is preferred (Tse08). The accelerated gradient step can be summarised as:

$$v^{\nu} = x^{\nu} + \theta_{\nu}(\theta_{\nu-1}^{-1} - 1)(x^{\nu} - x^{\nu-1}), \tag{3.33a}$$

$$x^{\nu+1} = \text{prox}_{\lambda g}(x^{\nu} - \lambda_\nu \nabla f(x^{\nu})), \tag{3.33b}$$

$$\theta_{\nu+1} = {}^1\!/_2(\sqrt{\theta_\nu^4 + 4\theta_\nu^2} - \theta_\nu^2). \tag{3.33c}$$

When the step-size λ is not available, then the step size that satisfies sufficient decrease condition given by (3.31) is calculated by backtracking. This condition is:

$$F(x_{\nu+1}) \leq \hat{F}_{\lambda_\nu}(x_{\nu+1}, v^{\nu}).$$

An optimisation algorithm is defined as monotonic if their iterates satisfy the decrease condition given as

$$F(x^{\nu}) \leq F(x^{\nu-1}), \tag{3.34}$$

where x^{ν} and $x^{\nu-1}$ are the iterates. The value function generated by the proximal-gradient method satisfy this monotonic condition. But the accelerated variant does not and can be enforced on it by discarding the extrapolation step when the value is not monotonic. This enforcement of monotonicity does not effect the theoretical convergence of the algorithm (BT09b).

3.4.3 Dual proximal gradient

Consider the optimisation

$$P^\star = \min_x f(x) + g(Hx), \tag{3.35}$$

where $x \in \mathbb{R}^n$, $H : \mathbb{R}^n \to \mathbb{R}^m$ is a linear operator, $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ is strongly convex function and $g : \mathbb{R}^m \to \bar{\mathbb{R}}$ is closed, proper and convex function. This formulation arises in many applications. For example, in constrained quadratic programming we may choose f to be the quadratic cost function and g to be the indicator of a hypercube while H is used to impose general polytopic constraints of the form $K_{\min} \leq Hx \leq K_{\max}$. In ℓ_1-regularised least squares, we may choose f to be the the least-squares part and g to be the regulariser (SFP16). Similarly, the formulation in (3.35) can be used — and has been used — as a mould for several convex optimisation problems.

Even though the proximal of g, prox_g is inexpensive, the proximal operator of $g(Hx)$, $\operatorname{prox}_{H \circ g}$ is typically not available in closed form and its computation requires the solution of an optimisation problem (using a numerical method). *Splitting* is a tool that allows to decompose the original variable to multiple variables. With this tool the above problem can be written as

$$\begin{aligned} P^\star = &\min_{x,z} f(x) + g(z), \\ &\text{subject to } Hx = z. \end{aligned} \tag{3.36}$$

Now the decision variables are x, z and are coupled by the constraint $Hx = z$. The dual problem associated with this is

$$D^\star = \min_y f^*(-H'y) + g^*(y), \tag{3.37}$$

where f^*, g^* are conjugates of f and g respectively. The Fenchel duality theorem (Roc72, Corol. 31.2.1) states that if there exists $x \in \operatorname{ri}(\operatorname{dom} f)$[4], $z \in \operatorname{ri}(\operatorname{dom} g)$ such that $Hx = z$, then strong duality holds, that is

[4] The *relative interior* of a set $A \subseteq \mathbb{R}^n$ is the interior of this set with respect to the relative topology in $\mathbb{R}^n$ induced by the affine hull of A. The affine hull of A is the smallest affine set which contains A.

$P^\star = D^\star$[5]. Strong duality is implied from the properties of the domains of f and g and it has nothing to do with the nature of f or g[6]. An optimal dual solution $y^\star$ (could be many) would give the primal optimal solution $(x^\star, z^\star)$.

$$x^\star = \operatorname*{argmin}_{x\in\mathbb{R}^n} \left\{ f(x) + \langle y^\star, Hx\rangle \right\}, \tag{3.38}$$

$$z^\star = \operatorname*{argmin}_{z\in\mathbb{R}^m} \left\{ \langle y^\star, z\rangle - g(z) \right\}, \tag{3.39}$$

with $Hx^\star = z^\star$.

Function f is σ-strongly convex, so f^* is smooth with $1/\sigma$-Lipschitz-continuous gradient (RW09, Prop. 12.60). In the dual formulation (3.37), the linear operator is right-composed with the smooth function f and the proximal-gradient algorithm is perfectly suitable for this formulation. Recall that using the Moreau decomposition property (3.23) the computation of $\operatorname{prox}_{\gamma g^*}$ boils down to being able to compute $\operatorname{prox}_{\gamma^{-1} g}$.

The Fenchel-Young inequality (3.6b) gives rise to a relation which connects conjugates and subgradients

$$y \in \partial f(x) \Leftrightarrow f(x) + f^*(y) = y'x \Leftrightarrow x \in \partial f^*(y). \tag{3.40}$$

This is also popular as the *conjugate-subgradient theorem* (Roc72, Thm. 23.5). In the present case the conjugate f^* is smooth and therefore the subdiffrential set contain only the gradient, $x = \nabla f^*(-H'y)$. From the definition of conjugate function (3.6a), we have

$$x := \nabla f^*(-H'y) = \operatorname*{argmin}_x \{\langle x, H'y\rangle + f(x)\}. \tag{3.41}$$

Now the proximal gradient algorithm applied to the dual problem is defined as the recursion on dual variables y^ν

$$y^{\nu+1} = \operatorname{prox}_{\lambda g^*}(y^\nu + \lambda H x^\nu), \tag{3.42}$$

starting from a dual-feasible vector y^0, *e.g.*, $y^0 = 0$. As mentioned previously, the proximal with g is inexpensive and therefore the proximal

[5] Such conditions (with which we may prove strong duality, $P^\star = D^\star$) are known as *constraint qualifications*.

[6] We hereafter assume that strong duality holds; this holds if, for instance, the primal problem can be written as a convex quadratic problem.

of its conjugate is given by the Moreau decomposition property (3.23). The proximal in (3.42) is determined as

$$z^{\nu} = \text{prox}_{\lambda^{-1}g}(\lambda^{-1}y^{\nu} + Hx^{\nu}), \tag{3.43}$$

$$y^{\nu+1} = y^{\nu} + \lambda(Hx^{\nu} - z^{\nu}). \tag{3.44}$$

The proximal gradient method, given by (3.41)–(3.44), produces a sequence y^{ν} which, for properly small λ, converges to a dual optimal solution $y^{\star}$, while the corresponding primal sequences x^{ν} and z^{ν} converge to the (unique) primal optimal solution $(x^{\star}, z^{\star})$.

Accelerated Proximal Gradient (APG) Algorithm: With the acceleration scheme mentioned in the Section 3.4.2 the dual gradient algorithm achieves a convergence rate of $\mathcal{O}(1/\nu^2)$ for the dual problem (Nes83). The accelerated algorithm is summarized as follows:

$$v^{\nu} = y^{\nu} + \theta_{\nu}(\theta_{\nu-1}^{-1} - 1)(y^{\nu} - y^{\nu-1}), \tag{3.45a}$$

$$x^{\nu} = \text{argmin}_z\{\langle z, H'v^{\nu}\rangle + f(z)\}, \tag{3.45b}$$

$$z^{\nu} = \text{prox}_{\lambda^{-1}g}(\lambda^{-1}v^{\nu} + Hx^{\nu}), \tag{3.45c}$$

$$y^{\nu+1} = v^{\nu} + \lambda(Hx^{\nu} - z^{\nu}), \tag{3.45d}$$

$$\theta_{\nu+1} = 1/2(\sqrt{\theta_{\nu}^4 + 4\theta_{\nu}^2} - \theta_{\nu}^2), \tag{3.45e}$$

starting with $y^0 = y^{-1}$, $\theta_0 = \theta_{-1} = 1$.

The primal iterate x^{ν} given in (3.45b) is not guaranteed to be feasible since Hx^{ν} might not belong to $\text{dom}\, g$. This is a possible weak point of algorithms on dual problems. The strong convexity of f, however, ensures that the Euclidean distance of the ergodic primal iterate $\bar{x}^{\nu}$, $\bar{x}^{\nu} = (1 - \theta_{\nu-1})\bar{x}^{\nu-1} + \theta_{\nu}x^{\nu}$, from the optimal $x^{\star}$ converge to zero with a convergence rate of $\mathcal{O}(1/\nu^2)$ (PB14b).

Choice of λ: The APG converges with a step-size $\lambda \in (0, 1/L]$ with optimal step-size $\lambda = 1/L$ where L is the Lipschitz constant of the dual gradient. In the dual formulation, the smoothness of the conjugate function f^* is implied from the strong convexity assumption of f. When f is σ-strong convexity, then the Lipschitz constant of the dual gradient ∇f^* can be computed as $1/\sigma$. Now the Lipschitz of $\nabla\hat{f}(y)$ where $\hat{f}(y) = f^*(-H'y)$ is

$$\begin{aligned}\|\nabla\hat{f}(y_1) - \nabla\hat{f}(y_2)\| &= \|H\|\|\nabla f^*(-H'y_1) - \nabla f^*(-H'y_2)\|, \\ &\leq \frac{\|H\|^2}{\sigma}\|y_1 - y_2\|. \end{aligned} \tag{3.46}$$

Even though the (best/smallest) Lipschitz constant is available in closed-form, in many cases it is difficult to calculate it either because of the large dimension of H or the fact that σ is not known. In such cases, an additional backtracking line search based on the quadratic upper bound in (3.31)[7] is included in the algorithm.

Algorithm 3 Line search for λ

Require: v^ν, $\lambda_{\nu-1}$ and $\beta \in (0,1)$ — e.g., $\beta = 0.5$.
  $\lambda \leftarrow \lambda_{\nu-1}$
  repeat
    $x \leftarrow \operatorname{argmin}_z \{\langle z, H'v^\nu \rangle + f(z)\}$
    $z \leftarrow \operatorname{prox}_{\lambda^{-1} g}(\lambda^{-1} v^\nu + Hx)$
    $y \leftarrow v^\nu + \lambda(Hx - z)$
    $\lambda \leftarrow \beta\lambda$
  until $f^*(-H'y) - f^*(-H'v^\nu) \leq \langle Hx, v^\nu - y \rangle + \frac{1}{2\lambda}\|v^\nu - y\|^2$
  return λ

Termination criteria

In dual methods, the goal is to find a dual optimal such that the dual cost satisfies a dual suboptimality condition of the form $D^\star - D(y^\nu) \leq \epsilon_d$ where $D(y) = f^*(-H'y) + g^*(y)$. Instead, we are interested in determining solutions with bounded dual and primal suboptimality and, in case the problem involves constraints, bounded infeasibility. For $\epsilon_V > 0$ and $\epsilon_g > 0$ we say that a primal solution (x, z) is (ϵ_V, ϵ_g)-suboptimal if the following conditions are satisfied

$$P(x,z) - P^\star \leq \epsilon_V, \tag{3.47a}$$
$$\|x - Hz\| \leq \epsilon_g, \tag{3.47b}$$

[7] In Section 3.4.2 we stated the backtracking condition for determining the step size online for the primal optimisation problem. Here, in Algorithm 3 we give the necessary condition for the *dual* proximal gradient algorithm. When L is not known and we cannot select the optimal value of a fixed step length λ, we start with a large and arbitrary λ_0 which, at every iteration we decrease using Algorithm 3. Further details on this line search algorithm can be found in (SGB14; G̈92; Nes12).

where $P(x,z) = f(x) + g(z)$ the first condition is the *primal suboptimality condition* and the second one will be referred to as the *primal residual*.

The *duality gap* at iteration ν which is quantified by $P(x^\nu, z^\nu) - D(y^{\nu+1})$ is bounded by ϵ_V

$$P(x^\nu, z^\nu) - D(y^{\nu+1}) \leq \epsilon_V. \tag{3.48}$$

Here $D(y^{\nu+1}) \leq P^\star$ and this implies $P(x^\nu, z^\nu) - P^\star \leq \epsilon_V$.

In the dual proximal gradient algorithm, the primal residual and the primal suboptimality associated with the ergodic primal iterate, $\bar{x}^\nu, \bar{z}^\nu$ [8] converges $\mathcal{O}(1/\nu^2)$ (PB14b). One can test the condition $P(\bar{x}^\nu, \bar{z}^\nu) - D(y^{\nu+1}) \leq \epsilon_V$ and $\|H\bar{x}^\nu - \bar{z}\| \leq \epsilon_g$ at each iteration and terminate the algorithm. However satisfy these conditions require a lot of iterations. On the other hand, the current iterate x^ν, z^ν reaches the required accuracy levels faster than the ergodic iterate. Patrinos *et al.* (PB14b) proposed a computationally tractable algorithm based on the current iterate for the (ϵ_V, ϵ_g) optimal solution and we used the same termination criteria here.

Preconditioning

First-order methods do not include curvature information and their performance deteriorates when applied in the solving of ill-conditioned problems. The proximal gradient iterate is equivalent to minimising the smooth convex upper bounded function (3.27). The quadratic term of this function use same curvature in every direction – the Lipschitz constant $1/\lambda$. But for ill-conditioned problems this serves as a bad approximation and results in poor convergence. In such cases preconditioning transforms the original optimisation problem to a new one by either scaling or applying a linear transformation to the original decision variables. This transformation should be such that it improves the curvature in the transformed space. For the Dual-APG algorithm, the preconditioning has to applied in the dual space. Let us define a preconditioning matrix $M \in \mathbb{R}^{n\times n}$. This preconditioning

[8] weighted sum of the primal iterates $\bar{x}^\nu = (1-\theta_{\nu-1})\bar{x}^{\nu-1} + \theta_\nu x^\nu$, $\bar{z}^\nu = (1-\theta_{\nu-1})\bar{z}^{\nu-1} + \theta_\nu z^\nu$

matrix should be non-singular and its inverse exists. Now the the preconditioned primal problem (3.36) is

$$P^\star = \min_{x,z} f(x) + g(z), \tag{3.49a}$$

subject to

$$\mathrm{MH}x = \mathrm{M}z, \tag{3.49b}$$

and the dual problem is

$$D^\star = \min_{\hat{y}} f^*(-H'\hat{y}) + g^*(\hat{y}), \tag{3.49c}$$

where $\hat{y} = My$.

Here we discuss the preconditioning when $f(x)$ is a quadratic function with a Hessian Q. Then the choice of preconditioning matrix is such that the Lipschitz constant of the dual gradient equal to 1 or

$$\|MHQ^{-1}H'M'\| = 1. \tag{3.50}$$

Exact calculation of the precondition matrix is formulated as minimising the ratio between the minimum and maximum eigenvalues of $MHQ^{-1}H'M'$.

Boyd *et al.* (BEFB94) constructed this problem as a quasiconvex problem solved as a convex semi-definite program. But this method is applicable only for small problems. So Bradley *et al.* (Bra10) suggested the heuristics based approach that find a positive and diagonal M such that the rows and columns of $MHQ^{-1}H'M'$ have same norm. Most popular norms are 1-norm, 2-norm and ∞-norm. These methods do not provide any guarantee for decrease in condition number, but still they work in practice. Moreover, the fact that the preconditioning matrix is diagonal is desirable for it does not alter the structure of the original problem — it merely *scales* the variables.

The simplest of these preconditioning is the ∞-norm also known as Jacobi scaling. This methods computes a diagonal approximation $\tilde{H}$ of the dual Hessian and perform a change of coordinates in the dual variable y with scaling matrix $\tilde{H}^{-\frac{1}{2}}$ (Ber99, Sec. 2.3.1). More elaborate preconditioning schemata have been proposed such as (GB14; GB15).

3.5 Parallelisable APG for stochastic optimal control problems

Stochastic optimal control problems are large-scale problems typically counting millions of decision variables. Nonetheless, such problems possess a structure — that dictated by the scenario tree — which can be exploited by proximal gradient algorithms. The objective function in the stochastic optimal control problem (3.18) is:

$$V(\pi) = \sum_{k=0}^{N-1} \sum_{i=1}^{\mu(k)} p_k^i \ell(x_k^i, u_k^i, i) + \sum_{i=1}^{\mu(N)} p_N^i V_f^i(x_N^i, i) \tag{3.51a}$$

where $\pi = \{x_k^i, u_k^i\}_{k,i}$ are the decision variables which are related by

$$x_{k+1}^j = A_k^j x_k^i + B_k^j u_k^i + w_{k+1}^j, j \in \text{child}(k, i) \tag{3.51b}$$

The decisions are causal in nature, i.e., they are restricted by the information flow defined by the scenario-tree. Here the cost is a summation of the individual cost over time and uncertainty. Without the system dynamics (3.51b), the cost is a summation of independent costs and easily decomposable. The cost functions are linked

- in time through the dynamics of the system
- in scenarios through the information structure dictated by the the scenario-tree.
- in space through interconnected components in the system model which is handy in large-scale distributed systems.

Decomposition is a technique by which a complex system is divide into smaller systems that are easier to solve. This decomposition is not unique and in literature two decomposition means are popular:

1. Nested or chained decomposition
2. Scenario decomposition

Nested decomposition

Nested or chained decomposition is based on the principle of *dynamic programming* (Ber00) — solve a sequence of simple problems; starting

from the smallest problem and extending to bigger problems. In this case these problems are in the order of time from $k = N$ to $k = 0$. Let us define a cost from a stage k to the last stage N as $V(x, k, N)$ — a *cost-to-go*. In dynamic programming, the optimal cost-to-go value from the stage $k+1$ is assumed to be known. Then the cost-to-go value for the stage k is given by the relation

$$V_k^\star(\mathbf{x}_k) = \min_{\mathbf{u}_k} \sum_{i=1}^{\mu(k)} p_k^i \ell(x_k^i, u_k^i, i) + V_{k+1}^\star(\mathbf{x}_{k+1}) \tag{3.52a}$$

where $V_{k+1}^\star(\mathbf{x}_{k+1})$ is the optimal cost from $k+1$ stage. This cost is the summation of the optimal cost at the nodes at the stage $k+1$ and written as:

$$V_{k+1}^\star(\mathbf{x}_{k+1}) = \sum_{i=1}^{\mu(k+1)} V_{k+1}^{\star i}(x_{k+1}^i). \tag{3.52b}$$

Now we have $\mu(k)$ subproblems at stage k which are given as:

$$V_k^{\star i}(x_k^i) = \min_{\mathbf{u_k^i}} \ell(x_k^i, u_k^i, i) + \sum_{j \in \mathrm{child}(k,i)} V^{\star i}(x_{k+1}^j), \text{ for all } i \in \mathbb{N}_{[1,\nu(k)]}. \tag{3.52c}$$

These subproblems are independent and can be solved in parallel across all the nodes at a stage — reusing a popular term we shall characterise such decompositions as *nodal* (BC05). The subproblems resulted through this approach are parametric non-smooth constrained optimisation problems which are hard to solve. This approach does not scale well with the prediction horizon – as the horizon increase the number of constrained problems also increase. Although dynamic programming cannot be applied directly for solving constrained optimisation problems, it will prove very useful in evaluating the gradient of the conjugate function f^* within the APG formulations — see Section 3.5.2.

Scenario decomposition

This decomposition generate subproblems based on the scenario fan – the non-anticipative constraints in the scenario tree are dualised to

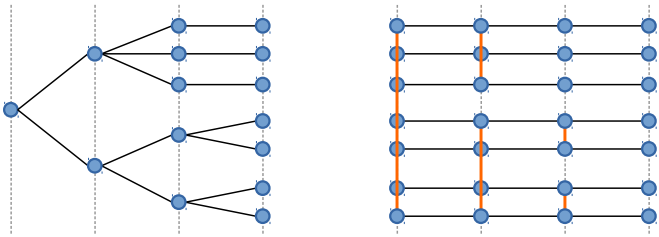

Figure 10: From the original tree, a scenario fan is constructed with additional equality constraints (*non-anticipativity or causality constraints*) here denoted by vertical orange lines.

produces a scenario fan which is path from the root node to the leave node (SDR09). This is depicted in the Figure 10. The original problem is divided into smaller problems that are equal to the number scenarios in the tree. These smaller problems are solved separately and coordinate the decisions to satisfy the non-anticipative constraints. Progressive Hedging algorithm (RW91) is a popular approach that use scenario decomposition.

Let us denote $\hat{x}_k^i, \hat{u}_k^i$ the state and control at stage k for scenario i. The non-anticipative constraint at a node i at stage k is given as

$$u_k^i = \hat{u}_k^j \text{ for } j \in \text{scen}(k,i). \tag{3.53a}$$

Another way to express this constraint is

$$u_k^i = {}^1\!/_{n_k^i} \sum_{j \in \text{scen}(k,i)} \hat{u}_k^j, \tag{3.53b}$$

where n_k^i is the cardinality of the set $\text{scen}(k,i)$.

The original scenario-based optimal control problem (3.18) for scenario fan is given as:

$$V^\star(p) = \min_{\{\hat{x}_k^i, \hat{u}_k^i\}_{i \in \mathbb{N}_{[1,\mu]}}} \sum_{i=1}^{\mu} \sum_{k=0}^{N-1} p_N^i \ell(\hat{x}_k^i, \hat{u}_k^i, j) + p_N^i V_f(\hat{x}_N^i, j), \tag{3.54a}$$

subject to :

$$\hat{x}^i_{k+1} = A^j_k \hat{x}^i_k + B^k_k \hat{u}^i_k + w^j_k, j \in s(N,i), \forall i \in \mathbb{N}_{[1,\mu]} \tag{3.54b}$$

$$\hat{u}^i_k = u^j_k, j \in s(N,i) \tag{3.54c}$$

By dualising the non-anticipativity constraints (3.54c), we obtain μ subproblems which can be solved independently and in parallel and later they coordinate/synchronise to satisfy (3.54c). The parallelisation on GPUs is conditioned by certain factors which discourage the use of the scenario decomposition: (i) the lock-step architecture of GPUs which only allows very simple operations of the same type (additions, multiplications, comparisons, etc) to be performed in parallel, (ii) constraints on the sharing of the available memory among parallel threads, (iii) the availability of libraries that facilitate and enable the development of algorithms. Matrix operations such as performing the operations $\{y_i \leftarrow \alpha y_i + \beta_i A_i x_i\}_i$ all in parallel or computing summations $y \leftarrow \alpha y + \sum_i \beta_i x_i$ are well supported by `cuBLAS` and other libraries for GPU programming (NVI12). However, performing dynamic programming computations all in parallel (that is, each worker to solve a dynamic programming problem) is very complex a problem to solve on a GPU. Besides, compared to the proposed decomposition which we discuss in the next section, the scenario decomposition leads to a problem formulation with a significantly larger number of decision variables without a clear advantage in terms of either computational complexity or speed of convergence.

3.5.1 Dual Decomposition

In this thesis, we proposed a third type of duality based on Fenchel dual formulation. The motivation to propose this decomposition are (i) provide flexibility for non-smooth objective (ii) easy of implementation (iii) able to scale for huge problems both in scenarios and prediction horizon. The duality used in the above scenario decomposition is Lagrangian duality – dualise equality and inequality constraints. This formulation is very restrictive and cannot include non-smooth penalties. The Fenchel dual formulation use conjugates and provide more flexibility (Roc99). In this section we show how the Fenchel dual formulation for the scenario-based optimal control problem (3.18) can be parallelised with proximal gradient method.

Let us define $\mathbf{x} = \{\mathbf{x}_k, \mathbf{u}_k\}$ the collection of the primal decision variable and $\mathbf{y} = \{\mathbf{y}_k\}$ is the dual variables. Let us define the dynamic as $\mathcal{X}(p) = \{\mathbf{x} | x^j_{k+1} = A^j_k x^i_k + B^j_k u^i_k + w^j_{k+1}, \forall k \in \mathbb{N}_{[0,N-1]}, i \in \mathbb{N}_{[1,\mu(k)]}, j \in \text{child}(i,k)\}$. Now the objective function of the stochastic optimal control problem (3.18) with the splitting introduced in (3.35) becomes

$$f(\mathbf{x}) = \sum_{k=0}^{N-1} \sum_{i=1}^{\mu(k)} p^i_k \phi(x^i_k, u^i_k, i) + \sum_{i=1}^{\mu(N)} p^i_N \phi_N(x^i_N, i) + \delta(\mathbf{x} | \mathcal{X}(p)) \tag{3.55a}$$

$$g(H\mathbf{x}) = \sum_{k=0}^{N-1} \sum_{i=1}^{\mu(k)} p^i_k \bar{\phi}(F^i_k x^i_k + G^i_k u^i_k, i) + \sum_{i=1}^{\mu(N)} p^i_N \bar{\phi}_N(F^i_N x^i_N, i), \tag{3.55b}$$

where $F^i_k \in \mathbb{R}^{n^i_{c,k} \times n_x}$, $G^i_k \in \mathbb{R}^{n^i_{c,k} \times n_u}$, $F^i_N \in \mathbb{R}^{n^i_f \times n_x}$. Let us define $\mathbf{z} = H\mathbf{x}$ where

$$H = \begin{bmatrix} \mathbf{H}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_2 & \cdots & \mathbf{0} \\ \vdots & \ddots & \ddots & \vdots \\ \mathbf{0} & \cdots & \cdots & \mathbf{H}_N \end{bmatrix}, \tag{3.55c}$$

where $\mathbf{H}_k = \text{blockdiag}[F^i_k G^i_k]$, for all $i \in \mathbb{N}_{[1,\mu(k)]}$ and all $k \in \mathbb{N}_{[0,N-1]}$ and $\mathbf{H}_N = \text{blockdiag}[F^i_N]$, for all $i \in \mathbb{N}_{[1,\mu(N)]}$.

With this choice of f, g and linear operator H the stochastic optimal control problem is equivalent to (3.35). However this is not the only way to split this problem – another way to split is exchange system dynamic in f (3.55a) with constraints in g (3.55b) (RJM13). But the present choice of splitting allows us to parallelise the computation across all nodes — it effectuates a *nodal decomposition*; this is better explained later in Section 3.5.2. It may be observed that scenario decomposition brings on additional overheads of coordination/synchronisation and would lead to a significantly higher number of decision variables without facilitating the solution of the problem and was, therefore, not further pursued.

Further the linear operator H is associated with the non-smooth term $g(H\mathbf{x})$. Therefore the dual of this problem is formulated which has a structure suitable for the proximal-gradient method and H in

the smooth function. This algorithm can be summarised in the steps in (3.45). The proximal operator with respect to g is inexpensive and $\mathbf{z}^\nu$ is easily evaluated even for large vectors. The main computational burden is calculating the dual gradient which is explained below.

3.5.2 Computation of dual gradient

The dual gradient (3.45b) is the most computational burden in the algorithm. This is also results in the primal iterate for the corresponding dual iterate $\mathbf{x}^\nu(\mathbf{y}^\nu)$ simplify written as $\mathbf{x}^\nu$. Efficient computation of this step is crucial for the performance of the dual proximal gradient. This step (3.45b) can be expanded as:

$$\mathbf{x}^\nu = \underset{\mathbf{x}\in\mathcal{X}(p)}{\operatorname{argmin}} \left\{ \langle \mathbf{x}, H'\mathbf{y}^\nu \rangle + f(\mathbf{x}) \right\}$$

where $f(\mathbf{x})$ is given as (3.55a). This is a standard convex optimisation with equality constraints which are given by the system dynamic equations over the scenario tree. This problem must be solved multiple times, once per iteration for different values of the dual vector $\mathbf{y}^\nu$. A popular method is to build the KKT (Karush-Kuhn-Tucker) matrix and solve the KKT system by eliminating the equality constraints. A Ricatti-like recursion where certain matrices are pre-computed – constituting a factorisation of the underlying dynamic programming procedure – is available for the case of deterministic MPC (OSB13). This method is division-free and the flops required are linear in prediction horizon and quadratic on the size of the dual vector (OSB13). But the dynamics of the stochastic MPC is convoluted than deterministic MPC because of the tree structure dictated by the non-anticipativity constraints. Moreover the dual variable range in millions and pursuing this approach would need require to solve a factorisation which have the same complexity as the original problem. Another disadvantage with this approach is the numerical instabilities with the factorisation of KKT matrix.

Another approach to solve (3.45b) is using dynamic programming[9] which allows to decompose the original problem into smaller prob-

[9] Alternatively, we may also use the approach of *co-state equations* (Boy09). In both cases we end up with the Ricatti-like recursion we shall describe in what follows.

lems. This lead to an equation similar to (3.52a) with an additional linear term based on the dual variable. This is

$$L_k^\star(\mathbf{x}_k, \mathbf{y}_k) = \min_{\mathbf{u}_k} \sum_{i=1}^{\mu(k)} p_k^i \phi(x_k^i, u_k^i, i) + y_k^{i\prime} H_k^i x_k^i + L_{k+1}^\star(\mathbf{x}_{k+1}, \mathbf{y}_{k+1}), \tag{3.56a}$$

subject to

$$x_{k+1}^j = A_k^j x_k^i + B_k^j u_k^i + w_{k+1}^j, j \in \text{child}(k, i), \tag{3.56b}$$

where $L_k^\star(\mathbf{x}_k, \mathbf{y}_k)$ is the *cost-to-go* function from the stage k and the stage cost can be written as a summation of the cost-to-go at the individual nodes $L_{k+1}^\star(\mathbf{x}_{k+1}, \mathbf{y}_{k+1})$ as

$$L_{k+1}^\star(\mathbf{x}_{k+1}, \mathbf{y}_{k+1}) = \sum_{i=1}^{\mu(k+1)} L_{k+1}^{\star i}(\mathbf{x}_{k+1}, \mathbf{y}_{k+1}). \tag{3.57}$$

The minimisation operator has the following separability property: $\min_{x_1,x_2}\{f_1(x_1) + f_2(x_2)\} = \min_{x_1} f_1(x_1) + \min_{x_2} f_2(x_2)$, that is, the optimisation can be split into smaller problems. This property allows to split (3.56a) to $\mu(k)$ smaller problems given by:

$$L_k^{\star i}(x_k^i, y_k^i) = \min_{\mathrm{u}_k^i} p_k^i \phi(x_k^i, u_k^i, i) + y_k^{i\prime} H_k^i x_k^i + \sum_{j \in \text{child}(k,i)} L_{k+1}^{\star j}(x_{k+1}^j, y_{k+1}^j),$$

where $x_{k+1}^j = A_k^j x_k^i + B_k^j u_k^i + w_{k+1}^j$.

The cost-to-go at last stage is written with the terminal cost and the dualised terminal constraints as:

$$\begin{aligned} L_N^\star(\mathbf{x}_N, \mathbf{y}_N) &= \sum_{i=1}^{\mu(N)} L_N^{\star i}(x_N^i, y_N^i) = \sum_{i=1}^{\mu(N)} p_N^i \phi_N(x_N^i, i) + y_N^{i\prime} H_N^i x_N^i \\ &= \sum_{i=1}^{\mu(N)} x_N^{i\prime}(p_N^i P_N^i) x_N^i + y_N^{i\prime} H_N^i x_N^i \end{aligned} \tag{3.58}$$

This is a quadratic cost function and outcome of the dynamic programming iterate at stage $k = N - 1$ is another quadratic function

as:

$$L^{\star}_{N-1}(\mathbf{x}_{N-1}, \mathbf{y}_{N-1}) = \sum_{i=1}^{\mu(N-1)} L^{\star i}_{N-1}(x^i_{N-1}, y^i_{N-1}) \\ = \sum_{i=1}^{\mu(N-1)} x^{i\prime}_{N-1} P^i_{N-1} x^i_{N-1} + q^{i\prime}_{N-1} x^i_{N-1} + c^i_{N-1}, \tag{3.59}$$

where c^i_{N-1} is a constant term. The optimal control action is parametrised in state x^i_{N-1} and the dual vector y^i_{N-1} as

$$u^i_{N-1} = K^i_{N-1} x^i_{N-1} + \Phi^i_{N-1} y^i_{N-1} + \sum_{j \in \text{child}(N-1,i)} \Theta^j_{N-1} q^j_N + \sigma^i_{N-1}. \tag{3.60a}$$

with

$$\bar{P}^i_N = \sum_{j \in \text{child}(N-1,i)} B^{j\prime}_N P^j_N B^j_N, \tag{3.60b}$$

$$\bar{R}^i_{N-1} = 2(p^i_{N-1} R^i_{N-1} + \bar{P}^i_N), \tag{3.60c}$$

$$\Phi^i_{N-1} = -(\bar{R}^i_{N-1})^{-1} G^{i\prime}_{N-1}, \tag{3.60d}$$

$$K^i_{N-1} = -2(\bar{R}^i_{N-1})^{-1} \sum_{j \in \text{child}(N-1,i)} B^{j\prime}_N P^j_N A^j_N, \tag{3.60e}$$

$$\sigma^i_{N-1} = -2(\bar{R}^i_{N-1})^{-1} \sum_{j \in \text{child}(N-1,i)} B^{j\prime}_N P^j_N w^j_N, \tag{3.60f}$$

$$\Theta^j_{N-1} = -(\bar{R}^j_{N-1})^{-1} B^{j\prime}_N F^{j\prime}_N. \tag{3.60g}$$

This dynamic programming iterate is carried till the first stage $k = 0$. This whole process can be summarised in the two step algorithm: the factor step (Algorithm 4) and the solution step (Algorithm 5). In case of one scenario as in deterministic optimal control these algorithms are the factor and solve steps described in (PB14b). Notice that both algorithms are parallelisable across all nodes at every stage of the scenario tree.

The factor step is performed once before the execution of the APG algorithm, and produces the matrices Φ^i_k, Θ^i_k, Λ^i_k, D^i_k, K^i_k and vectors

Algorithm 4 Factor step

for $k = N-1, \ldots, 0$ **do**
  for $i \in \mu(k)$ **do** {in parallel}
    $\bar{P}^i_{k+1} \leftarrow \sum_{j \in \text{child}(k,i)} B^{j\prime}_{k+1} P^j_{k+1} B^j_k$
    $\bar{R}^i_k \leftarrow 2(p^i_k R^i_k + \bar{P}^i_{k+1})$, $\Phi^i_k = -(\bar{R}^i_k)^{-1} G^{i\prime}_k$
    $K^i_k \leftarrow -2(\bar{R}^i_k)^{-1} \sum_{j \in \text{child}(k,i)} B^{j\prime}_{k+1} P^j_{k+1} A^j_k$
    $\sigma^i_k \leftarrow -2(\bar{R}^i_k)^{-1} \sum_{j \in \text{child}(k,i)} B^{j\prime}_k P^j_{k+1} w^j_{k+1}$
    $\bar{A}^j_k \leftarrow A^j_k + B^j_k K^i_k, \forall j \in \text{child}(k,i)$
    $D^j_k \leftarrow F^j_k + G^j_k K^i_k, \forall j \in \text{child}(k,i)$
    $c^i_k \leftarrow 2 \sum_{j \in \text{child}(k,i)} \bar{A}^{j\prime}_k P^j_{k+1} w^j_{k+1}$
    if $k = N-1$ **then**
      $\Theta^j_{N-1} \leftarrow -(\bar{R}^j_{N-1})^{-1} B^{j\prime}_N F^{j\prime}_N, \forall j \in \text{child}(N-1,i)$
      $\Lambda^j_{N-1} \leftarrow F^j_N \bar{A}^{j\prime}_N, \forall j \in \text{child}(N-1,i)$
    else
      $\Theta^j_k \leftarrow -(\bar{R}^j_k)^{-1} B^{j\prime}_{k+1}, \forall j \in \text{child}(k,i)$
      $\Lambda^j_k \leftarrow \bar{A}^{j\prime}_k, \forall j \in \text{child}(k,i)$
    end if
    $P^i_k \leftarrow p^i_k (Q^i_k + K^{i\prime}_k R^i_k K^i_k) + \sum_{j \in \text{child}(k,i)} \bar{A}^{j\prime}_{k+1} P^j_{k+1} \bar{A}^j_k$
  end for
end for

σ^i_k and c^i_k which are then used in Algorithm 5 to calculate the dual gradient for a given dual vector $\mathbf{y}^\nu$.

In Algorithm 4 we do not need to compute the inverse of $\bar{R}^i_k$; since this is a symmetric positive definite matrix we may, for instance, compute its Cholesky factorisation and solve the involved linear systems with it. The computations required in the factor step can be parallelised across the nodes of the scenario tree at each stage as we may observe in Algorithm 4. As these computations are carried out once for each execution of the APG, the computational overhead associated with this algorithm can be neglected [10]. Moreover, when for a scenario

[10] We shall demonstrate with simulations that the runtime of the factor step is much smaller compared to the total runtime of APG.

tree structure and the system dynamics is fixed, these calculations can be carried off-line.

Algorithm 5 Solve step

$q_N^i \leftarrow y_N^i, \forall i \in \mathbb{N}_{[1,\mu(N)]}$, `%Backward substitution`
for $k = N-1, \ldots, 0$ **do**
  for $i \in \mu(k)$ **do** {in parallel}
    $u_k^i \leftarrow \Phi_k^i y_k^i + \sum_{j \in \text{child}(k,i)} \Theta_k^j q_{k+1}^j + \sigma_k^i$
    $q_k^i \leftarrow D_k^{i\prime} y_k^i + \sum_{j \in \text{child}(k,i)} \Lambda_k^{j\prime} q_{k+1}^j + c_k^i$
  end for
end for
$x_0^1 = p$, `%Forward substitution`
for $k = 0, \ldots, N-1$ **do**
  for $i \in \mu(k)$ **do** {in parallel}
    $u_k^i \leftarrow K_k^i x_k^i + u_k^i$
    for $j \in \text{child}(k,i)$ **do** {in parallel}
      $x_{k+1}^j \leftarrow A_k^j x_k^i + B_k^j u_k^i + w_k^j$
    end for
  end for
end for

The solve step (Algorithm 5) is executed at every iterate of the APG algorithm to calculate the dual gradient or primal iterate ($\{x_k^i, u_k^i\}_{k,i}$). In fact this step does not require any matrix-matrix multiplications or expensive factorisations. The only computations involved are matrix-vector products. This algorithm has two parts: the *backward substitution* and the *forward substitution*. In the backward substitution, the tree is traversed from the leaf nodes to the root node to calculate the offset that depends on the dual vectors. In the forward substitution, the tree is traversed from the root to the leaf nodes using the system dynamics to calculate $\{x_k^i, u_k^i\}_{k,i}$. This algorithm is parallelisable stage-wise across all nodes of the scenario tree at each stage.

3.5.3 GPU for parallel computations

From computational viewpoint, the factor and solve step are stage-wise parallelisable algorithms. Therefore, it is advantageous to use a hardware that can support parallel execution which could exploit this feature. Graphics Processing Unit (GPU) are special hardware units for parallel computing. Traditional hardware like CPU have few but powerful cores (typically 8 to 16 cores) whereas GPUs have smaller cores but many cores(ranging from 256 to 1024). So GPUs are capable of performing many simple operations in parallel and are well-suited for data-parallel lock-step computations — the same program is executed on many data elements in parallel. In both factor and solve steps, the operations performed at each node are similar and all the operations can be executed in parallel with GPUs.

Programming multi-core machines is more complicated as it need additional overheads with synchronisation than single core machine. To facilitate programming GPUs, special programming models have been developed. One such programming model is CUDA — an NVIDIA proprietary programming model that supports NVIDIA's GPUs. CUDA-C (NBGS08) is developed as an extension of C with additional primitives that simplify and allow massive parallelism. CUDA introduces three levels of abstraction for the organisation and distribution of parallel execution: *threads* which are the very basic parallel computational entities, *groups* which are bundles of threads inside which parallel memory sharing is possible and *grids* which are the highest level of organisation.

In addition, NVIDIA provides many libraries for CUDA. The library `cuBLAS` is BLAS's counterpart for CUDA and it is available for dense linear algebra operations (NVI12). We used routines from this library to implement the APG algorithm. It is advised to use them instead of implementing custom code as these routines are optimised for a large gamut of architectures and are updated to follow changes in new architectures.

Computations complexity with GPUs: In order the analyse the computational complexity of Algorithm 4 we consider a the m-ary scenario tree, that is a scenario tree with constant branching factor m. We also assume that the row-dimension of all F_k^i is constant and equal to n_c.

The first part of the algorithm is backward substitution where we

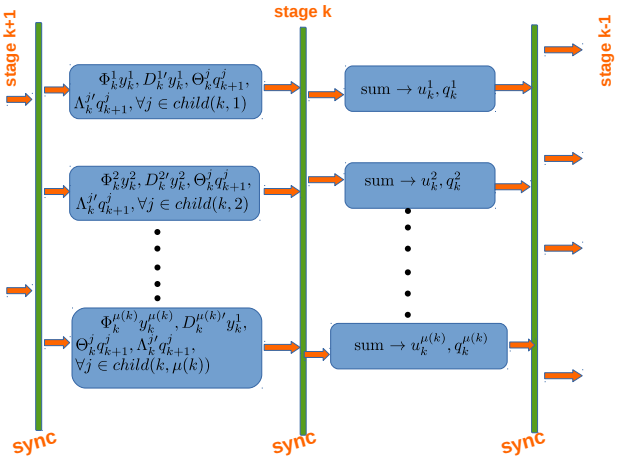

Figure 11: Parallelisation in backward substitution of Algorithm 5

traverse backward from stage $k = N$ to $k = 0$. From the algorithm we can notice that the computations performed at stage $k = i$ are

$$u_k^i = \Phi_k^i y_k^i + \sum_{j \in \text{child}(k,i)} \Theta_k^j q_{k+1}^j + \sigma_k^i, \text{ for all } i \in \mathbb{N}_{[1,\mu(k)]},$$

$$q_k^i = D_k^{i\prime} y_k^i + \sum_{j \in \text{child}(k,i)} \Lambda_k^{j\prime} q_{k+1}^j + c_k^i \text{ for all } i \in \mathbb{N}_{[1,\mu(k)]}.$$

At every node, we require $2(1+m)$ matrix-vector products - one each $\Phi_k^i y_k^i$ and $D_k^{i\prime} y_k^i$, m each $\Theta_k^j q_{k+1}^j$ and $\Lambda_k^{j\prime} q_{k+1}^j$. Altogether we require $2m^i(1+m)$ matrix-vector. All these matrix-vector products are independent and can be performed in parallel in the GPU using the `cuBLAS` routine `cublasSgemmBatched`. Once the multiplications are completed these vectors are summed in parallel. This process continued till first stage and depicted in Figure 11. Roughly this step involves $\mathcal{O}(N(n_x(n_u+n_c)+n_u(n_x+n_c)))$ flops and with perfect parallelisation this step is equivalent to $\mathcal{O}(Nm(n_x(n_u+n_c)+n_u(n_x+n_c)))$.

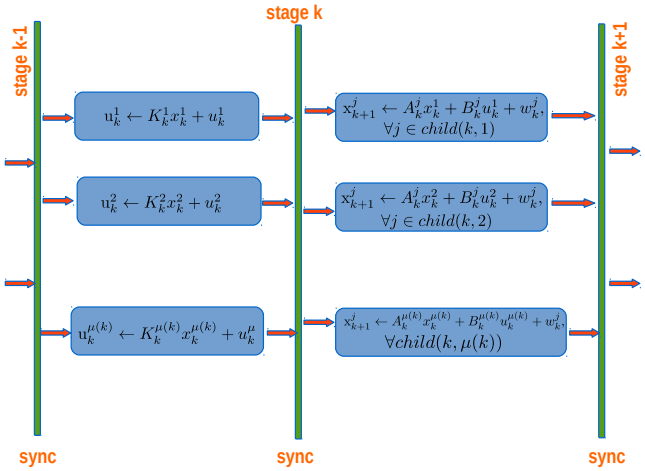

Figure 12: Parallelisation in forward substitution of Algorithm 5

The second part of the algorithm we traverse forward from stage $k = 0$ to $k = N$. At stage $k = i$, we have

$$u_k^i = K_k^i x_k^i + u_k^i, \text{ for all } i \in \mathbb{N}_{[1,\mu(k)]}, \tag{3.61}$$

$$x_{k+1}^j = A_k^j x_k^i + B_k^j u_k^i + w_k^j, j \in \text{child}(k,i), i \in \mathbb{N}_{[1,\mu(k)]}. \tag{3.62}$$

In this part we have two steps which need to be performed sequentially - first u_k^i is calculated and after this x_{k+1}^j are updated. The first step require m^i matrix-vector product, $K_k^i x_k^i$, which are independent and computed in parallel. In the next step, there are $2m^{i+1}$ matrix-vector products, $A_k^j x_k^i$, $B_k^j u_k^i$, but these are computed with $2m^i$, x_k^i, u_k^i, vectors. Therefore we have $2m^i$ parallel operations in GPU. This process is continued till the last stage $k = N$ and shown in Figure 12. Overall this has a rough count of $\mathcal{O}(N\mu(n_x^2 + 2n_x n_u))$ flops and with perfect parallelisation we have $\mathcal{O}(Nm(N\mu(n_x^2 + 2n_x n_u)))$ flops.

3.6 Simulation results

To evaluate the proposed algorithm we formulate the stochastic optimal control problem which corresponds to the stochastic model predictive control problem for a linear discrete-time system with additive and parametric uncertainty as in (3.8). We consider a system of m aligned interconnected masses by $m-1$ linear spring-dampers of stiffness constant $\kappa = 1$ and damping ratio $\beta = 0.1$. The manipulated variables are the forces we may exercise on each spring along their principal axes and the state variables are the positions and speeds of the masses. We assume that the system dynamics is obtained by discretising the continuous-time dynamics with sampling time $T_s = 0.5$ and is written as in (3.8) with $n_x = 2m$, and $n_u = m-1$. On the system state and input variables we impose the constraints $-5 \leq x_k^i \leq 5$ and $-1 \leq u_k^i \leq 1$ for all $k \in \mathbb{N}_{[0,1]}$ and $i \in \mathbb{N}_{[1,\mu(k)]}$. The stage cost was chosen to be $\ell(x,u,\xi) = x'Qx + u'Ru$ with $Q = I_{n_x}$ and $R = I_{n_u}$.

APG was implemented in CUDA-C (NBGS08) as presented in the previous section and matrix-vector multiplications were performed using cuBLAS. We compared the GPU-based implementation of APG with the interior point solver of Gurobi which runs on a dual-core environment. The active set algorithm of Gurobi, as well as qpOASES (FKP$^+$14) and QPC (SPM10) give computation times that are not very competent, and will therefore be omitted.

Computations on CPU were performed on a $4 \times 2.60\,\text{GHz}$ Intel i5 machine with $8\,\text{GB}$ RAM running 64-bit Ubuntu 14.04 and GPU-based computations were carried out on a NVIDIA Tesla C2075 using the CUDA-6.0 API.

The dependence of the computation time on the size of the scenario tree is shown in Fig. 13; trees considered in this experiment had a fixed horizon $N = 14$ and in their first stages were binary, i.e., had branching factor 2 and eventually evolved without branching until the end of the horizon. Notice that for a case of 8192 leaf nodes, Gurobi takes $32.6\,\text{s}$ on average (max. $46.8\,\text{s}$), whereas APG with $\epsilon_g = \epsilon_V = 0.005$ requires just $1.3\,\text{s}$ (max. $5.92\,\text{s}$). This problem counts $6.39 \cdot 10^5$ primal variables, and $1.75 \cdot 10^6$ dual variables.

In Fig. 14 we show how the computation times scale with the increase of the horizon of the problem. The problem that corresponds to $N = 60$ counts $0.92 \cdot 10^6$ primal and $2.0 \cdot 10^6$ dual variables and notice that APG with $\epsilon_g = \epsilon_V = 0.005$ can solve it 17.6 times faster

than Gurobi on average (6.43 times faster for the maximum time).

Compared to a MATLAB implementation, a very high speedup is achieved on GPU for the same algorithm which can be up to $\times 85$ and scales with the problem size as shown in Fig. 15. On average, the GPU implementation of APG for a scenario tree of 8192 leaf nodes is as high as $\times 83$.

3.7 Conclusions

In this chapter we proposed a dual accelerated proximal gradient algorithm tailored for the solution of stochastic optimal control problems. The computation of the dual gradient at every iteration of the algorithm can be parallelised to offer a significant benefit in terms of speed-up. In particular computations are executed in parallel across all nodes at every stage of the scenario tree. As a result, for the special case of a tree with branching only at the root node (known as a *scenario fan*) stochastic MPC can be solved at the computational cost of deterministic MPC provided that the GPU has adequate computational capacity to accommodate the problem size. A CUDA-C implementation of the algorithm that runs on a GPU was found to outperform most state-of-the-art solvers that run on a multi-core CPU.

Now we have an optimisation method that can solve the stochastic MPC problem efficient. In the next chapter, we apply this method to solve the optimisation problem associated with the SMPC formulation on the DWN.

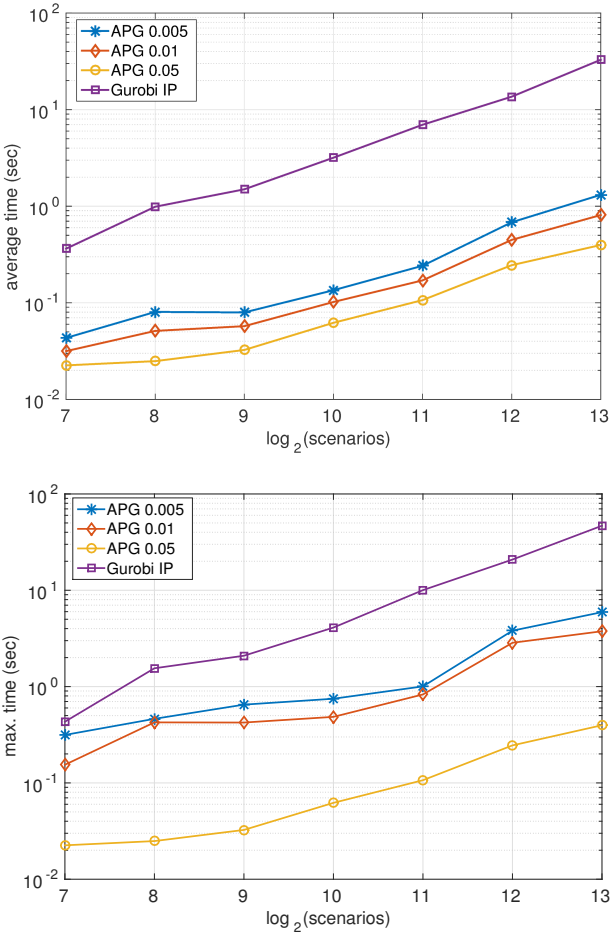

Figure 13: Dependence of the computation time on the number of scenarios for a system of 10 masses (20 states, 9 inputs, bound constraints) with a fixed prediction horizon $N = 14$. Average and maximum computation times reported here are for a random sampling of 100 initial states $x_0 = p$.

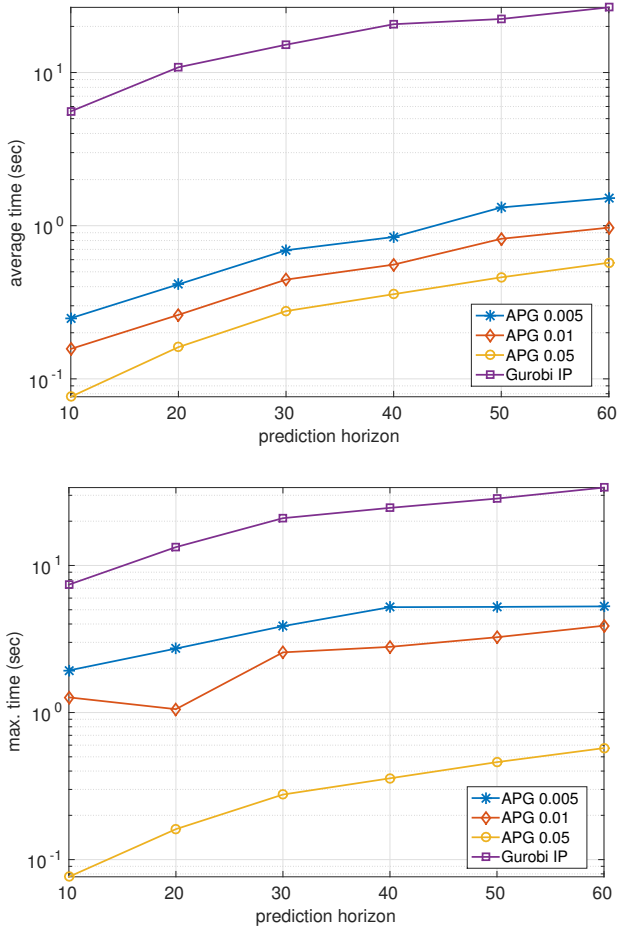

Figure 14: Dependence of the computation time on the prediction horizon for a system of 10 masses and 500 scenarios. Average and maximum computation times reported here are for a random sampling of 100 initial states $x_0 = p$.

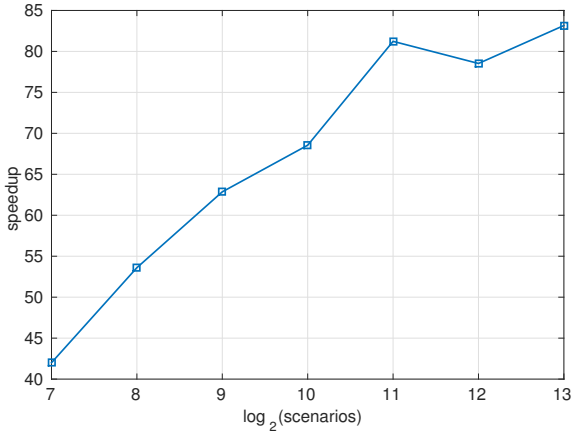

Figure 15: Speedup with respect to a CPU implementation of APG vs. the size of the scenario tree.

Chapter 4

Stochastic predictive control applied to drinking water networks

Despite the proven advantages of scenario-based stochastic model predictive control for the operational control of water networks, its applicability is limited by its considerable computational footprint. In the previous chapter, we devised a dual proximal gradient algorithm that exploit their structure and can execute in parallel. In this chapter, we apply this algorithm to solve the stochastic MPC formulation for the drinking water network. The proposed methodology is applied and validated on a case study: the water network of the city of Barcelona. The results presented in this chapter have been provincially accepted to publish and its archive version is (SSBP16).

4.1 Introduction

Operation of drinking water networks is a complex decision system. By integrating real-time forecasting with model predictive control the reliability of the system improved. At the same, this formulation take a conservative approach towards the forecast error. Stochastic model predictive control is an advanced control scheme which can address effectively the above challenges and has already been used for the management of water networks (GOMPJ14; GMOMP14). However, unless restrictive assumptions are adopted regarding the form of the disturbances, such problems are known to be computationally intractable (GMOMP14; NS06). But combining parallel accelerated dual proximal-gradient algorithms with general-purpose graphics processing units (GPGPUs) to deliver a computationally feasible solution for the control of water networks.

4.1.1 Background

The *pump scheduling problem* (PSP) is an optimal control problem for determining an open-loop control policy for the operation of a water network. Such open-loop approaches are known since the 80's (Cre98; ZS89). More elaborate schemes have been proposed such as (YPS94) where a non-linear model is used along with a demand forecasting model to produce an optimal open-loop 24-hour-ahead policy. Recently, the problem was formulated as a mixed-integer non-linear program to account for the on/off operation of the pumps (BBMJ$^+$13). Heuristic approaches using evolutionary algorithms, genetic algorithms, and simulated annealing have also appeared in the literature (MP04). However, a common characteristic and shortcoming of these studies is that they assume to know the future water demand and they do not account for the various sources of uncertainty which may alter the expected smooth operation of the network.

The effect of uncertainty can be attenuated by feedback from the network combined with the optimisation of a performance index taking into account the system dynamics and constraints as in PSP. This, naturally, gives rise to model predictive control (MPC) which has been successfully used for the control of drinking water networks (SGS$^+$14; OMPC$^+$09). Recently, Bakker *et al.* demonstrated experimentally on five full-scale water supply systems that MPC will lead to a more

efficient water supply and better water quality than a conventional level controller (BVP+13). Distributed and decentralised MPC formulations have been proposed for the control of large-scale water networks (LZNDS10; OMFBP10) while MPC has also been shown to be able to address complex system dynamics such as the Hazen-Williams pressure-drop model (SKN+15).

Most MPC formulations either assume exact knowledge of the system dynamics and future water demands (OMPC+09; OMFBP10) or endeavour to accommodate the worst-case scenario (SGS+14; TB09; GN14; WM97). The former approach is likely to lead to adverse behaviour in presence of disturbances which inevitably act on the system while the latter turns out to be too conservative as we will later demonstrate in this chapter.

When probabilistic information about the disturbances is available it can be used to refine the MPC problem formulation. The uncertainty is reflected onto the cost function of the MPC problem deeming it a random variable; in *stochastic MPC* (SMPC) the index to minimise is typically the expectation of such a random cost function under the (uncertain) system dynamics and state/input constraints (CKW09; BB12).

SMPC leads to the formulation of optimisation problems over spaces of random variables which are, typically, infinite-dimensional. Assuming that disturbances follow a normal probability distribution facilitates their solution (vHB02; BB07; NS06); however, such an assumption often fails to be realistic. The normality assumption has also been used for the stochastic control of drinking water networks aiming at delivering high quality of services – in terms of demand satisfaction – while minimising the pumping cost under uncertainty (GOMPJ14).

An alternative approach, known as *scenario-based stochastic MPC*, treats the uncertain disturbances as discrete random variables without any restriction on the shape of their distribution (CC06; CGP09; PGL12). The associated optimisation problem in these cases becomes a discrete multi-stage stochastic optimal control problem (SDR09). Scenario-based problems can be solved algorithmically, however, their size can be prohibitively large making them impractical for control applications of water networks as pointed out by Goryashko and Nemirovski (GN14). This is demonstrated by Grosso *et al.* who provide a comparison of the two approaches (GMOMP14). Although compression methodologies have been proposed – such as the scenario tree

generation methodology of Heitsch and Römisch (HR09) – multi-stage stochastic optimal control problems may still involve up to millions of decision variables.

Graphics processing units (GPUs) have been used for the acceleration of the algorithmic solution of various problems in signal processing (McC07), computer vision and pattern recognition (BMTVG12) and machine learning (JPJ08; GDP09) leading to a manifold increase in computational performance. To the best of knowledge, GPU technology is not exploited in a stochastic optimal control problem.

4.1.2 Outline

This chapter addresses this challenge by devising an optimisation algorithm which makes use of the problem structure and sparsity. It exploits the structure of the problem, which is dictated by the structure of the scenario tree, to parallelise the involved operations. Then, the algorithm runs on a GPU hardware leading to a significant speed-up.

In Section 4.2, we reiterated the DWN model with stochastic demand model and use this model to formulate the stochastic MPC problem in the next Section 4.3. In Section 4.4, we solve the stochastic MPC problem with the Accelerated dual proximal gradient algorithm. Finally in the Section 4.5, we consider the Barcelona network as the case study and compared the computational times of the APG method against the commercial solver Gurobi. Here we also studied the closed-loop performance of the system in terms of the quality of service and process economics. It is shown that the number of scenarios allows us to refine our representation of uncertainty and trade the economic operation of the network for reliability and quality of service.

4.1.3 Mathematical preliminaries

Let $\bar{\mathbb{R}} = \mathbb{R} \cup \{+\infty\}$ denote the set of extended-real numbers. The set of of nonnegative integers $\{k_1, k_1 + 1, \ldots, k_2\}, k_2 \geq k_1$ is denoted by $\mathbb{N}_{[k_1,k_2]}$. For $x \in \mathbb{R}^n$ we define $[x]_+$ to be the vector in $\mathbb{R}^n$ whose i-th element is $\max\{0, x_i\}$. For a matrix $A \in \mathbb{R}^{n\times m}$ we denote its transpose by A'.

The *indicator function* of a set $C \subseteq \mathbb{R}^n$ is the extended-real valued function $\delta(\cdot|C) : \mathbb{R}^n \to \bar{\mathbb{R}}$ and it is $\delta(x|C) = 0$ for $x \in C$ and $\delta(x|C) =$

$+\infty$ otherwise. A function $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ is called *proper* if there is a $x \in \mathbb{R}^n$ so that $f(x) < \infty$ and $f(x) > -\infty$ for all $x \in \mathbb{R}^n$. A proper convex function $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ is called *lower semi-continuous* or *closed* if for every $x \in \mathbb{R}^n$, $f(x) = \liminf_{z \to x} f(z)$. For a proper closed convex function $f : \mathbb{R}^n \to \bar{\mathbb{R}}$, we define its *conjugate* as $f^*(y) = \sup_x \{y'x - f(x)\}$. We say that f is *σ-strongly convex* if $f(x) - \frac{\sigma}{2}\|x\|_2^2$ is a convex function. Unless otherwise stated, $\|\cdot\|$ stands for the Euclidean norm.

4.2 Modelling of drinking water networks

In this section, the flow-based model of the DWN from the Section 2.2.1 is reiterated. Now the demand is modelled as a stochastic process. Using the forecasting models of Section 2.3, the water demand is predicted.

4.2.1 Flow-based control-oriented model

Dynamical models of drinking water networks have been studied in depth in the last two decades (OMPC$^+$09; MF84; OMFBP10). Flow-based models are derived from simple mass balance equations of the network which lead to the following pair of equations

$$x_{k+1} = Ax_k + Bu_k + G_d d_k, \tag{4.1a}$$

$$0 = Eu_k + E_d d_k, \tag{4.1b}$$

where $x \in \mathbb{R}^{n_x}$ is the state vector corresponding to the volumes of water in the storage tanks, $u \in \mathbb{R}^{n_u}$ is the vector of manipulated inputs and $d \in \mathbb{R}^{n_d}$ is the vector of water demands. [1] Equation (4.1a) forms a linear time-invariant system with additive uncertainty and (4.1b) is an algebraic input-disturbance coupling equation with $E \in \mathbb{R}^{n_e \times n_u}$ and $E_d \in \mathbb{R}^{n_e \times n_d}$ where n_e is the number of *junctions* in the network.

[1] The operation of valves is also accounted for by model (4.1a), (4.1b). Valves receive set-points where a local controller manipulates them to achieve the desired flow. The input vector u_k comprises of pumping actions and valve actions — both are flow set-points and flow constraints are imposed by (4.2a). Overall, the model we use in this chapter has been validated with actual data from the water network of Barcelona — see (SSG$^+$13) for details.

The maximum capacity of the tanks and the maximum pumping capacity of each pumping station is described by the following bounds:

$$u_{\min} \leq u_k \leq u_{\max}, \tag{4.2a}$$

$$x_{\min} \leq x_k \leq x_{\max}. \tag{4.2b}$$

In particular (4.2a) imposes constraints on the flow set-points which are sent out to the valves and pumps of the network and (4.2b) imposes constraints on the minimum and maximum allowed volumes of water in each tank. The above formulation has been widely used in the formulation of model predictive control problems for DWNs (SGS$^+$14; GOMPJ14; OMFBP10).

4.2.2 Demand prediction model

The water demand is the main source of uncertainty that affects the dynamics of the network. In the previous chapter 2.3, three different time series models have been proposed for the forecasting of future water demands– seasonal Holt-Winters, seasonal ARIMA, BATS and SVM (SGS$^+$14; WOPQ14). These models can be used to predict nominal forecasts of the upcoming water demand along a horizon of N steps ahead using measurements available up to time k, denoted by $\hat{d}_{k+j|k}$. Then, the actual future demands d_{k+j} — which are unknown to the controller at time k — can be expressed as

$$d_{k+j}(\epsilon_j) = \hat{d}_{k+j|k} + \epsilon_j, \tag{4.3}$$

where ϵ_j is the demand prediction error which is a random variable on a probability space $(\Omega_j, \mathfrak{F}_j, \mathrm{P}_j)$ and for convenience we define the tuple $\boldsymbol{\epsilon}_j = (\epsilon_0, \epsilon_1, \ldots, \epsilon_j)$ which is a random variable in the product probability space. We also define $\hat{\mathbf{d}}_k = (\hat{d}_{k|k}, \ldots \hat{d}_{k+N-1|k})$.

4.3 Stochastic MPC for DWNs

In this section, the control objectives for the controlled operation of a DWN are defined and the stochastic MPC problem is formulated.

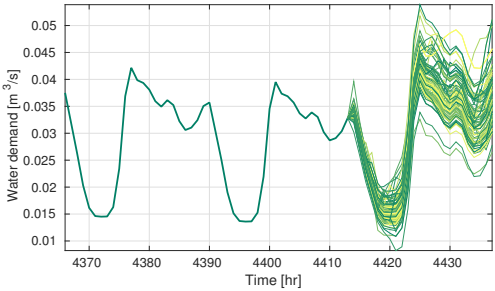

Figure 16: Collection of possible upcoming demands at a given time instant. These results were produced using the SVM model and the data in (SGS+14).

4.3.1 Control objectives

The following three cost functions which reflect the control objectives are defined. The *economic cost* quantifies the *production* and *transportation* cost

$$\ell^w(u_k, k) = W_\alpha(\alpha_1 + \alpha_{2,k})' u_k, \tag{4.4}$$

where the term $\alpha_1' u_k$ is the water production cost, $\alpha_{2,k}' u_k$ is the pumping (electricity) cost and W_α is a positive scaling factor.

The *smooth operation cost* is defined as

$$\ell^\Delta(\Delta u_k) = \Delta u_k' W_u \Delta u_k, \tag{4.5}$$

where $\Delta u_k = u_k - u_{k-1}$ and $W_u \in \mathbb{R}^{n_u \times n_u}$ is a symmetric positive definite weight matrix. It is introduced to penalise abrupt switching of the actuators (pumps and valves).

The *safety operation cost* penalises the drop of water level in the tanks below a given *safety level*. An elevation above this safety level ensures that there will be enough water in unforeseen cases of unexpectedly high demand and also maintains a minimum pressure for the flow of water in the network[2]. This is given by

$$\ell^S(x_k) = W_x d(x_k \mid \mathcal{C}_s), \tag{4.6}$$

[2] Equation (4.6) serves as a soft constraint: it enforces the requirement that the water in each tank should remain above a safety level — the safety storage

where $d(x \mid \mathcal{C}) = \inf_{y \in \mathcal{C}} \|x - y\|_2$ is the distance-to-set function, $\mathcal{C}_s = \{x \mid x \geq x_s\}$, and $x_s \in \mathbb{R}^{n_x}$ is the safety level and W_x is a positive scaling factor.

These cost functions have been used in many MPC formulations in the literature (SGS$^+$14; GOMPJ14; CCPC14). A comprehensive discussion on the choice of these cost functions can be found in (OMFBP10).

The total *stage cost* at a time instant k is the summation of the above costs and is given by

$$\ell(x_k, u_k, u_{k-1}, k) = \ell^w(u_k, k) + \ell^{\Delta}(\Delta u_k) + \ell^S(x_k). \tag{4.7}$$

4.3.2 SMPC formulation

Let's formulate the stochastic MPC problem with decision variables as $\pi = \{u_{k+j|k}, x_{k+j+1|k}\}_{j \in \mathbb{N}_{[0,N-1]}}$

$$V^\star(p, q, \hat{\mathbf{d}}_k, k) = \min_{\pi} \mathbb{E} V(\pi, p, q, k), \tag{4.8a}$$

where $\mathbb{E}$ is expectation operator and

$$V(\pi, p, q, k) = \sum_{j=0}^{N-1} \ell(x_{k+j|k}, u_{k+j|k}, u_{k+j-1|k}, k+j) \tag{4.8b}$$

subject to the constraints

$$x_{k|k} = p,\ u_{k-1|k} = q, \tag{4.8c}$$

$$x_{k+j+1|k} = Ax_{k+j|k} + Bu_{k+j|k} + G_d d_{k+j|k}(\epsilon_j), \quad j \in \mathbb{N}_{[0,N-1]}, \epsilon_j \in \Omega_j \tag{4.8d}$$

$$Eu_{k+j|k} + E_d d_{k+j|k}(\epsilon_j) = 0, j \in \mathbb{N}_{[0,N-1]}, \epsilon_j \in \Omega_j, \tag{4.8e}$$

$$x_{\min} \leq x_{k+j|k} \leq x_{\max}, j \in \mathbb{N}_{[1,N]}, \tag{4.8f}$$

$$u_{\min} \leq u_{k+j|k} \leq u_{\max}, j \in \mathbb{N}_{[0,N-1]}, \tag{4.8g}$$

where it is stressed out that the decision variables $\{u_{k+j|k}\}_{j=0}^{j=N-1}$ are required to be causal control laws of the form

$$u_{k+j|k} = \varphi_{k+j|k}(p, q, x_{k+j|k}, u_{k+j-1|k}, \boldsymbol{\epsilon}_j). \tag{4.8h}$$

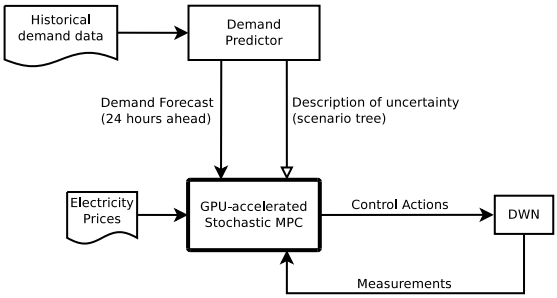

Figure 17: The closed-loop system with the proposed stochastic MPC controller running on a GPU device.

Solving the above problem would involve the evaluation of multi-dimensional integrals over an infinite-dimensional space which is computationally intractable. Hereafter, however, it shall be assumed that all Ω_j, for $j \in \mathbb{N}_{[0,N-1]}$, are finite sets. This assumption will allow to restate (4.8) as a finite-dimensional optimisation problem.

4.3.3 Scenario trees

A scenario tree is the structure which naturally follows from the finiteness assumption of Ω_j and is illustrated in Fig. 18. A scenario tree describes a set of possible future evolutions of the state of the system known as *scenarios*.

The nodes of a scenario tree are partitioned in *stages*. The (unique) node at stage $k = 0$ is called *root* and the nodes at the last stage are the *leaf nodes* of the tree. We denote the number of leaf nodes by n_s. The number of nodes at stage k is denoted by $\mu(k)$ and the total number of nodes of the tree is denoted by μ. A path connecting the root node with a leaf node is called a *scenario*. Non-leaf nodes define a set of *children*; at a stage $j \in \mathbb{N}_{[0,N-1]}$ for $i \in \mu(j)$ the set of children of the i-th node is denoted by $\text{child}(j, i) \subseteq \mathbb{N}_{[1,\mu(j+1)]}$. At stage $j \in \mathbb{N}_{[1,N]}$

volume x_s. This minimum volume corresponds to a minimum elevation and a minimum pressure at the outflow of each tank. The exact value of x_s is a technical specification which is provided by the network operator.

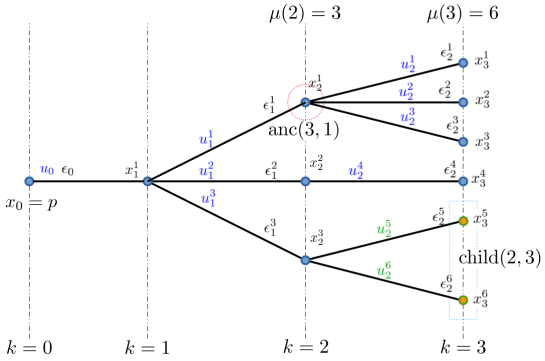

Figure 18: Scenario tree describing the possible evolution of the system state along the prediction horizon: Future control actions are decided in a non-anticipative (causal) fashion; for example u_1^2 is decided as a function of ϵ_1^2 but not of any of ϵ_2^i, $i \in \mathbb{N}_{[1,\mu(3)]}$.

the i-th node $i \in \mathbb{N}_{[1,\mu(j)]}$ is reachable from a single node at stage $k-1$ known as its *ancestor* which is denoted by $\text{anc}(j,i) \in \mathbb{N}_{[1,\mu(j-1)]}$.

The probability of visiting a node i at stage j starting from the root is denoted by p_j^i. For all for $j \in \mathbb{N}_N$ we have that $\sum_{i=1}^{\mu(j)} p_j^i = 1$ and for all $i \in \mathbb{N}_{[1,\mu(k)]}$ it is $\sum_{l \in \text{child}(j,i)} p_{j+1}^l = p_j^i$.

We define the maximum branching factor at stage j, b_j, to be the maximum number of children of the nodes at this stage. The maximum branching factor serves as a measure of the complexity of the tree at a given stage.

4.3.4 Generation of the scenario tree

Scenario tree is constructed from raw data which is normally obtained from historical time-series information. This data represents the discrete probability measure of the stochastic process. The scenario tree should approximate this probability distribution with a predefined structure specified by the branching factor at each stage. Wallace *et.al* (HW01)

suggested a technique that matches moments to generate the scenario tree. But this methods lacks theoretical support and similar moments does not guarantee similarity in their distributions.

Here we generate the tree based on a probability distance minimisation algorithm proposed by Heitsch *et al.* (HR09). Let us assume the probability measure of the original process is $\mathbb{P}$ and then generate a scenario tree with a predefined complexity and has a probability measure $\mathbb{Q}$ such that two probability measures are closed in terms of *Wasserstein-Kantorovitch* metric or simply the transportation distance. For discrete probability measure this is a linear programming problem and can be solved easily using a commercial solver. We generated the scenario tree using the *forward selection* algorithm given in by Heitsch *et al.* (HR03; DGKR03).

4.3.5 Reformulation as a finite-dimensional problem

Now the above tree structure is exploited to reformulate the optimal control problem (4.8) as a finite-dimensional problem. The water demand, given by (4.3), is now modelled as

$$d^i_{k+j|k} = \hat{d}_{k+j|k} + \epsilon^i_j, \tag{4.9}$$

for all $j \in \mathbb{N}_{[0,N-1]}$ and $i \in \mathbb{N}_{[1,\mu(j+1)]}$. The input-disturbance coupling (4.8e) is then readily rewritten as

$$Eu^i_{k+j|k} + E_d d^i_{k+j|k} = 0, \tag{4.10}$$

for $j \in \mathbb{N}_{[0,N-1]}$ and $i \in \mathbb{N}_{[1,\mu(j+1)]}$.

The system dynamics is defined across the nodes of the tree by

$$x^l_{k+j+1|k} = Ax^i_{k+j|k} + Bu^l_{k+j|k} + G_d d^l_{k+j|k}, \tag{4.11}$$

for $j \in \mathbb{N}_{[0,N-1]}$, $i \in \mathbb{N}_{[1,\mu(j)]}$ and $l \in \text{child}(j,i)$, or, alternatively,

$$x^i_{k+j+1|k} = Ax^{\text{anc}(j+1,i)}_{k+j|k} + Bu^i_{k+j|k} + G_d d^i_{k+j|k}, \tag{4.12}$$

for $j \in \mathbb{N}_{[0,N-1]}$ and $i \in \mathbb{N}_{[1,\mu(j+1)]}$.

Now the expectation of the objective function (4.8b) can be derived as a summation across the tree nodes

$$\mathbb{E}V(\pi, p, q, k) = \sum_{j=0}^{N-1} \sum_{i=1}^{\mu(j)} p_j^i \ell(x_{k+j|k}^i, u_{k+j|k}^i, u_{k+j-1|k}^{\mathrm{anc}(j,i)}, k+j), \tag{4.13}$$

where $x_{k|k}^1 = p$ and $u_{k-1|k} = q$.

In order to guarantee the recursive feasibility of the control problem, the state constraints (4.8f) are converted into *soft constraints*, that is, they are replaced by a penalty of the form

$$\ell^d(x) = \gamma_d d(x, \mathcal{C}_1), \tag{4.14}$$

where γ_d is a positive penalty factor and $\mathcal{C}_1 = \{x \mid x_{\min} \leq x \leq x_{\max}\}$. Using this penalty, let's construct the *soft state constraint penalty*

$$V_s(\pi, p) = \sum_{j=0}^{N} \sum_{i=1}^{\mu(j)} \ell^d(x_{k+j|k}^i). \tag{4.15}$$

The modified, soft-constrained, SMPC problem can be now written as

$$\tilde{V}^\star(p, q, \hat{\mathbf{d}}, k) = \min_{\pi} \mathbb{E}V(\pi, p, q, k) + V_s(\pi, p), \tag{4.16a}$$

subject to

$$x_{k|k}^1 = p,\ u_{k-1|k} = q, \tag{4.16b}$$

$$u_{\min} \leq u_{k+j|k}^i \leq u_{\max}, j \in \mathbb{N}_{[0,N-1]}, i \in \mathbb{N}_{[1,\mu(j)]}, \tag{4.16c}$$

and system equations (4.10) and (4.12).

4.4 Solution of the stochastic optimal control problem

In this section the GPU-based proximal gradient method proposed in Chapter 3 is extended to solve the SMPC problem (4.16). For ease of notation let's denote $x_{j|0} = x_j$, $u_{j|0} = u_j$, $\hat{d}_{j|0} = \hat{d}_j$.

4.4.1 Proximal gradient algorithm

For a closed, proper extended-real valued function $g : \mathbb{R}^n \to \bar{\mathbb{R}}$, we define its *proximal operator* with parameter $\gamma > 0$, $\text{prox}_{\gamma g} : \mathbb{R}^n \to \mathbb{R}^n$ as (PB14a)

$$\text{prox}_{\gamma g}(v) = \underset{x \in \mathbb{R}^n}{\text{argmin}} \left\{ g(x) + \frac{1}{2\gamma} \|x - v\|_2^2 \right\}. \tag{4.17}$$

The proximal operator of many functions is available in closed form (CP10; PB14a). When g is given in a *separable sum* form, that is

$$g(x) = \sum_{i=1}^{\kappa} g_i(x_i), \tag{4.18a}$$

then for all $i \in \mathbb{N}_{[1,\kappa]}$

$$(\text{prox}_{\gamma g}(v))_i = \text{prox}_{\gamma g_i}(v_i). \tag{4.18b}$$

This is known as the *separable sum property* of the proximal operator.

Let $z \in \mathbb{R}^{n_z}$ be a vector encompassing all states x_j^i for $j \in \mathbb{N}_{[0,N]}$ and $i \in \mathbb{N}_{[1,\mu(j)]}$ and inputs u_j^i for $j \in \mathbb{N}_{[0,N-1]}$, $i \in \mathbb{N}_{[1,\mu(j+1)]}$; this is the decision variable of problem (4.16).

Let $f : \mathbb{R}^{n_z} \to \bar{\mathbb{R}}$ be defined as

$$\begin{aligned} f(z) = \sum_{j=0}^{N-1} \sum_{i=1}^{\mu(j)} & p_j^i (\ell^w(u_j^i) + \ell^\Delta(\Delta u_j^i)) + \delta(u_j^i | \Phi_1(d_j^i)) \\ & + \delta(x_{j+1}^i, u_j^i, x_j^{\text{anc}(j+1,i)} | \Phi_2(d_j^i)), \end{aligned} \tag{4.19}$$

where $\Delta u_j^i = u_j^i - u_{j-1}^{\text{anc}(j,i)}$ and $\Phi_1(d)$ is the affine subspace of $\mathbb{R}^{n_u}$ induced by (4.10), that is

$$\Phi_1(d) = \{u : Eu + E_d d = 0\}, \tag{4.20}$$

and $\Phi_2(d)$ is the affine subspace of $\mathbb{R}^{2n_x+n_u}$ defined by the system dynamics (4.12)

$$\Phi_2(d) = \{(x_{k+1}, x_k, u) : x_{k+1} = Ax_k + Bu + G_d d\}. \tag{4.21}$$

We define the auxiliary variables ς and ζ which stand for *copies* of the state variables x_j^i — that is $\varsigma_j^i = \zeta_j^i = x_j^i$ — and the auxiliary

variable ψ which is a copy of input variables $\psi_j^i = u_j^i$. The reason for the introduction of these variables will be clarified in Section 4.4.6.

We introduce the variable $t = (\varsigma, \zeta, \psi) \in \mathbb{R}^{n_t}$ and define an extended real valued function $g : \mathbb{R}^{n_t} \to \bar{\mathbb{R}}$ as

$$g(t) = \sum_{j=0}^{N-1} \sum_{i=1}^{\mu(j)} \ell^S(\varsigma_{j+1}^i) + \ell^d(\zeta_{j+1}^i) + \delta(\psi_j^i | \mathcal{U}), \tag{4.22}$$

where $\mathcal{U} = \{\psi \in \mathbb{R}^{n_u} : u_{\min} \leq \psi \leq u_{\max}\}$.

Now the finite-dimensional optimisation problem (4.16) can be written as:

$$\tilde{V}^\star = \min_{z,t} f(z) + g(t) \tag{4.23a}$$

$$\text{s.t. } Hz = t \tag{4.23b}$$

where

$$H = \begin{bmatrix} I_{n_x} & 0 \\ I_{n_x} & 0 \\ 0 & I_{n_u} \end{bmatrix}. \tag{4.24}$$

The Fenchel dual of (4.23) is written as (Roc72, Corol. 31.2.1):

$$\tilde{D}^\star = \min_{y} f^*(-H'y) + g^*(y), \tag{4.25}$$

where y is the dual variable. The dual variable y can be partitioned as $y = (\tilde{\varsigma}_j^i, \tilde{\zeta}_j^i, \tilde{\psi}_j^i)$, where $\tilde{\varsigma}_j^i$, $\tilde{\zeta}_j^i$ and $\tilde{\psi}_j^i$ are the dual variables corresponding to ς_j^i, ζ_j^i and ψ_j^i respectively. We also define the auxiliary variable of state copies $\tilde{\xi}_j^i := (\tilde{\varsigma}_j^i, \tilde{\zeta}_j^i)$.

According to (RW09, Thm. 11.42), since function $f(z) + g(Hz)$ is proper, convex and piecewise linear-quadratic, then the primal problem (4.23) is feasible whenever the dual problem (4.25) is feasible and strong duality holds, i.e., $\tilde{V}^\star = \tilde{D}^\star$. Moreover, the optimal solution of (4.23) is given by $z^\star = \nabla f^*(-H'y^\star)$ where $y^\star$ is *any* solution of (4.25). Applying (RW09, Prop. 12.60) to f^* and since f is lower semi-continuous, proper and σ-strongly convex — as shown at the end of Section 4.4.3 — its conjugate f^* has Lipschitz-continuous gradient with a constant $1/\sigma$.

An accelerated version of proximal-gradient method which was first proposed by Nesterov in (Nes83) is applied to the dual problem.

This leads to the following algorithm

$$w^{\nu} = y^{\nu} + \theta_{\nu}(\theta_{\nu-1}^{-1} - 1)(y^{\nu} - y^{\nu-1}), \tag{4.26a}$$

$$z^{\nu} = \underset{z}{\operatorname{argmin}}\{\langle z, H'w^{\nu}\rangle + f(z)\}, \tag{4.26b}$$

$$t^{\nu} = \operatorname{prox}_{\lambda^{-1}g}(\lambda^{-1}w^{\nu} + Hz^{\nu}), \tag{4.26c}$$

$$y^{\nu+1} = w^{\nu} + \lambda(Hz^{v} - t^{v}), \tag{4.26d}$$

$$\theta_{\nu+1} = \frac{1}{2}\left(\sqrt{\theta_{\nu}^4 + 4\theta_{\nu}^2} - \theta_{\nu}^2\right), \tag{4.26e}$$

starting from a dual-feasible vector $y^0 = y^{-1} = 0$ and $\theta_0 = \theta_{-1} = 1$.

In the first step (4.26a) we compute an extrapolation of the dual vector. In the second step (4.26b) we calculate the dual gradient, that is $z^{\nu} = \nabla f^*(-H'w^{\nu})$, at the extrapolated dual vector using the conjugate subgradient theorem (Roc72, Thm. 23.5). The third step comprises of (4.26c), (4.26d) where we update the dual vector y and in the final step of the algorithm we compute the scalar θ_{ν} which is used in the extrapolation step.

This algorithm has a convergence rate of $\mathcal{O}(1/\nu^2)$ for the dual iterates as well as for the ergodic primal iterate defined through the recursion $\bar{z}^{\nu} = (1 - \theta_{\nu})\bar{z}^{(\nu-1)} + \theta_{\nu}z^{\nu}$, i.e., a weighted average of the primal iterates (PB14c).

The splitting given in (4.23), with this particular choice of f and g, is not unique.

4.4.2 Computation of primal iterate

The most critical step in the algorithm is the computation of z^{ν} which accounts for most of the computation time required by each iteration. This step boils down to the solution of an unconstrained optimisation problem by means of dynamic programming where certain matrices (which are independent of w^{ν}) can be computed once before the algorithm to facilitate the online computations. This process result in three fold calculation:

- The demand forecast and the electricity prices in the cost function update at every time k. These are the vectors $\beta_j^i, \hat{u}_j^i, e_j^i$ which are associated with them are updated along with them. These are mentioned in the Section 4.4.3.

- Matrices that depend on the system dynamics (network topology) – Λ, Φ, Ψ and $\bar{B}$. Their calculation is referred as the *factor step* which is explained in the Section 4.4.4. It should be noted that these matrices are independent of the complexity of the scenario tree.
- Finally, the *solve step* that is executed at each iterate of the dual variable w^ν. It use the vectors updated at the time instance, $\beta_j^i, \hat{u}_j^i, e_j^i$, and the constant matrices of the factor step to generate the primal iterate as output. This is mentioned in the Section 4.4.5.

4.4.3 Elimination of input-disturbance coupling

This section discusses how the input-disturbance equality constraints can be eliminated by a proper change of input variables and compute the parameters $\beta_j^i, \hat{u}_j^i, e_j^i \; \forall i \in \mu(j), j \in \mathbb{N}_N$ which are then provided as input to Algorithm 6. These depend on the nominal demand forecasts $\hat{d}_{k+j|k}$ and on the time-varying economic cost parameters $\alpha_{2,k+j}$ for $j \in \mathbb{N}_{[0,N-1]}$, therefore, they need to be updated at every time instant k.

The affine space $\Phi_1(d)$ introduced in (4.20) can be written as

$$\Phi_1(d) = \{v \in \mathbb{R}^{n_v} : u = Lv + \hat{u}(d)\} \tag{4.27}$$

where $L \in \mathbb{R}^{n_u \times n_v}$ is a *full rank* matrix whose range spans the nullspace of E, i.e., for every $v \in \mathbb{R}^{n_v}$, we have Lv is in the kernel of E and $\hat{u}(d)$ satisfies $E\hat{u}(d) + E_d d = 0$.

Substituting $u_j^i = Lv_j^i + \hat{u}_j^i, \forall i \in \mu(j), j \in \mathbb{N}_N$ in the dynamics $\Phi_2(d)$ in (4.21) gives

$$\begin{aligned}\Phi_2(d) =& \{(x_{j+1}, x_j, v) : x_{j+1} = Ax_j + \bar{B}v + e, \\ & \bar{B} = BL, e = B\hat{u} + G_d d\},\end{aligned} \tag{4.28}$$

and we define

$$e_j^i = B\hat{u}_j^i + G_d d_j^i. \tag{4.29}$$

Now the cost in (4.19) is transformed as:

$$\sum_{j=0}^{N-1} \sum_{i=1}^{\mu(j)} p_j^i(\ell^w(u_j^i) + \ell^\Delta(\Delta u_j^i)) = \sum_{j=0}^{N-1} \sum_{i=1}^{\mu(j)} p_j^i(\ell^w(v_j^i) + \ell^\Delta(\Delta v_j^i, \hat{u}_j^i)) \tag{4.30}$$

where

$$\hat{R} = W_u L, \tag{4.31a}$$
$$\bar{R} = L' \hat{R}, \tag{4.31b}$$
$$\bar{\alpha}_j = W_\alpha(\alpha_1 + \alpha_{2,j+k})L \tag{4.31c}$$
$$\ell^w(v_j^i) = \bar{\alpha}_j' v_j^i, \tag{4.31d}$$
$$\Delta v_j^i = v_j^i - v_{j-1}^{\mathrm{anc}(j,i)}, \tag{4.31e}$$
$$\Delta \hat{u}_j^i = \hat{u}_j^i - \hat{u}_{j-1}^{\mathrm{anc}(j,i)}, \tag{4.31f}$$
$$\ell^\Delta(\Delta v_j^i, \Delta \hat{u}_j^i) = \Delta v_j^i \bar{R} \Delta v_j^i + 2\Delta \hat{u}_j^{i\prime} \hat{R} \Delta v_j^i, \tag{4.31g}$$

By substituting and expanding Δv_j^i and $\Delta \hat{u}_j^i$ in $\ell^\Delta(\Delta v_j^i, \Delta \hat{u}_j^i)$ the cost in (4.31g) becomes

$$\sum_{j=0}^{N-1} \sum_{i=1}^{\mu(j)} p_j^i (\ell^w(u_j^i) + \ell^\Delta(\Delta u_j^i)) =$$
$$\sum_{j=0}^{N-1} \sum_{i=1}^{\mu(j)} \bar{p}_j^i v_j^{i\prime} \bar{R} v_j^i - 2p_j^i v_{j-1}^{\mathrm{anc}(j,i)\prime} \bar{R} v_j^i + \beta_j^{i\prime} v_j^i \tag{4.32}$$

where

$$\bar{p}_j^i = p_j^i + \sum_{l \in \mathrm{child}(j,i)} p_{j+1}^l \tag{4.33a}$$
$$\beta_j^i = p_j^i \bar{\alpha}_j + 2p_j^i \hat{R}\Big(\bar{p}_j^i \hat{u}_j^i - \hat{u}_{j-1}^{\mathrm{anc}(j,i)} - \sum_{l \in \mathrm{child}(j,i)} p_{j+1}^l \hat{u}_{j+1}^l\Big) \tag{4.33b}$$

Now $\hat{u}_j^i$, e_j^i, β_j^i are calculated from (4.27), (4.29) and (4.33b) respectively. Using our assumption that L is full-rank, we can see that $\bar{R}$ is a positive definite and symmetric matrix, therefore, f is strongly convex.

4.4.4 Factor step

Algorithm 6 solves the unconstrained minimisation problem (4.26b), that is

$$z^\star = \underset{z}{\mathrm{argmin}}\{\langle z, H'y \rangle + f(z)\} \tag{4.34}$$

where $z = \{x_j^i, u_j^i\}$, $y = \{\tilde{\varsigma}_j^i, \tilde{\zeta}_j^i, \tilde{\psi}_j^i\}$ for $i \in \mathbb{N}_{[1,\mu(j)]}$ and $j \in \mathbb{N}_{[0,N]}$, $f(z)$ is given by (4.19) and H is given by (4.24). Substituting H the optimisation problem becomes

$$\begin{aligned} z^\star = \underset{z}{\operatorname{argmin}} \sum_{j=0}^{N-1} \sum_{i=1}^{\mu(j)} & p_j^i(\ell^w(u_j^i) + \ell^\Delta(\Delta u_j^i)) \\ & + \tilde{\xi}_j^{i\prime} x_j^i + \tilde{\psi}_j^{i\prime} u_j^i + \delta(u_j^i | \Phi_1(d_j^i)) \\ & + \delta(x_{j+1}^i, u_j^i, x_j^{\mathrm{anc}(j+1,i)} | \Phi_2(d_j^i)) \end{aligned} \tag{4.35}$$

where $\tilde{\xi}_j^i := (\tilde{\varsigma}_j^i, \tilde{\zeta}_j^i)$.

The input-disturbance coupling constraints imposed by $\delta(u_j^i | \Phi_1(d_j^i))$ in the above problem are eliminated as discussed in Section 4.4.3. This changes the input variable from u_j^i to v_j^i given by (4.27) and the cost function as in (4.32). We, therefore, replace the decision variable z with $\bar{z} := \{x_j^i, v_j^i\}$ and the optimisation problem (4.35) reduces to

$$\begin{aligned} \bar{z}^\star = \underset{\bar{z}}{\operatorname{argmin}} \sum_{j=0}^{N-1} \sum_{i=1}^{\mu(j)} & \bar{p}_j^i v_j^{i\prime} \bar{R} v_j^i - 2 p_j^i v_{j-1}^{\mathrm{anc}(j,i)\prime} \bar{R} v_j^i + \beta_j^{i\prime} v_j^i + \tilde{\xi}_j^{i\prime} x_{j+1}^j \\ & + \tilde{\psi}_j^{i\prime} L v_j^i + \delta(x_{j+1}^i, v_j^i, x_j^{\mathrm{anc}(k,i)} | \Phi_2(d_j^i)), \end{aligned} \tag{4.36}$$

where $u_j^i = L v_j^i + \hat{u}_j^i$.

The above problem is an unconstrained optimisation problem with quadratic stage cost which is solved using dynamic programming (Ber00). This method transforms the complex problem into a sequence of sub-problems solved at each stage.

Using dynamic programming we find that the transformed control actions $v_j^{i\star}$ have to satisfy

$$v_j^{i\star} = v_{j-1}^{\mathrm{anc}(j,i)} + \frac{1}{2p_j^i} \big(\Phi(\tilde{\xi}_j^i + q_{j+1}^i) + \Psi \tilde{\psi}_j^i + \Lambda(\beta_j^i + r_{j+1}^i) \big) \tag{4.37}$$

where

$$\Lambda = -\bar{R}^{-1}, \tag{4.38a}$$

$$\Phi = \Lambda \bar{B}', \tag{4.38b}$$

$$\Psi = \Lambda L. \tag{4.38c}$$

Matrix $\bar{R}$ is symmetric and positive definite, therefore, this can compute once its Cholesky factorisation so that the inverse computation is obviated.

The q_{j+1}^i, r_{j+1}^i in (4.37) correspond to the linear cost terms in the cost-to-go function at node i of stage $j+1$. At stage j, these terms are updated by substituting the $v_j^{i\star}$ as:

$$r_j^s = \sum_{l \in \text{child}(j-1,s)} \sigma_j^l + \bar{B}'(\tilde{\xi}_j^l + q_{j+1}^l) + L\tilde{\psi}_j^l, \tag{4.39a}$$

$$q_j^s = A' \sum_{l \in \text{child}(j-1,s)} \tilde{\xi}_j^l + q_{j+1}^l \tag{4.39b}$$

where $s = \text{anc}(j, i)$.

Equations (4.37) and (4.39) form the solve step as in Algorithm 6. Matrices Λ, Φ and Ψ are required to be computed once.

4.4.5 Solve step

The computation of z^ν at each iteration of the algorithm requires the computation of the aforementioned matrices and is computed using Algorithm 6 to which is referred as the *solve step*. Computations involved in the solve step are merely matrix-vector multiplications. As the algorithm traverses the nodes of the scenario tree stage-wise backwards (from stage $N-1$ to stage 0), computations across the nodes at a given stage can be performed in parallel. Hardware such as GPUs which enable us to parallelisable such operations lead to a great speed-up as demonstrated in Section 4.5.

4.4.6 Computation of dual iterate

Function g given in (4.22) is given in the form of a separable sum

$$g(t) = g(\varsigma, \zeta, \psi) = g_1(\varsigma) + g_2(\zeta) + g_3(\psi), \tag{4.40}$$

Algorithm 6 Solve step (DWN)

Require: Output of the factor step (See Sections 4.4.3 and 4.4.4), i.e., $\Lambda, \Phi, \Psi, \bar{B}, \hat{u}_j^i, \beta_j^i, e_j^i$, p, q and $w^\nu = (\tilde{\varsigma}_j^i, \tilde{\zeta}_j^i, \tilde{\psi}_j^i)$.

$q_N^i \leftarrow 0$, and $r_N^i \leftarrow 0, \forall i \in \mathbb{N}_{[1,n_s]}$,
for $j = N-1, \ldots, 0$ **do**
  for $i = 1, \ldots, \mu(k)$ **do** {in parallel}
    $\sigma_j^l \leftarrow r_{j+1}^l + \beta_j^l, \forall l \in \text{child}(j,i)$
    $v_j^l \leftarrow \frac{1}{2p_j^l}\left(\Phi_j^l(\tilde{\xi}_j^l + q_{j+1}^l) + \Psi_j^l \tilde{\psi}_j^l + \Lambda_j^l \sigma_j^l\right)$,
      $\forall l \in \text{child}(j,i)$
    $r_j^i \leftarrow \sum_{l \in \text{child}(j,i)} \sigma_j^l + \bar{B}'(\tilde{\xi}_j^l + q_{j+1}^l) + L\tilde{\psi}_j^l$
    $q_j^i \leftarrow A' \sum_{l \in \text{child}(j,i)} \tilde{\xi}_j^l + q_{j+1}^l$
  end for
end for
$x_0^1 \leftarrow p$, $u_{-1} \leftarrow q$,
for $j = 0, \ldots, N-1$ **do**
  for $i = 1, \ldots, \mu(k)$ **do** {in parallel}
    $v_j^i \leftarrow v_{j-1}^{\text{anc}(j,i)} + v_j^i$
    $u_j^i \leftarrow Lv_j^i + \hat{u}_j^i$
    $x_{j+1}^i \leftarrow Ax_j^{\text{anc}(j,i)} + \bar{B}v_j^i + e_j^i$
  end for
end for
return $\{x_j^i\}_{j=1}^N$, $\{u_j^i\}_{j=0}^{N-1}$

where

$$g_1(\varsigma) = \sum_{j=0}^{N-1} \sum_{i=1}^{\mu(j)} \ell^S(\varsigma_{j+1}^i), \tag{4.41a}$$

$$g_2(\zeta) = \sum_{j=0}^{N-1} \sum_{i=1}^{\mu(j)} \ell^d(\zeta_{j+1}^i), \tag{4.41b}$$

$$g_3(\psi) = \sum_{j=0}^{N-1} \sum_{i=1}^{\mu(j)} \delta(\psi_j^i \mid \mathcal{U}). \tag{4.41c}$$

Functions $g_1(\cdot)$ and $g_2(\cdot)$ are in turn separable sums of distance functions from a set and $g_3(\cdot)$ is an indicator function. Their proximal mappings can be easily computed below and essentially are element-wise operations on the vector t that can be fully parallelised.

Proximal operators

Function g in (4.40) is a separable sum of distance and indicator functions and its proximal is computed according to (4.18). The proximal operator of the indicator of a convex closed set C, that is

$$\delta_C(x) = \begin{cases} 0, & \text{if } x \in \mathcal{C} \\ +\infty, & \text{otherwise} \end{cases}$$

is the projection operator onto C, i.e.,

$$\text{prox}_{\lambda\delta_C}(v) = \text{proj}_{\mathcal{C}}(v) = \underset{y \in C}{\text{argmin}} \, \|v - y\|, \tag{4.42}$$

When g is the distance function from a convex closed set C, that is

$$\begin{aligned} g(x) =& \mu d(x \mid \mathcal{C}) = \inf_{y \in \mathcal{C}} \mu \|x - y\|_2 \\ =& \mu \|x - \text{proj}_{\mathcal{C}}(x)\|_2 \end{aligned}$$

Then proximal operator of g given by (CP10)

$$\text{prox}_{\lambda g}(v) = \begin{cases} x + \frac{\text{proj}_{\mathcal{C}}(v) - v}{d(v|\mathcal{C})}, & \text{if } d(v \mid \mathcal{C}) > \lambda\mu \\ \text{proj}_{\mathcal{C}}(v), & \text{otherwise} \end{cases}$$

4.4.7 Preconditioning, Choice of λ & Termination

Preconditioning: First-order methods are known to be sensitive to scaling and preconditioning can remarkably improve their convergence rate. Various preconditioning method such as (GB15; Bra10) have been proposed in the literature. Additionally, a parallelizable preconditioning method tailored to stochastic programs for use with interior point solvers has been proposed in (CLZ16). Here, we employ a simple diagonal preconditioning which consists in computing a diagonal matrix $\tilde{H}_D$ with positive diagonal entries which approximates the dual

Hessian H_D and use $\tilde{H}_D^{-1/2}$ to scale the dual vector (Ber99, Sec. 2.3.1). Since the uncertainty does not affect the dual Hessian, the preconditioning matrix for a single branch of the scenario tree and use it to scale all dual variables.

Choice of λ : In a similar way, the parameter λ is computed. It is chosen as $\lambda = 1/L_{H_D}$ where L_{H_D} is the Lipschitz constant of the dual gradient which is computed as $\|H\|^2/\sigma$ as in (Ber99). It again suffices to perform the computation for a single branch of the scenario tree. In order to avoid future problems a backtracking step with initial guess of λ is also included.

Termination: The termination conditions for the above algorithm are based on the ones provided in (PB14c). However, rather than checking these conditions at every iteration, we perform always a fixed number of iterations which is dictated by the sampling time. The quality of the solution is checked *a posteriori* in terms of the duality gap in relative measure and the term $\|Hz^\nu - t^\nu\|_\infty$.

4.5 Case study: The Barcelona DWN

The proposed control methodology is applied to the drinking water network of the city of Barcelona discussed in the Chapter 2 which is based on the data found in (GOMPJ14; SGS$^+$14). The topology of the network is presented in Figure 19. The system model consists of 63 states corresponding to the level of water in each tank, 114 control inputs which are pumping actions and valve positions, 88 demand nodes and 17 junctions. The prediction horizon is $N = 24$ with sampling time of 1 hour. The future demands are predicted using the SVM time series model developed in Section 2.3.

4.5.1 Performance of GPU-accelerated algorithm

Accelerated proximal gradient (APG) was implemented in CUDA-C v6.0 and the matrix-vector computations were performed using cuBLAS. The GPU-based implementation is compared against the interior-point solver of Gurobi[3]. Active-set algorithms exhibited very poor perfor-

[3]Gurobi does not support computations on GPUs and as we read in http://www.gurobi.com/pdfs/webinar-parallel-and-distributed-optimisa-

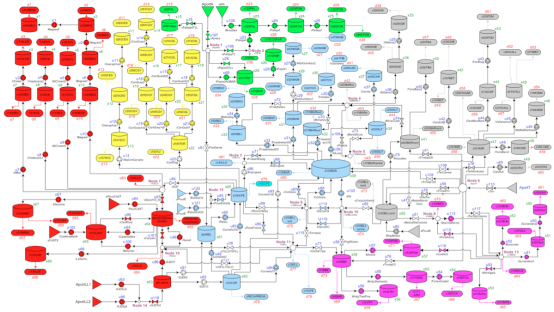

Figure 19: Structure of the DWN of Barcelona.

mance and did not include the respective results.

All computations on CPU were performed on a $4 \times 2.60\text{GHz}$ Intel i5 machine with 8GB of RAM running 64-bit Ubuntu v14.04 and GPU-based computations were carried out on a NVIDIA Tesla C2075.

The dependence of the computation time on the size of the scenario tree is reported in the Figure 20 where it can be noticed that APG running on GPU compared to Gurobi is 10 to 25 times faster. Furthermore, the speed-up increases with the number of scenarios as we may notice by looking at the insert 20.

The optimisation problems we are solving here are of noticeably large size. Indicatively, the scenario tree with 493 scenarios counts approximately 2.52 million dual decision variables (1.86 million primal variables) and while Gurobi requires $1329s$ to solve it, our CUDA implementation solves it in $58.8s$; this corresponds to a speed-up of $22.6\times$.

In all the simulations the obtained sequence of control actions across the tree nodes $U^{\star}_{apg} = \{u^i_j\}$ which was, element-wise, within $\pm 0.029 m^3/s$ (1.9%) from the solution produced by Gurobi. The max-

tion-english.pdf, the current status of GPU technology does not allow the implementation of a GPU-enabled version of Gurobi. Although there have been proposed interior point methods for the solution of stochastic optimal control problems (HOS01; DZZ+12; BL02), a clear advantage of APG is that it can fully exploit the structure of the problem while it generally allows for convex non-quadratic costs and soft constraints.

imum primal residual was $\|Hx - z\|_\infty = 1.7446$. Moreover, it should be noted that the control action $u_0^\star$ computed by APG with 500 iterations was consistently within $\pm 0.0025 m^3/s$ (0.08%) from the Gurobi solution. Given that only $u_0^\star$ is applied to the system while all other control actions u_j^i for $j \in \mathbb{N}_{[1,N-1]}$ and $i \in \mathbb{N}_{[1,\mu(j)]}$ are discarded, 500 iterations are well sufficient for convergence.

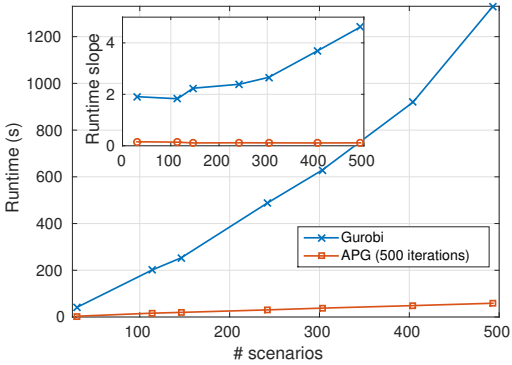

Figure 20: Runtime of the CUDA implementation against the number of scenarios considered in the optimisation problem. Comparison with the runtimes of Gurobi.

4.5.2 Closed-loop performance

This section analysis the closed-loop performance of SMPC with different scenario-trees. This analysis is carried for a period of 7 days ($H_s = 168$) from 1st to 8th July 2007. The demand data were provided by AGBAR (Aguas de Barcelona, s.a.), which is the company that manages the Barcelona DWN). Here, the operational cost and the quality of service of various scenario-tree structures were compared.

The weighting matrices in the operational cost are chosen as $W_\alpha = 2 \cdot 10^4$, $W_u = 10^5 \cdot I$ and $W_x = 10^7$, respectively and $\gamma_d = 5 \cdot 10^7$. The demand is predicted using SVM model presented in (SGS+14). The steps involved in SMPC using GPU based APG in closed-loop is

summarised in Algorithm 7.

Algorithm 7 SMPC control of DWN

Require: Scenario-tree, current state measurement x_0 and previous control u_{-1}.
Compute Λ, Φ, Ψ and $\bar{B}$ as in Section 4.4.4
Precondition the original optimisation problem and compute λ as in Section 4.4.7.
loop
Step 1. Predict the future water demands $\hat{\mathbf{d}}_k$ using current and past demand data.
Step 2. Compute $\hat{u}_j^i$, β_j^i, e_j^i as in Section 4.4.3.
Step 3. Solve the optimisation problem using APG on GPU using iteration (4.26) and Algorithm 6.
Step 4. Apply u_0^1 to the system, update $u_{-1} = u_0^1$
end loop

For the performance assessment of the proposed control methodology we used various controllers summarised in Table 2. The corresponding computational times are presented in Figure 20.

To assess the performance of closed-loop operation of the SMPC-controlled network we used the *key performance indicators* (KPIs) reported in (ABCJC06; GMOMP14). For a simulation time length H_s the performance indicators are computed by

$$\mathrm{KPI}_E = \frac{1}{H_s}\sum_{k=1}^{H_s}(\alpha_1 + \alpha_{2,k})'|u_k|, \tag{4.43a}$$

$$\mathrm{KPI}_{\Delta U} = \frac{1}{H_s}\sum_{k=1}^{H_s}\|\Delta u_k\|^2, \tag{4.43b}$$

$$\mathrm{KPI}_S = \sum_{k=1}^{H_s}\|[x_s - x_k]_+\|_1 \tag{4.43c}$$

$$\mathrm{KPI}_R = \frac{\|x_s\|_1}{\frac{1}{H_s}\sum_{k=1}^{H_s}\|x_k\|_1} \times 100\%. \tag{4.43d}$$

Controller	b_k	scenarios	primal variables	dual variables
CE-MPC	1	1	4248	5760
SMPC_1	$[3, 2]$	6	24072	32540
SMPC_2	$[6, 5]$	30	118059	160080
SMPC_3	$[6, 5, 5]$	114	430287	583440
SMPC_4	$[8, 5, 5]$	146	551355	747600
SMPC_5	$[10, 8, 5]$	242	915621	1241520
SMPC_6	$[12, 8, 5]$	303	1145544	1553280
SMPC_7	$[12, 8, 8]$	404	1520961	2062320
SMPC_8	$[12, 10, 8]$	493	1856022	2516640

Table 2: Various controllers used to assess the closed-loop performance of the proposed methodology. The numbers in the bracket denote the first *maximum branching factors*, b_j, of the scenario tree while all subsequent branching factors are assumed to be equal to 1.

KPI_E is the average economic cost, $\text{KPI}_{\Delta U}$ measures the average smoothness of the control actions, KPI_S corresponds to the total amount of water used from storage and KPI_R is the percentage of the safety volume x_s contained into the average volume of water.

Risk vs Economic utility

Figure 21 illustrates the trade-off between economic and safe operation: The more scenarios we use to describe the distribution of demand prediction error, the safer the closed-loop operation becomes as it is reflected by the decrease of KPI_S. Stochastic MPC leads to a significant decrease of economic cost compared to the certainty-equivalence approach, however, the safer we require the operation to be, the higher the operating cost we should expect. For example, if the designer opts for 30 scenarios, they will have struck a low operating cost which, nevertheless, comes with a high value of KPI_S, that is, operation under high risk. In order to decrease this risk, one needs to consider a higher number of scenarios which comes at a higher operating cost. We may also observe that for too few scenarios, the operation of the network will be both expensive and will incur a rather high risk.

Controller	KPI_E	$\mathrm{KPI}_{\Delta U}$	KPI_S	KPI_R
CE-MPC	$\mathbf{1801.4}$	$\mathbf{0.2737}$	$\mathbf{6507.7}$	64.89%
SMPC_1	1633.5	0.3896	1753.7	$\mathbf{67.96}\%$
SMPC_2	$\mathbf{1549.7}$	0.4652	2264.0	61.81%
SMPC_3	1574.0	0.4135	1360.0	49.65%
SMPC_4	1583.2	0.4088	885.7	48.13%
SMPC_5	1597.3	0.4470	508.5	46.05%
SMPC_6	1606.3	$\mathbf{0.4878}$	$\mathbf{302.3}$	$\mathbf{44.93}\%$

Table 3: KPIs for performance analysis of the DWN with different controllers. The lowest and the highest in each of the indicator is highlighted. The economical benefit and risk is presented with terms of number of scenarios

Quality of service

A measure of the reliability and quality-of-service of the network is KPI_S which reflects the tendency of water levels to drop under the safety storage levels. As expected, the CE-MPC controller leads to the most unsafe operation, whereas SMPC_6 leads to the lowest value.

Network utility

Network utility is defined as the ability to utilise the water in the tanks to meet the demands rather than pumping additional water and is quantified by KPI_R. In Table 3, we see the dependence of KPI_R on the number of scenarios of the tree. KPI_R remains always within reasonable limits; on average we operate away from the safety storage limit. The decrease in KPI_R one may observe is because the more scenarios are introduced, the more accurate the representation of uncertainty becomes and the system does not need to operate, on average, too far away from x_s.

Smooth operation

We may notice that the introduction of more scenarios results in an increase in $\mathrm{KPI}_{\Delta U}$. Then, the controller becomes more responsive to

accommodate the need for a less risky operation, although the value of $\mathrm{KPI}_{\Delta U}$ is not greatly affected by number of scenarios.

In Figure 22 we show two pumping actions during 168h (1 week) of operation. We may observe the tendency of the controller to be parsimonious with pumping when the corresponding pumping cost is high.

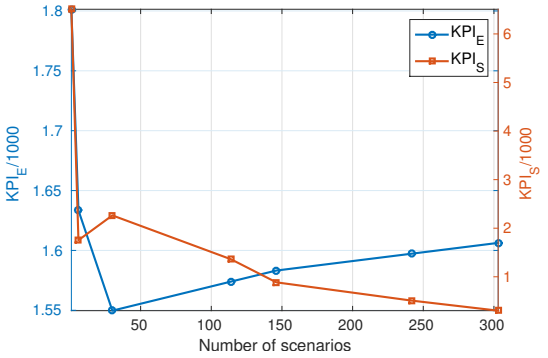

Figure 21: The figure shows the trade-off between risk and economic utility in terms of scenarios. The KPI_E represent the economical utility and KPI_S shows the risk of violation.

4.5.3 Implementation details

At every time instant k, state measurement and a sequence of demand predictions were needed to be loaded onto the GPU(see Figure 17), that is $\hat{\mathbf{d}}_k$. This amounts to $8.4kB$ and is rapidly uploaded on the GPU (less than $0.034ms$). In case the scenario-tree values were need to be updated,i.e.,$\boldsymbol{\epsilon}_k$, and for the case of SMPC_8 it is need to upload $3.52MB$ which is done in $3.74ms$. Therefore, the time needed to load these data on the GPU is not a limiting factor.

The parallelisation of matrix-vector multiplications required in Algorithm 6 are implemented using method `cublasSgemmBatched` of cuBLAS. Vector additions are performed using `cublasSaxpy` and summations over the set of children of a node were done using a custom

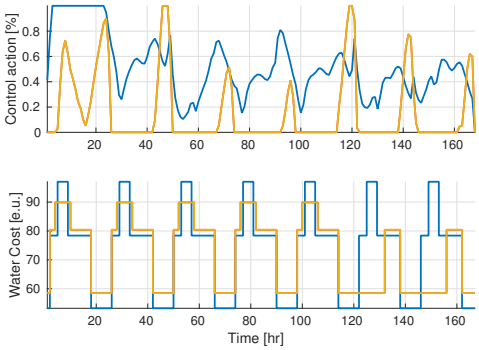

Figure 22: Pumping actions using SMPC_4 (expressed in $\%$ of $u_{\max}$) and the corresponding weighted time-varying cost $W_\alpha \alpha_{2,k}$ in economic units.

kernel.

4.6 Conclusions

This chapter have presented a framework for the formulation of a stochastic model predictive control problem for the operational management of drinking water networks. It have proposed a novel approach for the efficient numerical solution of the associated optimisation problem on a GPU. The computational feasibility of this algorithm and the benefits for the operational management of the system in terms of performance quantified using certain KPIs from the literature is demonstrated.

The SMPC controller provides set-points for the controller which are realised by local controllers. The topology of the drinking-water network prohibit for a centralised controller. In the next chapter, we address this by spatial decomposition of the control problem with local decentralised controllers.

Chapter 5

Decentralised control of large-scale drinking water networks

In the previous chapters, we investigated the economic stochastic model predictive control formulation and efficient algorithm for its online solution. This chapter address the problem of the spatial decomposition of the control problem under realistic conditions: (i) hard state-input constraints, and (ii) different time scales. Indeed, the tank level references are instructed by the MPC module based on economical criteria once every hour, while at a lower control layer control actions are commanded by local controllers (such as PID, or LQR) at a higher rate which can be at the order of magnitude of 1Hz or higher. Economic stochastic MPC is not guaranteed to satisfy the state constraints in closed loop; this is the motivation for the work presented in this chapter and these results are published in (SSB16).

5.1 Introduction

5.1.1 Motivation and background

Large-scale systems (such as drinking water networks and power distribution networks) call for control strategies based on the spatial and temporal decomposition of the overall dynamics so as to leverage the high computational cost of a centralised control approach (FBB$^+$80). In large scale systems hierarchical control is often the basis for a decentralised control scheme (CHL10; IY06) and various decentralised and hierarchical control schemes have been proposed in the literature for which Scattolini (Sca09) provides a thorough review. An overview of the current architectural trends in decentralised control for large-scale interconnected systems is provided by Bakule (Bak08).

Drinking Water Networks (DWNs) are large-scale systems whose operation is liable to set of operating, safety and quality-of-service constraints. The optimal management of DWNs is a complex task with outstanding socio-economic and environmental implications and has received considerable attention by the scientific community (OMPB11; OMBPB12b). One key reason for the use of decentralised control schemes is the need to isolate certain parts of the network for maintenance purposes without the need to re-model the overall system.

In the previous Chapters (2 and 4) we proposed a control framework for large-scale DWNs where pumping actions are computed by minimising a cost index. Such approaches are in the spirit of economic MPC (AAR12), and, despite the fact that are proven to lead to improved closed-loop behaviour, may fail to guarantee the satisfaction of state constraints in closed loop. The proposed methodology allows the operator to command reference signals to the sub-systems of the network according to some cost-optimisation strategy in such a way so as to satisfy the constraints during controlled operation.

Various decentralised and hierarchical control schemes have been proposed in the literature for which Scattolini (Sca09) provides a thorough review. The use of reference governors has been recommended by various authors so as to mitigate the computational burden of a centralised approach by separating the constraint satisfaction problem from the stabilisation problem (BCM97; GK99; CMP04; CGT14; RFT12). Recently, Kalabić and Kolmanovsky (KK13) proposed a methodology for the design of reference governors for constrained large-scale

linear systems. Two-layer hierarchical control systems are considered in the majority of relevant publications (see (PVSC10) and references therein). Recently, Vermillion *et al.* (VMK13) proposed a hierarchical architecture where both the upper and the lower layer implement MPC-based controllers with contractive terminal sets and they provide stabilising conditions but without considering state constraints. Magni and Scattolini (MS06) also make use of contractive sets which they impose as terminal sets in the formulation of the MPC problem for the decentralised control of nonlinear systems. Vaccarini *et al.* (VLK09) proposed a decentralised MPC scheme for unconstrained systems. Also, very recently, Riverso *et al.* (RFFT13) employed tube-based MPC to design a plug-and-play decentralised control scheme for linear constrained systems while Betti *et al.* (BFS13) also employ a tube-based method for the tracking of constant reference signals and model the interactions between subsystems of the partitioned system as disturbances. An overview of the current architectural trends in decentralised control for large-scale interconnected systems is provided by Bakule (Bak08).

Multirate control schemes are quite popular as they increase the flexibility in the quest for the desired properties (stability, optimality, constraints satisfaction) (SS94; PVSC10; HLndlPn$^+$11). A multi-rate control approach is adopted in this chapter with a quantification of the effect that the ratio of the two sampling rates has on the control of the system. We will show that the adoption of different reference rates in the upper and the lower control layers offers great flexibility and enables us to strike a balance between responsiveness to set-point changes and optimality.

In this chapter we propose a hierarchical multi-rate decentralised control scheme for the control of large-scale systems whose states and inputs are subject to linear constraints. The hierarchical scheme comprises two control layers: At the lower one, a linear controller stabilises the open-loop process without considering the constraints. A higher-level controller commands reference signals at a lower uniform sampling frequency so as to enforce linear constraints on the process variables. We propose a methodology for large-scale dynamically coupled linear systems which are partitioned into interconnected subsystems with state and input constraints. Worst-case interactions between subsystems are modeled and accounted for in a robust man-

ner. By optimally constraining the magnitude and rate of variation of the reference signals to each lower-level controller, quantitative criteria are provided for selecting the ratio between the sampling rates of the upper and lower layers of control at each location, in a way that closed-loop stability is preserved and the fulfilment of the prescribed constraints is guaranteed. This chapter builds on previous work by Barcelli *et al.* (BBR10; BB09) and on the ideas presented in (ABB11).

5.2 Multirate decentralised hierarchical control

5.2.1 Notation

Let $\mathbb{R}, \mathbb{R}^n, \mathbb{R}^{n\times m}, \mathbb{N}, \mathbb{N}_{[k1,k2]}, \mathbb{S}^n_+, \mathbb{S}^n_{++}$ denote the sets of real numbers, the n-dimensional vectors, the n-by-m real matrices, the set of natural numbers, the natural numbers in the interval $[k_1, k_2]$, the set of symmetric positive semi-definite and the set of positive definite n-by-n matrices respectively. The infinity-norm of $x \in \mathbb{R}^n$ is defined as $\|x\|_\infty := \max_{i\in\mathbb{N}_{[1,n]}} |x_i|$.

Let $A \in \mathbb{R}^{n\times m}$, $\mathcal{I} \subseteq \mathbb{N}_{[1,n]}$ and $\mathcal{J} \subseteq \mathbb{N}_{[1,m]}$; we denote by $A_{\mathcal{I}\mathcal{J}} \in \mathbb{R}^{|\mathcal{I}|\times|\mathcal{J}|}$ the submatrix of A formed by the rows and columns of A whose indices are in $\mathcal{I}$ and $\mathcal{J}$ respectively and $|\mathcal{I}|$ stands for the cardinality of the set $\mathcal{I}$. For a vector $x \in \mathbb{R}^n$, $x_\mathcal{I}$ denotes the vector of $\mathbb{R}^{|\mathcal{I}|}$ formed by the elements of x whose indices are in $\mathcal{I}$. We denote by $(A)_i$ the i-th row of A, while $(x)_i$ denotes the i-th element of x. Finally, we denote by 1_n the n-vector having all entries equal to 1.

5.2.2 Problem formulation

The proposed setting comprises two control layers: the lower control layer (LCL) and the upper control layer (UCL) which operate at different sampling frequencies. The lower control layer comprises m independent controllers whose role is the stabilisation of the open-loop dynamics of the controlled system without taking into account the prescribed state and input constraints. The lower layer controllers operate at a higher sampling frequency, namely $1/T_L$, and receive reference signals from corresponding upper layer controllers which

operate at lower sampling frequencies $1/T_H^{\langle i \rangle}$, $i \in \mathbb{N}_{[1,m]}$. We define $N^{\langle i \rangle} := T_H^{\langle i \rangle}/T_L$ to be the ratio between sampling frequencies of UCL and LCL which are positive integers refereed to as *reference rates*. To simplify the notation, the state variable of the system (involving all sub-systems) at the LCL sampling instants is denoted by x_k for $k \in \mathbb{N}$ (referring to all sub-systems) and the state at the UCL sampling instants is denote by $x^\nu := x_{\nu N}$ for $\nu \in \mathbb{N}$.

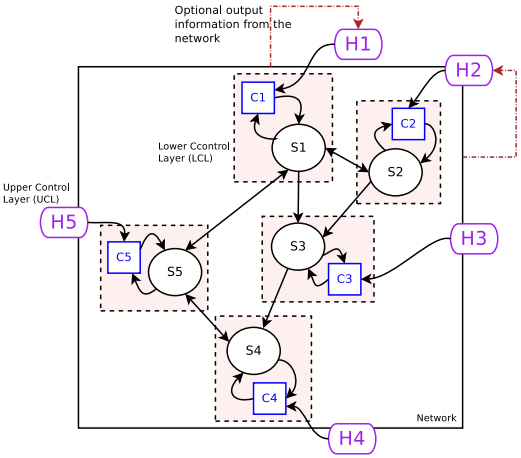

Figure 23: Two-layer (LCL and UCL) decentralised hierarchical control scheme over a network of interconnected, dynamically coupled components.

Let x_k, u_k, y_k respectively be the state, the input and the output of the lower layer process in discrete time and the dynamics of the system be given by:

$$x_{k+1} = \bar{A}x_k + \bar{B}u_k, \tag{5.1a}$$

$$y_k = \bar{C}x_k + \bar{D}u_k, \tag{5.1b}$$

where $x_k \in \mathbb{R}^{n_x}$, $y_k \in \mathbb{R}^{n_r}$, $u_k \in \mathbb{R}^{n_u}$ and $\bar{A}$, $\bar{B}$, $\bar{C}$ and $\bar{D}$ are known matrices of proper dimensions.

The feedback law defining the LCL is:

$$u_k = Fx_k + Er_k, \tag{5.2}$$

where $r_k \in \mathbb{R}^{n_r}$ stands as a reference signal to be decided by the Upper Layer Controller (ULC).

The reference-to-output gain $\Theta \in \mathbb{R}^{n_r \times n_r}$ of (5.1) under feedback control law (5.2), is:

$$\Theta := ((\bar{C} + \bar{D}F)(I - \bar{A} - \bar{B}F)^{-1}\bar{B} + \bar{D})E. \tag{5.3}$$

The closed-loop system (5.1) can be rewritten as

$$x_{k+1} = Ax_k + Br_k, \tag{5.4a}$$
$$y_k = Cx_k + Dr_k, \tag{5.4b}$$

where $A := \bar{A} + \bar{B}F$, $B := \bar{B}E$, $C := \bar{C} + \bar{D}F$ and $D := \bar{D}E$. Additionally, matrix E must be chosen so that (A, B) is a controllable pair.

The sparsity pattern of $\bar{A}$ in (5.1) can be exploited so as to decompose (5.1) into m subsystems which are as decoupled as possible; the components of the state vector are rearranged so that $\bar{A}$ in the new coordinates is as close as possible to a block-diagonal form. Let $\mathcal{I}_x^{\langle i \rangle}$, $\mathcal{I}_u^{\langle i \rangle}$ and $\mathcal{I}_r^{\langle i \rangle}$ ($i \in \mathbb{N}_{[1,m]}$) denote the sets of state, input and output indices that participate in the i-th subsystem and let $n_x^{\langle i \rangle}$, $n_u^{\langle i \rangle}$ and $n_r^{\langle i \rangle}$ be their cardinalities respectively. These sets are not assumed to be necessarily disjoint as some states and input may belong to multiple subsystems.

Assumption 1 *The pair $(\bar{A}, \bar{B})$ is stabilisable and F is an asymptotically stabilising gain for $(\bar{A}, \bar{B})$ and E possess the following structure:*

$$F_{s,j} = 0, \forall s \in \mathcal{I}_u^{\langle i \rangle}, \text{ and } j \notin \mathcal{I}_x^{\langle i \rangle}, \forall i \in \mathbb{N}_{[1,m]}, \tag{5.5}$$
$$E_{s,j} = 0, \forall s \in \mathcal{I}_u^{\langle i \rangle}, \text{ and } j \notin \mathcal{I}_r^{\langle i \rangle}. \forall i \in \mathbb{N}_{[1,m]}, \tag{5.6}$$

Under Assumption 1 the LCL can be decomposed into a set of local controllers whereby the i-th controller produces the control action $u^{\langle i \rangle} \in \mathbb{R}^{n_u^{\langle i \rangle}}$ using state measurements only from the i-th subsystem according to:

$$u_k^{\langle i \rangle} = F^{\langle i \rangle} x_k^{\langle i \rangle} + E^{\langle i \rangle} r_k^{\langle i \rangle}, \tag{5.7}$$

where $F^{\langle i \rangle} := F_{\mathcal{I}_u^{\langle i \rangle} \mathcal{I}_x^{\langle i \rangle}}$ and $E^{\langle i \rangle} := F_{\mathcal{I}_u^{\langle i \rangle} \mathcal{I}_r^{\langle i \rangle}}$ and $x_k^{\langle i \rangle} := x_{\mathcal{I}_x^{\langle i \rangle}}$, $u_k^{\langle i \rangle} := x_{\mathcal{I}_u^{\langle i \rangle}}$ and $r_k^{\langle i \rangle} := r_{\mathcal{I}_r^{\langle i \rangle}}$ for $i \in \mathbb{N}_{[1,m]}$.

The dynamics of the different subsystems are described by the set of difference equations:

$$\Sigma^{\langle i \rangle} : x_{k+1}^{\langle i \rangle} = A^{\langle i \rangle} x_k^{\langle i \rangle} + B^{\langle i \rangle} r_k^{\langle i \rangle} + d_k^{\langle i \rangle}, \tag{5.8}$$

where $A^{\langle i \rangle} := A_{\mathcal{I}_x^{\langle i \rangle} \mathcal{I}_x^{\langle i \rangle}}$, $B^{\langle i \rangle} := B_{\mathcal{I}_x^{\langle i \rangle} \mathcal{I}_r^{\langle i \rangle}}$ and $d_k^{\langle i \rangle}$ is a disturbance term to compensate for the unmodeled dynamics due to neglected state couplings between the subsystem $\Sigma^{\langle i \rangle}$ and its neighbours. The gains $F^{\langle i \rangle}$ are chosen so that the subsystems $\Sigma^{\langle i \rangle}$ are open-loop stable (with $r_k \equiv 0$ and $d_k \equiv 0$).

Assumption 2 *In addition to Assumption 1, for every $i \in \mathbb{N}_{[1,m]}$ the feedback gain $F^{\langle i \rangle}$ stabilises subsystem $\Sigma^{\langle i \rangle}$.*

Various methodologies have been proposed for the computation of such sparse stabilising gains (BBB10; Š91).

Let us define $\mathcal{J}_x^{\langle i \rangle} := \mathbb{N}_{[1,n_x]} \backslash \mathcal{I}_x^{\langle i \rangle}$, and $\mathcal{J}_r^{\langle i \rangle} := \mathbb{N}_{[1,n_r]} \backslash \mathcal{I}_r^{\langle i \rangle}$. The vectors $\tilde{x}^{\langle i \rangle} := x_{\mathcal{J}_x^{\langle i \rangle}}$ and $\tilde{r}^{\langle i \rangle} := r_{\mathcal{J}_r^{\langle i \rangle}}$ will be referred to as *neglected* states and references. The pair $(\tilde{A}^{\langle i \rangle}, \tilde{B}^{\langle i \rangle})$ with $\tilde{A}^{\langle i \rangle} := A_{\mathcal{I}_x^{\langle i \rangle} \mathcal{J}_x^{\langle i \rangle}}$ and $\tilde{B}^{\langle i \rangle} := B_{\mathcal{I}_x^{\langle i \rangle} \mathcal{J}_r^{\langle i \rangle}}$ will be used to describe the effect of the neglected states and references on the system $\Sigma^{\langle i \rangle}$.

Then the UCL comprises m subcontrollers which produce the reference signals $r_k^{\langle i \rangle}$ so as to keep the state $x^{\langle i \rangle}$ and the reference $r^{\langle i \rangle}$ inside the polytope:

$$\mathcal{Z}^{\langle i \rangle} := \{ \left[\begin{smallmatrix} x \\ r \end{smallmatrix} \right] \in \mathbb{R}^{n_x^{\langle i \rangle} + n_r^{\langle i \rangle}} : H_x^{\langle i \rangle} x + H_r^{\langle i \rangle} r \leq K^{\langle i \rangle} \}, \tag{5.9}$$

where $H_x^{\langle i \rangle} \in \mathbb{R}^{q_i \times n_x^{\langle i \rangle}}$, $H_r^{\langle i \rangle} \in \mathbb{R}^{q_i \times n_r^{\langle i \rangle}}$, and $K^{\langle i \rangle} \in \mathbb{R}^{q_i}$. The overall set of constraints is then defined as $\mathcal{Z} := \{ \left[\begin{smallmatrix} x \\ r \end{smallmatrix} \right] \in \mathbb{R}^{n_x + n_r} : (x^{\langle i \rangle}, r^{\langle i \rangle}) \in \mathcal{Z}^{\langle i \rangle}, \forall i \in \mathbb{N}_{[1,m]} \}$.

Let $A_0^{\langle i \rangle} \in \mathbb{R}^{n_x^{\langle i \rangle} \times n_x}$ be the matrix obtained by collecting the rows of A with indices in $\mathcal{I}_x^{\langle i \rangle}$ and setting to zero the elements in the columns $\mathcal{I}_x^{\langle i \rangle}$. Similarly, we construct $B_0^{\langle i \rangle} \in \mathbb{R}^{n_x^{\langle i \rangle} \times n_r}$ by collecting from B the rows indexed by $\mathcal{I}_x^{\langle i \rangle}$ and then zeroing the columns whose index is in $\mathcal{I}_r^{\langle i \rangle}$. Then, it holds that:

$$x_{k+1}^{\langle i \rangle} = A^{\langle i \rangle} x_k^{\langle i \rangle} + B^{\langle i \rangle} r_k^{\langle i \rangle} + A_0^{\langle i \rangle} x_k + B_0^{\langle i \rangle} r_k. \tag{5.10}$$

Additionally, let us define the set $\mathcal{Z} := \{(x, r) : (x^{\langle i \rangle}, r^{\langle i \rangle}) \in \mathcal{Z}^{\langle i \rangle}, \forall i \in \mathbb{N}_{[1,m]}\}$, which is a polytope and can be written in the form $\mathcal{Z} = \{(x, r) : H_x x + H_r r \leq K\}$. Let the reference vector $r^{\langle i \rangle}$ be constrained in the set:

$$\mathcal{R}^{\langle i \rangle} := \{r^{\langle i \rangle} \in \mathbb{R}^{n_r^{\langle i \rangle}} : (H_x^{\langle i \rangle} G^{\langle i \rangle} + H_r^{\langle i \rangle}) r^{\langle i \rangle} \leq K^{\langle i \rangle} - \Delta K^{\langle i \rangle}\},$$

where $G^{\langle i \rangle} := (I - A^{\langle i \rangle})^{-1} B^{\langle i \rangle}$ is the reference-to-state static gain for $\Sigma^{\langle i \rangle}$ and $\Delta K^{\langle i \rangle} \geq 0$. We assume that the reference signals $r_k^{\langle i \rangle}$ retain the tracking error $\Delta x_k^{\langle i \rangle} := x_k^{\langle i \rangle} - G^{\langle i \rangle} r_k^{\langle i \rangle}$ in the set:

$$\mathcal{E}^{\langle i \rangle} = \{\Delta x^{\langle i \rangle} \in \mathbb{R}^{n_x^{\langle i \rangle}} : H_x^{\langle i \rangle} \Delta x^{\langle i \rangle} \leq \Delta K^{\langle i \rangle}\}. \tag{5.11}$$

Notice that $\Delta x_k^{\langle i \rangle} \in \mathcal{E}^{\langle i \rangle}$ if and only if $(x_k^{\langle i \rangle}, r_k^{\langle i \rangle}) \in \tilde{\mathcal{E}}^{\langle i \rangle}$ where:

$$\tilde{\mathcal{E}}^{\langle i \rangle} := \{\begin{bmatrix} x^{\langle i \rangle} \\ r^{\langle i \rangle} \end{bmatrix} \in \mathbb{R}^{n_x^{\langle i \rangle} + n_r^{\langle i \rangle}} : x^{\langle i \rangle} - G^{\langle i \rangle} r^{\langle i \rangle} \in \mathcal{E}^{\langle i \rangle}\}. \tag{5.12}$$

If we set $z^{\langle i \rangle} := G^{\langle i \rangle} r^{\langle i \rangle} = A^{\langle i \rangle} z^{\langle i \rangle} + B^{\langle i \rangle} r^{\langle i \rangle}$, then the dynamics of $\Sigma^{\langle i \rangle}$ can be described in terms of $\Delta x^{\langle i \rangle} = x^{\langle i \rangle} - z^{\langle i \rangle}$ as follows:

$$\Delta x_{k+1}^{\langle i \rangle} = A^{\langle i \rangle} \Delta x_k^{\langle i \rangle} + d_k^{\langle i \rangle}, \tag{5.13}$$

where, under the assumptions that $(x_k^{\langle i \rangle}, r_k^{\langle i \rangle}) \in \mathcal{Z}^{\langle i \rangle}$ and $\Delta x_k^{\langle i \rangle} \in \mathcal{E}^{\langle i \rangle}$ for all $k \in \mathbb{N}$ and $i \in \mathbb{N}_{[1,m]}$, the disturbance $d_k^{\langle i \rangle}$ is drawn from the polytope:

$$\mathcal{D}^{\langle i \rangle} = \left\{ d^{\langle i \rangle} \in \mathbb{R}^{n_x^{\langle i \rangle}} \;\middle|\; \begin{array}{l} \exists r \in \mathbb{R}^{n_r}, \exists x \in \mathbb{R}^{n_x}, \text{s.t.:} d^{\langle i \rangle} = A_0^{\langle i \rangle} x + B_0^{\langle i \rangle} r, \\ \text{and } \forall j \in \mathbb{N}_{[1,m]} : (x^{\langle j \rangle}, r^{\langle j \rangle}) \in \mathcal{Z}^{\langle j \rangle} \cap \tilde{\mathcal{E}}^{\langle j \rangle} \end{array} \right\}. \tag{5.14}$$

The size of this polytope determines how strongly the i-th subsystem is dynamically coupled with its neighbours.

Let $\Omega^{\langle i \rangle}(0)$ be the maximal robustly positive invariant set for (5.13) under the constraints $\Delta x^{\langle i \rangle} \in \mathcal{E}^{\langle i \rangle}$ and for $d_k^{\langle i \rangle} \in \mathcal{D}^{\langle i \rangle}$ for all $k \in \mathbb{N}$. Let $\Omega^{\langle i \rangle}(0)$ have the minimal representation $\Omega^{\langle i \rangle}(0) = \{x \in \mathbb{R}^{n_x^{\langle i \rangle}} : H_0^{\langle i \rangle} x \leq K_0^{\langle i \rangle}\}$, counting $n_0^{\langle i \rangle}$ inequalities. Under Assumption 2 this set exists and is a finitely generated polytope.

The complexity of the computation of a maximal RPI set for the overall large-scale system can prove preventive even for offline computations. Note, however, that the computation of the maximal RPI sets is done in a decentralised fashion. For $r \in \mathcal{R}^{\langle i \rangle}$ we define the sets

$$\Omega^{\langle i \rangle}(r) := \{x \in \mathbb{R}^{n_x^{\langle i \rangle}} : x - G^{\langle i \rangle} r \in \Omega^{\langle i \rangle}(0)\}. \tag{5.15}$$

Lemma 1 *For all $i \in \mathbb{N}_{[1,m]}$ let $x_0^{\langle i \rangle} \in \Omega^{\langle i \rangle}(r^{\langle i \rangle})$ and assume that $r_k^{\langle i \rangle} = r^{\langle i \rangle} \in \mathcal{R}^{\langle i \rangle}$ for all $k \in \mathbb{N}$. Then $(x_k^{\langle i \rangle}, r_k^{\langle i \rangle}) \in \mathcal{Z}^{\langle i \rangle}$ for all $k \in \mathbb{N}$ and $i \in \mathbb{N}_{[1,m]}$.*

Proof. Since $x_0^{\langle i \rangle} \in \Omega^{\langle i \rangle}(r^{\langle i \rangle})$ for all $i \in \mathbb{N}_{[1,m]}$ it is $\Delta x_k^{\langle i \rangle} \in \mathcal{E}^{\langle i \rangle}$ for all $k \in \mathbb{N}$, or what is the same:

$$\begin{aligned} & H_x^{\langle i \rangle} \Delta x_k^{\langle i \rangle} \leq \Delta K^{\langle i \rangle} \\ \Leftrightarrow & H_x^{\langle i \rangle} x_k^{\langle i \rangle} - H_x^{\langle i \rangle} G^{\langle i \rangle} r^{\langle i \rangle} \leq \Delta \end{aligned}$$

But because $r^{\langle i \rangle} \in \mathcal{R}^{\langle i \rangle}$ one has that $\Delta K^{\langle i \rangle} \leq K^{\langle i \rangle} - H_x^{\langle i \rangle} G^{\langle i \rangle} r^{\langle i \rangle} - H_r^{\langle i \rangle} r^{\langle i \rangle}$, therefore $H_x^{\langle i \rangle} G^{\langle i \rangle} r^{\langle i \rangle} \leq \Delta K^{\langle i \rangle} \leq K^{\langle i \rangle} - H_x^{\langle i \rangle} G^{\langle i \rangle} r^{\langle i \rangle} - H_r^{\langle i \rangle} r^{\langle i \rangle}$ which proves the assertion. □

5.2.3 Computation of maximum reference variations

Assume that a set of fixed reference rates $N^{\langle i \rangle}$ for $i \in \mathbb{N}_{[1,m]}$ is given. In this section we will compute upper bounds on the element-wise variations of the reference rates $r^{\langle i \rangle}$ so that $(x_k^{\langle i \rangle}, r_k^{\langle i \rangle})$ satisfies the prescribed constraints (5.9). For every subsystem $i \in \mathbb{N}_{[1,m]}$ we formulate the problem of determining the minimum element-wise change in the reference signal that may lead the initial state $x_{\nu N}^{\langle i \rangle}$ outside $\Omega^{\langle i \rangle}(r^{\langle i \rangle, \nu})$; the problem is stated as follows:

$$\mathbb{P}_N^{\langle i \rangle} \colon \rho^{\langle i \rangle}(N) := \min_{r^1, r^2, x_0, d_0, \ldots, d_{N-1}} \|r^1 - r^2\|_\infty \tag{5.16a}$$

subject to:

$$r^1, r^2 \in \mathcal{R}^{\langle i \rangle}, \tag{5.16b}$$

$$x_0 \in \Omega^{\langle i \rangle}(r^1), \tag{5.16c}$$

$$d_j^{\langle i \rangle} \in \mathcal{D}^{\langle i \rangle}, \forall j \in \mathbb{N}_{[0,N-1]}, \tag{5.16d}$$

$$(A^{\langle i \rangle})^N x_0 + \Gamma_N^{\langle i \rangle} r^2 + \sum_{j=0}^{N-1} (A^{\langle i \rangle})^{N-j-1} d_j^{\langle i \rangle} \notin \Omega^{\langle i \rangle}(r^2), \tag{5.16e}$$

where $\Gamma_N^{\langle i \rangle} := \sum_{j=0}^{N-1} (A^{\langle i \rangle})^j B^{\langle i \rangle}$. The above optimisation problem can be formulated as a MILP.

The value function of (5.16a) enjoys a very useful property: it is non-decreasing with respect to N. If $\mathbb{P}_N$ is infeasible for some N, this implies that for all $r^{\nu-1}, r^\nu \in \mathcal{R}$ it is $x^{\nu+1} \in \Omega(r^\nu)$ whenever $x^\nu \in \Omega(r^{\nu-1})$. In this case we set $\rho(N) = \infty$.

Theorem 1 *Let F be a (decentralised) asymptotically stabilising gain satisfying Assumption 2. Assume that for every subsystem $i \in \mathbb{N}_{[1,m]}$ there is a $\sigma^{\langle i \rangle} > 0$ so that the references $r^{\langle i \rangle, \nu}$ produced by the upper-layer controllers satisfy the following rate constraint at all time instants $\nu \in \mathbb{N}$:*

$$\|r^{\langle i \rangle, \nu} - r^{\langle i \rangle, \nu-1}\|_\infty \leq \rho^{\langle i \rangle}(N^{\langle i \rangle}) - \sigma^{\langle i \rangle}, \tag{5.17a}$$

$$r^{\langle i \rangle, \nu-1}, r^{\langle i \rangle, \nu} \in \mathcal{R}^{\langle i \rangle}. \tag{5.17b}$$

Let $x_0^{\langle i \rangle} \in \Omega^{\langle i \rangle}(r^{-1,\langle i \rangle})$ for all $i \in \mathbb{N}_{[1,m]}$. Then the linear system (5.1) *with the feedback control law* (5.2) *satisfies the the constraints $\left[\begin{smallmatrix} x_k \\ r_k \end{smallmatrix}\right] \in \mathcal{Z}$ for all $k \in \mathbb{N}$. Additionally, if $\lim_{k\to\infty} r_k = r$ with $r \in \mathcal{R}$, then $\lim_{k\to\infty} x_k = Gr$.*

Proof. If $x_0^{\langle i \rangle} \in \Omega^{\langle i \rangle}(r^{-1,\langle i \rangle})$ for all $i \in \mathbb{N}_{[1,m]}$, then by equations 5.17 and Lemma 1 it follows that $(x_k^{\langle i \rangle}, r_k^{\langle i \rangle}) \in \mathcal{Z}^{\langle i \rangle}$ for all $k \in \mathbb{N}$. Let $\zeta_k := x_k - Gr_k$ and $v_k := r_k - r$, so the dynamics of the overall system can be rewritten as $\zeta_{k+1} = A\zeta_k + Bv_k$. Based on above decomposition the subsystem i can be represented as

$$\Sigma^{\langle i \rangle} : \zeta_{k+1}^{\langle i \rangle} = A^{\langle i \rangle} \zeta_k^{\langle i \rangle} + B^{\langle i \rangle} r_k^{\langle i \rangle} + d_k^{\langle i \rangle}. \tag{5.18}$$

For i^{th} subsystem, $A^{\langle i \rangle}$ is a stable matrix under assumption 2 and whenever $\mathcal{D} = \{0\}$, the system (5.18) is input-to-state stable (JW01),

from which it follows that $\lim_{k\to\infty} \zeta_k = 0$ (equivalently $\lim_{k\to\infty} x_k = Gr$), therefore $\lim_{k\to\infty} y_k = \Theta r$. □

The UCL control action can be computed by a model predictive control strategy where any optimality criterion can be used so long as the constraints (5.17) are satisfied.

The next result, Theorem 2, states that for every system i there is a rate $N_\star^{\langle i\rangle}$ for which $\rho^{\langle i\rangle}(N^\star)$ is equal to infinity, that is, if the upper control layer is slow enough — so that the lower control layer has enough time to steer the system state very closed to the previously assigned set-point — then there is no need to impose constraints on $\Delta r^{\langle i\rangle}$. Note that we require that "the sampling time of the lower control layer is fixed", so as $N^{\langle i\rangle}$ increases the UCL becomes slower.

Theorem 2 *Assume that Assumption 2 is satisfied and $\mathcal{R}^{\langle i\rangle}$ is a nonempty compact set. Assume further that the sampling time of the lower control layer is fixed. If $\Omega(0)^{\langle i\rangle}$ is of full affine dimension, then there exists a $N_\star^{\langle i\rangle} \in \mathbb{N}$ so that $\rho^{\langle i\rangle}(N_\star^{\langle i\rangle}) = \infty$.*

Proof. We assume that $\mathcal{D}^{\langle i\rangle} = \{0\}$ and fix an $i \in \mathbb{N}_{[1,m]}$. Since $\Omega^{\langle i\rangle}(0)$ has full affine dimension and contains the origin in its interior, there is an $\epsilon > 0$ so that $\epsilon\mathcal{B}_\infty \subsetneq \Omega^{\langle i\rangle}(0)$. The compactness or $\mathcal{R}^{\langle i\rangle}$ implies that there is a $\delta > 0$ so that $\mathcal{R}^{\langle i\rangle} \subseteq \delta\mathcal{B}_\infty$. Since $A^{\langle i\rangle}$ is strictly Schur, the limit $\lim_{N\to\infty} \Gamma_N^{\langle i\rangle}$ exists and is equal to $G^{\langle i\rangle}$.

This implies that there is a $N_1^{\langle i\rangle} \in \mathbb{N}$ so that for all $N \geq N_1^{\langle i\rangle}$ it is $\|\Gamma_N^{\langle i\rangle} - G^{\langle i\rangle}\|_\infty \leq \frac{\epsilon}{2}\delta$. For $z \in \mathcal{R}^{\langle i\rangle}$ we have $\|(\Gamma_N^{\langle i\rangle} - G^{\langle i\rangle})z\|_\infty \leq \|\Gamma_N^{\langle i\rangle} - G^{\langle i\rangle}\|_\infty \cdot \|z\| \leq \frac{\epsilon}{2}$, or what is the same $\Gamma_N^{\langle i\rangle} z \in G^{\langle i\rangle} z \oplus \frac{\epsilon}{2}\mathcal{B}_\infty$ for all $z \in \mathcal{R}^{\langle i\rangle}$. We can hence infer that[1]

Again because of Assumption 2 and because of the compactness of $\mathcal{R}^{\langle i\rangle}$, there exists a $N_2^{\langle i\rangle} \in \mathbb{N}$, $N_2^{\langle i\rangle} \geq N_1^{\langle i\rangle}$ so that for all $N \geq N_2^{\langle i\rangle}$ it is $(A^{\langle i\rangle})^N x_0 \in (\epsilon/2)\mathcal{B}_\infty$ for all $x_0 \in \mathcal{R}$. Because of (5.16e), for $\mathbb{P}_N^{\langle i\rangle}$ to be infeasible for some N it is necessary and sufficient that for all

[1]We make use of the notation $\Gamma_N^{\langle i\rangle}\mathcal{R}^{\langle i\rangle} := \{\Gamma_N^{\langle i\rangle} r; r \in \mathcal{R}^{\langle i\rangle}\}$, $\Omega^{\langle i\rangle}(\mathcal{R}^{\langle i\rangle}) := \{\Omega^{\langle i\rangle}(r); r \in \mathcal{R}^{\langle i\rangle}\}$ and $\delta\mathcal{B}_\infty := \{\delta z : z \in \mathcal{B}_\infty\} = \{z : \|z\|_\infty \leq \delta\}$.

$$\Gamma_N^{\langle i\rangle}\mathcal{R}^{\langle i\rangle} \subseteq G^{\langle i\rangle}\mathcal{R}^{\langle i\rangle} \oplus \tfrac{\epsilon}{2}\mathcal{B}_\infty. \tag{5.19}$$

$r^1, r^2 \in \mathcal{R}^{\langle i \rangle}$ and $x_0 \in \Omega^{\langle i \rangle}(r^1)$ it is $(A^{\langle i \rangle})^N x_0 + \Gamma_N^{\langle i \rangle} r^1 \in \Omega^{\langle i \rangle}(r^2)$, which can be equivalently writen as:

$$\Gamma_N^{\langle i \rangle} \mathcal{R}^{\langle i \rangle} \subseteq \Omega^{\langle i \rangle}(\mathcal{R}^{\langle i \rangle}) \ominus (A^{\langle i \rangle})^N \mathcal{R}^{\langle i \rangle}. \tag{5.20}$$

For $N \geq N_\star^{\langle i \rangle} := \max\{N_1^{\langle i \rangle}, N_2^{\langle i \rangle}\}$ and in light of (5.19) this inclusion is satisfied if $G^{\langle i \rangle} \mathcal{R}^{\langle i \rangle} \oplus (\epsilon/2)\mathcal{B}_\infty \subseteq \Omega^{\langle i \rangle}(\mathcal{R}^{\langle i \rangle}) \ominus (\epsilon/2)\mathcal{B}_\infty$ from which we have that:

$$G^{\langle i \rangle} \mathcal{R}^{\langle i \rangle} \subseteq \Omega^{\langle i \rangle}(\mathcal{R}^{\langle i \rangle}) \ominus \epsilon \mathcal{B}_\infty, \tag{5.21}$$

but by definition — see (5.15) — $\Omega^{\langle i \rangle}(r) = \{G^{\langle i \rangle} r\} \oplus \Omega^{\langle i \rangle}(0)$, thus $\Omega^{\langle i \rangle}(\mathcal{R}^{\langle i \rangle}) = G^{\langle i \rangle} \mathcal{R}^{\langle i \rangle} \oplus \Omega^{\langle i \rangle}(0)$ and a condition for infeasibility is:

$$G^{\langle i \rangle} \mathcal{R}^{\langle i \rangle} \subseteq G^{\langle i \rangle} \mathcal{R}^{\langle i \rangle} \oplus \left(\Omega^{\langle i \rangle}(0) \ominus \epsilon \mathcal{B}_\infty \right), \tag{5.22}$$

which is true since $\Omega^{\langle i \rangle}(0) \ominus \epsilon \mathcal{B}_\infty$ is nonempty. □

5.3 Simulations - Control of a system of interconnected Tanks

5.3.1 System dynamics and decomposition

The proposed methodology is tested on Johansson's quadruple-tank process (Joh00) where the control objective is to track given (possibly time-varying, piece-wise constant) references s_1 and s_2 for the levels of tanks 1 and 2, namely h_1 and h_2, as in Fig. 24 by manipulating the inflows q_a and q_b. Constraints are imposed on the maximum flow that can be achieved by each pump and on the upper and lower allowed levels of water in the tanks.

The system is subject to state and input constraints and its dynamics is described in (ALdlP$^+$11) by the system of continuous-time non-

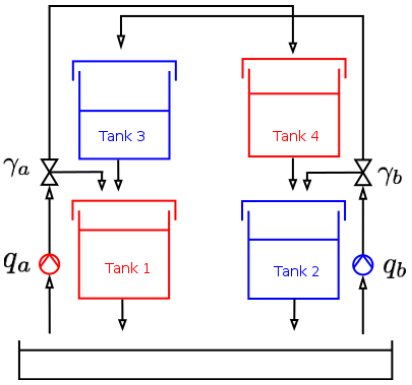

Figure 24: Johansson's quadruple-tank process where the two sub-systems are denoted with different colours.

linear equations

$$S_1 \frac{\mathrm{d}h_1}{\mathrm{d}t} = -a_1\sqrt{2gh_1} + a_3\sqrt{2gh_3} + \gamma_a q_a, \tag{5.23a}$$

$$S_2 \frac{\mathrm{d}h_2}{\mathrm{d}t} = -a_2\sqrt{2gh_2} + a_4\sqrt{2gh_4} + \gamma_b q_b, \tag{5.23b}$$

$$S_3 \frac{\mathrm{d}h_3}{\mathrm{d}t} = -a_3\sqrt{2gh_3} + (1-\gamma_b) q_b, \tag{5.23c}$$

$$S_3 \frac{\mathrm{d}h_4}{\mathrm{d}t} = -a_4\sqrt{2gh_4} + (1-\gamma_a) q_a. \tag{5.23d}$$

The maximum allowed level for tanks 1 and 2 is set to 1.36 m and for tanks 3 and 4 to 1.30 m. The minimum allowed level in all tanks is 0.2 m. The maximum flows are $q_{a,\max} = 3.26$ m^3/h and $q_{b,\max} = 4$ m^3/h; no negative flows are possible. The values of the other parameters of the system are $a_1 = 1.31 \cdot 10^{-4}$ m^2, $a_2 = 1.51 \cdot 10^{-4}$ m^2, $a_3 = 9.27 \cdot 10^{-5}$ m^2, $a_4 = 8.82 \cdot 10^{-5}$ m^2, $S_1 = S_2 = 0.06$ m^2, $S_3 = S_4 = 0.20$ m^2, and $\gamma_a = \gamma_b = 0.5$. The nonlinear system is linearised about the steady state $u^0 = (2.6, 2.6)'$ m^3/h and $x^0 = (0.6545, 0.4926, 0.7852, 0.8583)'$ m and discretised with sampling period $T_s = 10$s.

We define the discrete-time state vector $x_k = (h_{1,k}, h_{2,k}, h_{3,k}, h_{4,k})'$

which comprises the levels of the four tanks, the discrete-time input vector $u_k = (q_{a,k}, q_{b,k})'$ of manipulated variables which are the two flows, and the discrete output $y_k = x_k$. The linearised discrete-time system is written in the form of (5.1).

5.3.2 Control of Johansson's system

We consider that the lower control layer operates at sampling time $T_s = 10$s. The overall system is partitioned into two subsystems with $\mathcal{I}_x^{\langle 1\rangle} = \{1, 4\}$, $\mathcal{I}_x^{\langle 2\rangle} = \{2, 3\}$ and $\mathcal{I}_u^{\langle 1\rangle} = \{1\}$, $\mathcal{I}_u^{\langle 2\rangle} = \{2\}$. The system is controlled by means of the proposed decentralised hierarchical control methodology which is compared to its centralised hierarchical variant. Reference commands from the upper layer controller are computed so that they minimise a quadratic cost function. In particular, the UCL for subsystem 1 solves the following minimisation problem at the UCL sampling time instant ν:

$$J^{\langle 1\rangle\star}(x^\nu) = \min_{\{r_1^{\nu+j}\}_{j=0}^{N-1}} \sum_{k=0}^{N-1} (h_1^{\nu+k} - s_1)^2 + \lambda (r_a^{\nu+k} - r_a^s)^2, \qquad (5.24)$$

subject to the (linearised) system dynamics, measurements from the system, the requirement $r^{\nu+k} \in \mathcal{R}$ for all $k = 0, \ldots, N-1$, and the bounds on the maximum reference variation that accrue from Theorem 1. In what follows, the weight λ is fixed to 0.01. Then, the solution of problem 5.24 yields an optimal sequence of references $\{r_1^{\nu+k,\star}\}_{k=0}^{N-1}$, the first one of which – namely $r_1^{\nu,\star}$ is applied to the corresponding controlled LCL system in a receding horizon fashion. The UCL controller for sub-system 2 works in an analogous fashion where the minimisation problem becomes

$$J^{\langle 2\rangle\star}(x^\nu) = \min_{\{r_2^{\nu+j}\}_{j=0}^{N-1}} \sum_{k=0}^{N-1} (h_2^{\nu+k} - s_2)^2 + \lambda (r_b^{\nu+k} - r_b^s)^2,$$

subject to the corresponding constraints. According to Theorem 1 the closed-loop system will satisfy the prescribed constraints.

For the decentralised control case, the dependence of the maximum reference change $\rho^{\langle i\rangle}$ on N is presented in Figure 25. The reference rate $N = 40$ was selected for which $\rho^{\langle 1\rangle}(N) = 0.0034$ and

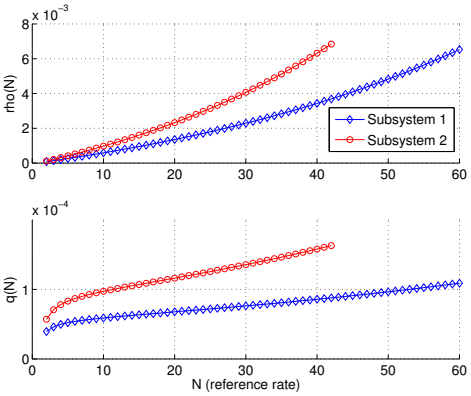

Figure 25: The functions $\rho^{\langle i \rangle}(N)$ and $q^{\langle i \rangle}(N) := \rho^{\langle i \rangle}(N)/N$.

$\rho^{\langle 2 \rangle}(N) = 0.0063$ for the decentralised control system and $\rho(N) = 0.0035$ for the centralised control approach. The maximum reference variation $\rho^{\langle i \rangle}(N)$ for the two subsystems is presented in Figure 25. Notice that for $N \geq N^\star = 42$, it is $\rho^{\langle 2 \rangle}(N) = \infty$.

The controlled trajectories of the tank levels are presented in Figures 26 to 28. The tank levels h_1 and h_2 are steered towards four different set-points and the set-point values are kept constant for 5.55h. In order to quantify the performance of the three controllers, we use the following index introduced by Alvarado *et al.* (ALdlP$^+$11) for the same system:

$$J = \sum_{k=0}^{N_s-1} (h_{1,k}-s_{1,k})^2+(h_{2,k}-s_{2,k})^2 + \kappa((q_{a,k}-q^s_{a,k})^2 + (q_{b,k}-q^s_{b,k})^2), \tag{5.25}$$

where $\kappa = 0.01$ and $q^s_{a,k}$ and $q^s_{b,k}$ are the steady-state values of the input variables that correspond to the set-point defined by s_1 and s_2, and $N_s = 8000$ $(22h)$ is the simulation horizon. The values of the performance index J are presented in Table 4.

The maximal robust positive invariant sets $\Omega^{\langle i \rangle}(0)$, $i \in \{1, 2\}$ for the decentralised control case were computed offline in 1.97s and 2.19s

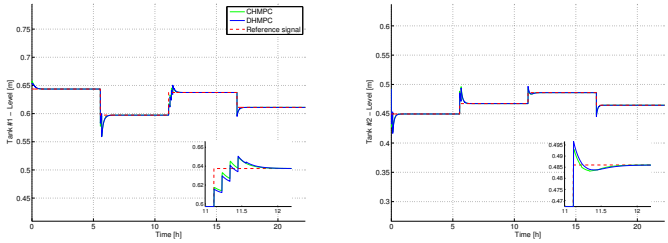

Figure 26: The level in tank 1 and 2: Comparison between centralised hierarchical MPC (CHMPC, green) and decentralised hierarchical MPC (DHMPC, blue). The dashed red line represents the set-point s_1. The inset shows the convergence of the tank level to the desired set-point in the interval 11 to 12.2h.

Controller	$\tau_{s,1}$ (h)	$\tau_{s,2}$ (h)	J
DHMPC	0.1674	0.1500	0.1495
CHMPC	0.1146	0.1458	0.1516

Table 4: Performance of a decentralised and a centralised controller for Johansson's system.

and their minimal representations involved 5 and 4 inequalities respectively. The maximal positive invariant set $\Omega(0)$ for the centralised control system was computed in 0.60s and its minimal representation comprised 12 linear inequalities. The associated MILPs $\mathbb{P}_N^{\langle i \rangle}$ as in (5.16) were solved offline in 2.12s for subsystem 1 and 2.27s for subsystem 2 on average. The corresponding centralised computation required 6.33s on average. All reported computation times were measured in MATLAB 2013a running on a Mac OS X machine, 2.66GHz Intel Core 2 Duo, 4GB RAM.

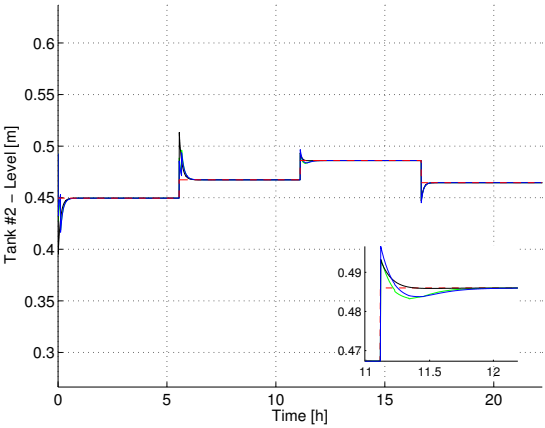

Figure 27: The level in tank 2 and comparison between CHMPC, DHMPC and offset-free MPC operating at the LCL sampling frequency.

5.4 Conclusions

We developed a methodology for the decentralised hierarchical multi-rate control of constrained linear systems with additive uncertainties and quantified the effect of the sampling times on the responsiveness of the closed-loop while prescribing sufficient conditions for robust stability of the controlled system. The applicability of the proposed method is demonstrated on the Johansson's system. For future work, we would like to extend this to complete drinking water network.

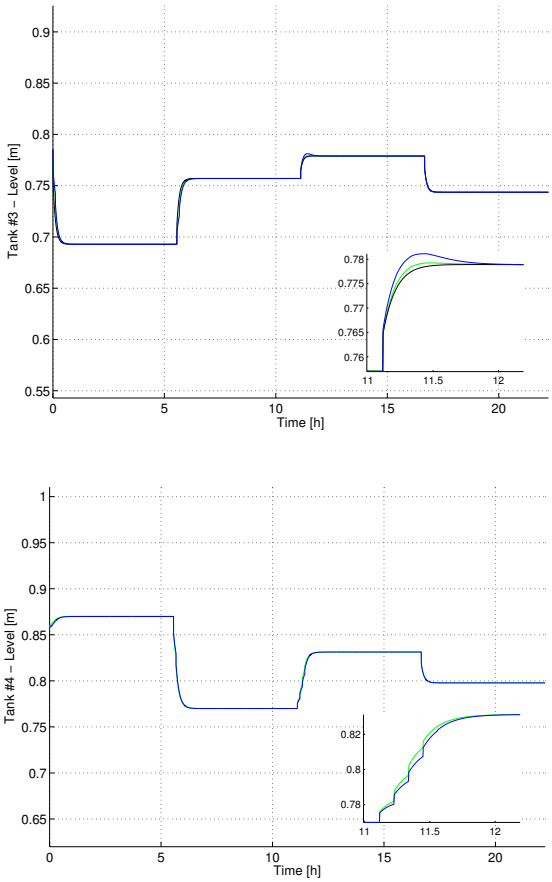

Figure 28: The level in tanks 3 and 4: Closed-loop trajectories for CHMPC (green), and DHMPC (blue).

Chapter 6

Proximal quasi-Newton methods for scenario-based stochastic optimal control

In Chapters 3 and 4 we have shown that the structure of the stochastic optimal control problems can be exploited by first-order methods leading to an efficient parallelisable solution method. In this chapter, we use the *forward-backward envelope* — a real-valued, continuously differentiable penalty function which shares the same minimisers with the original optimisation problem and allows us to recast the original nonsmooth problem as an unconstrained problem which we solve with a limited-memory BFGS algorithm.

6.1 Introduction

6.1.1 Motivation and background

Scenario-based stochastic model predictive control is becoming increasingly popular in control applications for its ability to deliver control actions with foresight under uncertainty and has been used for the control of power dispatch (HSB$^+$15; PTB11), heating, ventilation and air-conditioning (HVAC) of buildings (ZSSM13), macroeconomic systems (PSSB14), supply chains (SM16) and many another. The optimisation problems associated with such problems are typically of rather large dimension (involving millions of decision variables), but they possess a rich structure which gradient-based methods have been shown to be able to exploit in previous Chapters (see Chap. 3 and 4). Such methods converge at a rate of $\mathcal{O}(1/k)$ and $\mathcal{O}(1/k^2)$ using Nesterov's extrapolation technique (Nes83). Nevertheless, first-order methods are sensitive to ill-conditioning which may not always be possible to mitigate by preconditioning.

A straightforward approach to improve the convergence properties of first-order methods is to introduce second-order information. However, this is not available in many cases of interest, or, it is very hard to compute. In such cases, quasi-Newton algorithms (NW06) produce successive approximations of the Hessian by measuring the change of gradient and update a Hessian estimate using the present gradient and the previous Hessian estimate. This algorithms generate a sequence iterates which converges Q-superlinearly to the optimal solution. The simplest quasi-Newton method is symmetric rank-one (SR1) (CGT91) where it makes a symmetric rank-one change to Hessian at each iterate. This method is computationally simpler but the update does not preserve the positive definiteness of the Hessian even though the initial Hessian is positive definite and sometimes can lead to numerical errors (division by zero).

The BFGS (Broyden-Fletcher-Goldfarb-Shanno) method (Bro70) is another algorithm belonging to the class of quasi-Newton algorithms which updates the Hessian by performing a rank-2 update at each iterate. This method always guarantee positive definiteness of the Hessian and doesn't have any numerical issues. But this could be severe limitation as one needs to store and update a very large dense matrix; it is thus unsuitable for large-scale optimisation.

The limited-memory BFGS (L-BFGS) method (LN89) has been successfully used in optimisation problems where computing Hessian approximations are expensive and has also been applied for solving huge-scale problems (CWZ14; LNC$^+$11). This method modifies the computation of the Hessian from most recent iterates; Curvature information from earlier iterations, which is less likely to be relevant to the actual behaviour of the Hessian at the current iteration, is discarded in the interests of saving storage. It *implicitly* updates a diagonal approximation of the Hessian using a computationally cheap algorithm known as the *two-loop recursion* (NW06). Despite its popularity it comes with two limitations which have hindered its use for the solution of optimal control problems. First, it can only be applied to unconstrained optimal control problems or problems with only box constraints on the input variables (BLNZ95) and second, it cannot be applied to nonsmooth problems (LMO08).

Becker *et al.* (BF12) approximated the smooth function f in the proximal-gradient algorithm by means of the Q-norm, where Q is the Hessian approximation that is calculated using quasi-Newton update. Now the proximal step becomes prox_{Qg} which is computationally demanding unless we restrict the structure of the Q to, say, diagonal matrices. These limitations are lifted using the *forward-backward envelope* (FBE) of the original optimisation problem which allows us to reformulate it as an unconstrained problem of a continuously differentiable function (PSB14; STP16).

We previously showed that stochastic optimal control problems possess a certain structure which can be exploited for their efficient numerical solution using an accelerated proximal gradient algorithm in Chapter 3. In this chapter, we propose a quasi-Newtonian algorithm combining the limited-memory BFGS method with the forward-backward envelope function to achieve faster convergence while retaining the separability and parallelisability characteristics of the problem.

6.1.2 Outline

In Section 6.2 we introduce the stochastic optimal control problem and the corresponding scenario-based formulation. Section 6.3 defines the forward-back envelope function for the corresponding optimisation

problem of scenario MPC. This function is smooth, continuously differentiable real-valued which can be solved with the L-BFGS method. We demonstrate that the computation of the gradient of the FBE can be performed in parallel exploiting the structure of the problem. Finally in Section 6.4 we compare the convergence of the L-BFGS algorithm and the accelerated proximal gradient algorithm.

6.1.3 Notation

Let $\mathbb{R}$, $\mathbb{N}$, $\mathbb{R}^n$, $\mathbb{R}^{m\times n}$, $\mathbb{S}^n_+$, $\mathbb{S}^n_{++}$ denote the sets of real numbers, nonnegative integers, column real vectors of length n, real matrices of dimensions m-by-n, symmetric positive semidefinite and positive definite n-by-n matrices respectively. Let $\overline{\mathbb{R}} = \mathbb{R} \cup \{\pm\infty\}$ denote the set of extended-real numbers. The transpose of a matrix A is denoted by $A^\top$ and $\langle x, y\rangle$ stands for the standard inner product of x and y. The set of of nonnegative integers $\{k_1, k_1+1, \ldots, k_2\}, k_2 \geq k_1$ is denoted by $\mathbb{N}_{[k_1,k_2]}$.

The *indicator function* of a set $C \subseteq \mathbb{R}^n$ is the extended-real valued function $\delta(\cdot|C) : \mathbb{R}^n \to \overline{\mathbb{R}}$ and for $x \in C$ it is $\delta(x|C) = 0$ and $\delta(x|C) = +\infty$ otherwise. A function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ is called *lower semi-continuous* or *closed* if for every $x \in \mathbb{R}^n$, $f(x) = \liminf_{z\to x} f(z)$. A $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ is called *proper* if there is a $x \in \mathbb{R}^n$ so that $f(x) < \infty$ and $f(x) > -\infty$ for all $x \in \mathbb{R}^n$. For a closed convex function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$, we define its *conjugate* as $f^*(x^*) = \sup_x\{\langle x, x^*\rangle - f(x)\}$. A $F : \mathbb{R}^n \to \mathbb{R}^m$ is called *β-Lipschitz continuous*, with $\beta \geq 0$, if $\|F(x_1) - F(x_2)\|_* \leq \beta\|x_1 - x_2\|$ for every $x_1, x_2 \in \mathbb{R}^n$. We call f *σ-strongly convex* if $f(x) - \frac{\sigma}{2}\|x\|_2^2$ is a convex function. Unless otherwise stated $\|\cdot\|$ stands for $\|\cdot\|_2$.

Every nonempty closed convex set $C \subseteq \mathbb{R}^n$ defines the convex function $\mathrm{proj}(x|C) = \mathrm{argmin}_{c\in C} \|x - c\|_2$, which is called the *(Euclidean) projection* of x onto C. The Euclidean distance of a $x \in \mathbb{R}^n$ from C is defined as $d(x|C) = \min_{c\in C} \|x - c\|_2$.

6.2 Problem statement

In this section, we just recap the stochastic optimal control problem and the reduced scenario tree based MPC from Chapter 3.

6.2.1 Stochastic optimal control

We first provide a formal statement of general stochastic optimal control problems for linear dynamical systems. Let $(\Omega, \mathfrak{F}, \mathrm{P})$ be a probability space and $\{\varnothing, \Omega\} = \mathfrak{F}_0 \subseteq \mathfrak{F}_1 \subseteq \ldots \subseteq \mathfrak{F}_{N-1} = \mathfrak{F}$ be a nested sequence of σ-algebras known as a *filtration* (SDR09). We shall use the notation $v \lhd \mathfrak{F}_k$ to denote that $v : \Omega \to \mathbb{R}$ is a $\mathfrak{F}_k$-measurable random variable. Consider the stochastic discrete-time linear system

$$x_{k+1} = A_{\xi_k} x_k + B_{\xi_k} u_k + w_{\xi_k}, \tag{6.1}$$

where $\xi_k \lhd \mathfrak{F}_k$, $u_k \lhd \mathfrak{F}_{k-1}$ and with known initial condition $x_0 = p$. This practically means that u_k is a *causal* control law, i.e., it is a function $u_k = \psi_k(p, \xi_0, \ldots, \xi_{k-1})$ for $k \in \mathbb{N}_{[1,N-1]}$ and $u_0 = \psi_0(x_0)$[1].

A *stochastic optimal control problem* for (6.1) with horizon N and decision variable $\pi = \{u_k\}_{k \in \mathbb{N}_{[0,N-1]}}$ can be formulated as

$$V^\star(p) = \min_{\pi} \ \mathbb{E}\left[V_f(x_N, \xi_N) + \sum_{k=0}^{N-1} \ell(x_k, u_k, \xi_k) \right], \tag{6.2}$$

subject to (6.1) and the condition $x_0 = p$ and $\mathbb{E}$ is the expectation operator of the product probability space of the filtered probability space $(\Omega, \mathfrak{F}, \{\mathfrak{F}_k\}_k, \mathrm{P})$. In (6.2) functions $\ell(\cdot)$ and V_f are extended real valued functions which, as we are about to discuss, can be used to encode hard and/or soft constraints, so this formulation is quite general.

We assume that in (6.2) the cost function ℓ is written as $\ell(x, u, \xi) = \phi(x, u, \xi) + \bar{\phi}(F_\xi x + G_\xi u, \xi)$, where ϕ is real-valued, smooth in (x, u), and strongly convex over the affine space that defines the system dynamics, while $\bar{\phi}$ is an extended real valued function, lower semicontinuous, proper, convex and possibly nonsmooth. Likewise, V_f can be decomposed as $V_f(x, \xi) = \phi_N(x, \xi) + \bar{\phi}_N(F_{N,\xi} x, \xi)$.

Function $\bar{\phi}$ can be chosen to be any nonsmooth function as dictated by the problem it need to solve. The function $\bar{\phi}$ is used to encode arbitrary hard constraints on states and inputs of the form $F_{\xi_k} x_k + G_{\xi_k} u_k \in Y_{\xi_k}$ by choosing

$$\bar{\phi}(\cdot, \xi_k) = \delta(\cdot | Y_{\xi_k}), \tag{6.3}$$

[1] In some applications we may assume that $u_k \lhd \mathfrak{F}_k$, i.e., u_k is decided as a function of p and all $\xi_0, \ldots, \xi_k$.

where Y_{ξ_k} are non-empty convex closed sets for which projections $\mathrm{proj}(\cdot|Y_{\xi_k})$ can be easily computed. Soft constraints can be encoded by choosing

$$\bar{\phi}(\cdot, \xi_k) = \eta_{\xi_k} d(\cdot|Y_{\xi_k}), \tag{6.4}$$

where $\eta_{\xi_k} > 0$ is a scaling factor, while one may also choose

$$\bar{\phi}(\cdot, \xi_k) = \|\cdot\|_1 \tag{6.5}$$

to force the optimiser to be sparse.

The smooth part of the stage cost ℓ is a quadratic function of the form $\phi(x_k, u_k, \xi_k) = x_k' Q_{\xi_k} x_k + u_k' R_{\xi_k} u_k$, where $R_{\xi_k} \in \mathbb{S}^{n_u}_{++}$ and $Q_{\xi_k} \mathbb{S}^{n_x}_{+}$. The smooth part of the terminal cost function V_f is a quadratic function $\phi_N(x_N, \xi_N) = x_N' P_{\xi_N} x_N$, with $P_{\xi_N} \in \mathbb{S}^{n_x}_{++}$. The function $\bar{\phi}_N$ can be selected in the same way as explained for $\bar{\phi}$, *e.g.*, terminal constraints of the form $F_{N,\xi} x_N \in X_f$ can be encoded using $\bar{\phi}_N(\cdot, \xi) = \delta(\cdot|X_f)$, where X_f is assumed to be such that $\mathrm{proj}(\cdot|X_f)$ can be easily evaluated computationally.

6.2.2 Scenario-based formulation

The scenario-based formulation of (6.2) accrues from the assumption that $\mathfrak{F}_{N-1}$ is finite and produces the scenario tree structure shown in Fig. 29. A scenario tree describes the probable evolution of the state sequence $\{x_k\}_{k \in \mathbb{N}_{[0,N]}}$. The elementary events $\{\xi^i_{N-1}\}_{i \in \mathbb{N}_{[1,\mu]}}$ identify a set of *final outcomes* which correspond to the *leaf nodes* of the scenario tree. In turn, each leaf node identifies a single scenario, i.e., a sequence of realisations of the random process $\{\xi_k\}_{k \in \mathbb{N}_{[0,N-1]}}$. The tree is partitioned in N stages. The observable scenarios at stage k are the *nodes* of the tree at that stage; the number of nodes at stage k is denoted by μ_k. The probability that at stage k, the scenario ξ^i_{N-1} happens is denoted by p^i_k.

As shown in Figure 29, at stage $k \in \mathbb{N}_{[0,N-1]}$ the i-th node defines a set of *children nodes* at stage $k+1$ denoted by $\mathrm{child}(k,i) \subseteq \mathbb{N}_{[1,mu_{k+1}]}$. Every node at stage $k \in \mathbb{N}_{[1,N]}$ has a unique ancestor node at stage $k-1$ denoted by $\mathrm{anc}(k,i) \in \mathbb{N}_{[1,\mu_{k-1}]}$.

The system dynamics along scenarios can be written as

$$x^j_{k+1} = A^j_k x^i_k + B^j_k u^i_k + w^j_k, \tag{6.6}$$

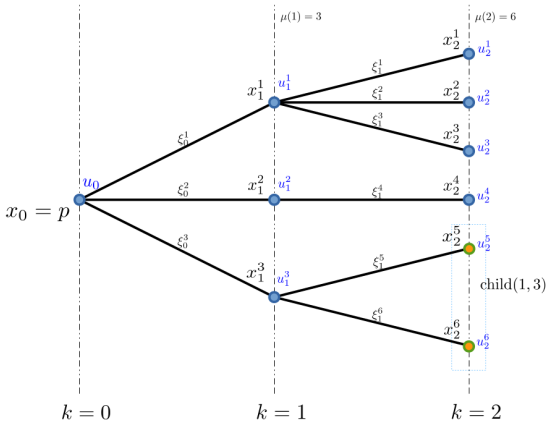

Figure 29: Scenario tree structure describing the evolution of the state.

with $i = \text{anc}(k+1, j)$, where $A_k^j = A_{\xi_k^j}$, $B_k^j = B_{\xi_k^j}$ and $w_k^j = w_{\xi_k^j}$.

Let $\mathbf{x}$ be a vector comprising all x_k^i and u_k^i and let us define the dynamic as $\mathcal{X}(p) = \{\mathbf{x} | x_{k+1}^j = A_k^j x_k^i + B_k^j u_k^i + w_{k+1}^j, \forall k \in \mathbb{N}_{[0,N-1]}, i \in \mathbb{N}_{[1,\mu(k)]}, j \in \text{child}(i,k)\}$ be the linear space of all $\mathbf{x}$ satisfying for $x_0 = p$. Define

$$f(x) = \sum_{k=0}^{N-1} \sum_{i=1}^{\mu_k} p_k^i \phi(x_k^i, u_k^i, i) + \sum_{i=1}^{\mu_N} p_{N-1}^i \phi_N(x_N^i, i) + \delta(x | \mathcal{X}(p)),$$

$$g(Hx) = \sum_{k=0}^{N-1} \sum_{i=1}^{\mu_k} p_k^i \bar{\phi}_k^i (F_k^i x_k^i + G_k^i u_k^i, i) + \sum_{i=1}^{\mu_N} p_{N-1}^i \bar{\phi}_N^i (F_N^i x_N^i, i).$$

Then the problem (6.2) can be written as

$$P^\star = \min_x f(x) + g(Hx), \tag{6.8}$$

where H is a linear operator with $z = Hx$ with $z_k^i = F_k^i x_k^i + G_k^i u_k^i$ for $k \in \mathbb{N}_{[0,N-1]}$, $i \in \mathbb{N}_{[0,\mu_k]}$ and $z_N^i = F_N^i x_N^i$. The Fenchel dual of (6.8) is

$$D^\star = \min_y f^*(-H'y) + g^*(y). \tag{6.9}$$

Under the prescribed assumptions, strong duality holds, i.e., $P^\star = D^\star$ and f^* is differentiable with L-Lipschitz gradient because of (RW09, Prop. 12.60).

For notational convenience we define $f_\circ(y) := f^*(-H^\top y)$, thus $\nabla f_\circ(y) = -H\nabla f^*(-H^\top y)$.

6.3 Optimisation algorithm

In this section, we define the Forward-Backward Envelope (FBE) function and its relationship with forward-backward algorithm.

6.3.1 Forward-backward envelope (FBE) function

The proximal operator of a proper, closed, convex function g play a major role in modern optimisation theory and is defined as

$$\operatorname{prox}_{\lambda g}(v) = \operatorname*{argmin}_z \{g(z) + \tfrac{1}{2\lambda}\|v - z\|^2\}. \tag{6.10}$$

The corresponding value function is the *Moreau envelope* of g, that is

$$g^\lambda(v) = \min_z \{f(z) + \tfrac{1}{2\lambda}\|v - z\|^2\}. \tag{6.11}$$

Let us denote the function to be minimised in (6.9) as $\varphi(y) = f^*(-H'y) + g^*(y)$. Now the proximal gradient algorithm also know as forward-backward splitting algorithm to solve (6.9) iteratively as:

$$y^{\nu+1} = \operatorname{prox}_{\lambda g^*}(y^\nu - \lambda\nabla f_\circ(y)), \tag{6.12a}$$

for $\lambda \in (0, 2/L)$ with L the Lipschitz of f^*. This iterate is equivalent to minimising:

$$y^{\nu+1} = \operatorname*{argmin}_{u\in\mathbb{R}^n} \{f_\circ(y^\nu) + \langle y^\nu - u, \nabla f_\circ(y^\nu)\rangle + g^*(y^\nu) + \tfrac{1}{2\lambda}\|y^\nu - u\|^2\} \tag{6.12b}$$

The optimality condition is

$$y - \operatorname{prox}_{\lambda g^*}(y - \lambda\nabla f_\circ(y)) = 0, \tag{6.13}$$

By virtue of the Moreau decomposition formula, (6.13) is equivalently written as

$$\nabla f_\circ(y) + \operatorname{prox}_{\lambda^{-1}g}(\lambda^{-1}y - \nabla f_0(y)) = 0. \tag{6.14}$$

We define the *forward-backward mapping*

$$T_\lambda(y) := \operatorname{prox}_{\lambda g^*}(y - \lambda\nabla f_\circ(y)), \tag{6.15}$$

which using the Moreau decomposition property becomes

$$T_\lambda(y) = y - \lambda\nabla f_\circ(y) - \lambda \operatorname{prox}_{\lambda^{-1}g}(\lambda^{-1}y - \nabla f_\circ(y)), \tag{6.16}$$

and the *fixed point residual* mapping

$$R_\lambda(y) := \lambda^{-1}(y - T_\lambda(y)). \tag{6.17}$$

The *forward-backward envelope* (FBE) of (6.2) is a real-valued function φ_λ given by (STP16; PB13)

$$\varphi_\lambda(y) = f_\circ(y) + g^*(T_\lambda(y)) - \lambda\left\langle \nabla f_\circ(y), R_\lambda(y)\right\rangle + \tfrac{\lambda}{2}\|R_\lambda(y)\|^2, \tag{6.18a}$$

This is the value function of the forward-backward algorithm (6.12b)

$$\varphi_\lambda(y) = \min_{u\in\mathbb{R}^n}\{f_\circ(y) + \langle y-u, \nabla f_\circ(y)\rangle + g^*(y) + \tfrac{1}{2\lambda}\|y-u\|^2\} \tag{6.18b}$$

This function φ_λ is always real-valued function irrespective of original function φ.

Assuming that $f_\circ$ is twice continuously differentiable we have φ_λ as continuously differentiable with Lipschitz-continuous gradient given by

$$\nabla\varphi_\lambda(y) = (I - \lambda\nabla^2 f_\circ(y))R_\lambda(y). \tag{6.19a}$$

By rearranging these terms we get

$$T_\lambda(y) = y + \lambda(I - \lambda\nabla^2 f_\circ(y)^{-1}\nabla\varphi_\lambda(y), \tag{6.19b}$$

For $\lambda \in (0, 1/L)$, the matrix $(I - \lambda\nabla^2 f_\circ(y)$ will be positive definite symmetric matrix and we can notice that the proximal-gradient iterate to minimise φ is equivalent to variable-metric iterate on φ_λ. The set

of minimizers of (6.2) coincides with $\operatorname{argmin} \varphi_\lambda \equiv \operatorname{zer} \nabla\varphi_\lambda := \{y : \nabla\varphi_\lambda(y) = 0\}$, although φ_λ is not always convex. However, if $f_\circ$ is convex quadratic, then φ_λ is also convex.

We should highlight here that the value and gradient of the FBE are computed at the computational cost of a forward-backward step. Moreover, for the evaluation of $\nabla\varphi_\lambda(y)$ it suffices to have a way to compute products $\nabla^2 f^*(y) \cdot d$. In case a closed-form formula is not available, such products can be evaluated by some numerical approximation method.

6.3.2 Computation of the dual gradient

The efficient computation of the the dual gradient is of crucial importance for the performance of the algorithm we are about to describe. By virtue of the *conjugate-subgradient* theorem (Roc72, Thm. 23.5), it is

$$\nabla f^*(-H^\top y) = \operatorname*{argmin}_z \{\left\langle z, H^\top y \right\rangle + f(z)\}. \tag{6.20}$$

This equality-constrained optimisation problem can be solved by dynamic programming when ϕ_k using the following algorithm (SSBP15), where Φ_k^i, Θ_k^i, D_k^i, Λ_k^i, K_k^i and σ_k^i, c_k^i are computed once offline following Algorithm 4.

6.3.3 Computation of the dual Hessian

The computation of $\nabla\varphi_\lambda(y)$ requires the computation of products of the form $H_f(d) := \nabla^2 f_\circ \cdot d$. Notice that to a great extent the computations in Alg. 9 can be parallelised. The dual Hessian is then used for the computation of $\nabla\varphi_\lambda$.

6.3.4 Computation of $\nabla\varphi_\lambda$

The gradient of the FBE $\nabla\varphi_\lambda(y)$ is computed as

$$\nabla\varphi_\lambda(y) = \lambda^{-1} t(y) - H_f(t(y)), \tag{6.21}$$

Algorithm 8 Dual gradient computation

$q_N^i \leftarrow y_N^i, \forall i \in \mathbb{N}_{[1,\mu(N)]}$, `%Backward substitution`
for $k = N-1, \ldots, 0$ **do**
  for $i \in \mu(k)$ **do** {in parallel}
    $u_k^i \leftarrow \Phi_k^i y_k^i + \sum_{j \in \text{child}(k,i)} \Theta_k^j q_{k+1}^j + \sigma_k^i$
    $q_k^i \leftarrow D_k^{i\prime} y_k^i + \sum_{j \in \text{child}(k,i)} \Lambda_k^{j\prime} q_{k+1}^j + c_k^i$
  end for
end for
$x_0^1 = p$, `%Forward substitution`
for $k = 0, \ldots, N-1$ **do**
  for $i \in \mu(k)$ **do** {in parallel}
    $u_k^i \leftarrow K_k^i x_k^i + u_k^i$
    for $j \in \text{child}(k,i)$ **do** {in parallel}
      $x_{k+1}^j \leftarrow A_k^j x_k^i + B_k^j u_k^i + w_k^j$
    end for
  end for
end for

where $H_f(t(y))$ is the dual Hessian along the direction $t(y)$ (see Alg. 9, Sec. 6.3.3), and $t(y)$ is given by

$$x(y) = \underset{z}{\text{argmin}}\{\left\langle z, H^\top y \right\rangle + f(z)\}, \tag{6.22a}$$

$$z(y) = \text{prox}_{\lambda^{-1} g}\{\lambda^{-1} y + Hx(y)\}, \tag{6.22b}$$

$$t(y) = z(y) - Hx(y), \tag{6.22c}$$

where $x(y) = \nabla f^*(-H^\top y)$ is computed as explained in Section 6.3.2 and $z(y)$ is a prox-step which typically consists in simple element-wise operations which can be fully parallelised.

6.3.5 L-BFGS method

L-BFGS – limited memory BFGS method – is a computationally less intensive quasi-Newton method suitable for large-scale optimisation

Algorithm 9 Computation of Hessian-vector products

Require: Vector d
  $\hat{q}_N^i \leftarrow d_N^i, \forall i \in \mathbb{N}_{[1,\mu_N]}$
  for $k = N-1, \ldots, 0$ **do**
    for $i = 1, \ldots, \mu_k$ **do** {in parallel}
      $\hat{u}_k^i \leftarrow \Phi_k^i d_k^i + \sum_{j \in \text{child}(k,i)} \Theta_k^j \hat{q}_{k+1}^j$
      $\hat{q}_k^i \leftarrow D_k^{i\top} d_k^i + \sum_{j \in \text{child}(k,i)} \Lambda_k^{j\top} \hat{q}_{k+1}^j$
    end for
  end for
  $\hat{x}_0^1 = 0$
  for $k = 0, \ldots, N-1$ **do**
    for $i = 1, \ldots, \mu_k$ **do** {in parallel}
      $u_k^i \leftarrow K_k^i \hat{x}_k^i + \hat{u}_k^i$
      for $j \in \text{child}(k,i)$ **do** {in parallel}
        $\hat{x}_{k+1}^j \leftarrow A_k^j \hat{x}_k^i + B_k^j \hat{u}_k^i$
      end for
    end for
  end for

problems. The iterate with BFGS-method is:

$$x^{\nu+1} = x^\nu + \tau B^\nu \nabla f(x^\nu),$$

where B^ν is the approximate of the inverse Hessian, τ is the step-size and $\nabla f(x^\nu)$ is the gradient. The inverse Hessian B^ν is updated is given as:

$$\begin{aligned} B^\nu = (I - \rho_{\nu-1} y_{\nu-1} s'_{\nu-1})' B^{\nu-1} (I - \rho_{\nu-1} y_{\nu-1} s'_{\nu-1}) \\ + \rho_{\nu-1} s_{\nu-1} s'_{\nu-1} \end{aligned} \tag{6.23a}$$

where

$$\rho_{\nu-1} = \frac{1}{y'_{\nu-1} s_{\nu-1}} \quad , \tag{6.23b}$$

$$y_{\nu-1} = \nabla f(x^\nu) - \nabla f(x^{\nu-1}), \tag{6.23c}$$

$$s_{\nu-1} = x^\nu - x^{\nu-1} \tag{6.23d}$$

It can be observed that the inverse Hessian approximation $B^{\nu-1}$ is a dense matrix and storing and updating is quadratic complexity in both computational and memory requirement. Instead L-BFGS method stores information from the past m iterations and uses only this information to implicitly do operations requiring the inverse Hessian – in particular computing the next search direction, $B^{\nu}\nabla f(x^{\nu})$. In L-BFGS, a buffer of size m is maintained for y_{ν} and s_{ν}. For the first m iterates the bufferer is filled and once it is completely filled, the oldest y_{ν}, s_{ν} are replaced at each iterate. This calculation of search direction with L-BFGS method is summarised in two-loop recursion (NW06, Alg. 7.4) given below as:

Algorithm 10 Two-loop recursion

Require: $d \leftarrow -\nabla f(x^{\nu})$
  for $i = \nu - 1, \ldots, \nu - m$ **do**
    $\alpha_i \leftarrow \rho_i s_i' d$;
    $d \leftarrow d - \alpha_i y_i$;
  end for
  $d \leftarrow B^{\nu} d$
  for $i = \nu - m, \ldots, \nu - 1$ **do**
    $\beta \leftarrow \rho_i y_i' d$
    $d \leftarrow d + s_i(\alpha_i - \beta)$
  end for
  return search direction $d = B^{\nu}\nabla f(x^{\nu})$

The computation of d^{ν} requires only $4mn_d$ multiplications, where m is the memory length of the L-BFGS buffer and n_d is the dimension of d and when $m << n_d$, the complexity is $\mathcal{O}(n_d)$.

6.3.6 L-BFGS method for the FBE

In the above section, we described how the gradient of the FBE can be computed. The idea is to use past gradient information to approximate the Hessian using L-BFGS method and calculate the decent direction. The Algorithm 11 summarises the basic steps of the proposed algorithm. At every iteration, an L-BFGS direction d^{ν} is computed using the two-loop recursion, Alg. 10 , that is, in line 3 of Alg. 11 the

matrix B^ν — which is an approximation of the inverse Hessian when this exists — does not need to be constructed or stored.

This step involves the computation of the gradient of the FBE at y^ν which is performed as discussed in Section 6.3.4.

The dual vector y^ν is updated as in line 4 where τ_ν is chosen so as to satisfy the Wolfe conditions (NW06, Sec. 3.1):

$$\varphi_\lambda(y^{\nu+1}) \leq \varphi_\lambda(y^\nu) + c_1\tau_\nu \left\langle \nabla\varphi_\lambda(y^\nu), d^\nu \right\rangle \tag{6.24a}$$

$$\left\langle \nabla\varphi_\lambda(y^{\nu+1}), d^\nu \right\rangle \geq c_2 \left\langle \nabla\varphi_\lambda(y^\nu), d^\nu \right\rangle, \tag{6.24b}$$

where $0 < c_1 < c_2 < 1$. The first inequality is a sufficient decrease condition, while the second one is known as the *curvature condition* and is used to rule out unacceptably short step lengths.

Algorithm 11 Forward-Backward L-BFGS

Require: $\lambda \in (0, 1/L)$, y^0, m (memory), ϵ (tolerance), $\nu_{\max}$ (maximum iterations)

1: Initialise an L-BFGS buffer of length m
2: **while** $\|R_\lambda(y^\nu)\| > \epsilon\|R_\lambda(y^0)\|$ and $\nu \leq \nu_{\max}$ **do**
3: $d^\nu \leftarrow -B^\nu\nabla\varphi_\lambda(y^\nu)$ (L-BFGS direction)
4: $y^{\nu+1} \leftarrow y^\nu + \tau_\nu d^\nu$ where τ_k satisfies (6.24)
5: $s^\nu \leftarrow y^{\nu+1} - y^\nu$, $q^\nu \leftarrow \nabla\varphi_\lambda(y^{\nu+1}) - \nabla\varphi_\lambda(y^\nu)$
$\rho_\nu \leftarrow 1/\langle s^\nu, q^\nu\rangle$
6: **if** $\rho_\nu > 0$ **then**
7: Push (s^ν, q^ν, ρ_ν) in the L-BFGS buffer
8: **end if**
9: $\nu \leftarrow \nu + 1$
10: **end while**

The algorithm produces a sequence y^ν which converges to the dual optimal solution $y^\star$ satisfying (6.13), while the sequence $x^\nu = x(y^\nu)$ as in (6.22a) converges to a primal optimal solution $x^\star$.

Typically, in quasi-Newton methods for the Hessian approximations to be positive definite, the Wolfe conditions are used to determine the step-size τ_ν (LO13) and an inexact line search is used to compute an appropriate step size as in (NW06, Alg. 3.5).

The algorithm is terminated once the fixed-point residual becomes adequately small; we use the termination condition

$$\|R_\lambda(y^\nu)\| > \epsilon \|R_\lambda(y^0)\|.$$

6.4 Simulations

We compare the performance of Algorithm 11 with different L-BFGS buffers against the dual APG of Chapter 3 to solve the stochastic model predictive control problem for a linear discrete-time system with additive and parametric uncertainty. We consider a system of m aligned interconnected masses by $m-1$ linear spring-dampers of stiffness constant $\kappa = 1$ and damping ratio $\beta = 0.1$. The manipulated variables are the forces we may exercise on each spring along their principal axes and the state variables are the positions and speeds of the masses. We assume that the system dynamics is obtained by discretising the continuous-time dynamics with sampling time $T_s = 0.5$ and is written as in (6.1) with $n_x = 2m$, and $n_u = m - 1$. On the system state and input variables we impose the constraints $-5 \leq x_k^i \leq 5$ and $-1 \leq u_k^i \leq 1$ for all $k \in \mathbb{N}_{[0,1]}$ and $i \in \mathbb{N}_{[1,\mu(k)]}$. The stage cost was chosen to be $\ell(x, u, \xi) = x'Qx + u'Ru$ with $Q = I_{n_x}$ and $R = I_{n_u}$.

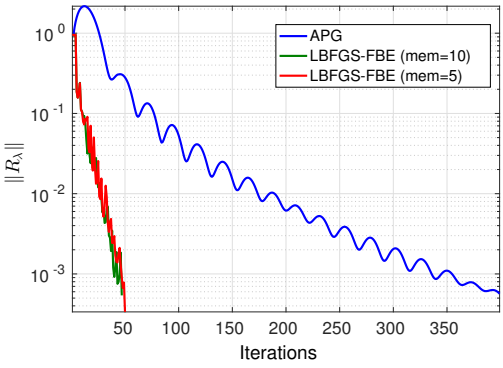

Figure 30: Convergence of the proposed algorithm.

The dependence of iterations on size of scenario; the trees is considered in this experiment had a fixed horizon $N = 10$ and in their first stages were binary, i.e., had branching factor 2 and eventually evolved without branching until the end of the horizon and for the $N = 10$, we have 1024 scenarios. The algorithm terminates when the fixed-point residual satisfies the condition $\|R_\lambda(y^\nu)\| \leq 5.10^{-4} \times \min(1, \|R_\lambda(y^0)\|)$. The simulations were performed on a $4 \times 2.60\,\mathrm{GHz}$ Intel i5 machine with $8\,\mathrm{GB}$ RAM running 64-bit Ubuntu 14.04.

As a rule of thumb, the L-BFGS memory is chosen in the range 3 and 20. We considered two buffer sizes — 5 and 10. We observed no significant difference in the convergence rate in these choice of buffer sizes which is also shown in Figure 31. It can be observed that using L-BFGS direction reduces dramatically the number of iterations compared to APG. We can also observe that the convergence with L-BFGS method is faster and leads faster to higher precision solutions compared to APG.

6.5 Conclusions

In this chapter, we constructed the smooth forward-backward envelope function for the stochastic optimal control problem and solve it using L-BFGS method. The main characteristics of this algorithm is fast convergence and high parallelisability. The computation of the gradient at every iteration of the algorithm can be parallelized to offer a significant benefit in terms of speed-up. In particular computations are executed in parallel across all nodes at every stage of the scenario tree. Future work include implementing the L-BFGS-FBE algorithm in hardware like GPU that can perform parallel computations. Also implement the global-FBE and accelerated global-FBE algorithms (STP16) that have simpler line-search compared to L-BFGS-FBE.

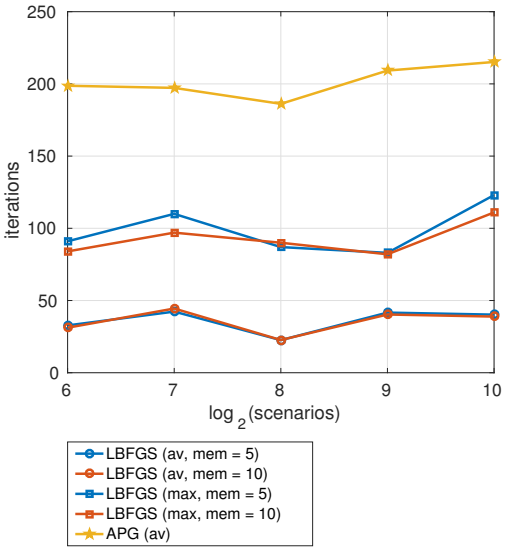

Figure 31: Iterations required for the proposed algorithm to converge.

Chapter 7

Conclusions and future work

Drinking Water Networks are critical infrastructures and an integral part of the urban life. Managing these networks is crucial to counter water security problem and over-exhaustion of the water resources. This network is complex in dynamics, large-scale in topology and an energy-intensive system that is subjected to uncertainty demand. An optimal management policy should reduce the energy consumption of pumping and cater the water demands of the consumers uninterrupted. In this thesis we achieve this objective by proposing a model predictive control based technique for management of these networks.

In this chapter, we provide a summary of the thesis highlighting the main contribution of the work. Then we propose some research direction to extend the work.

7.1 Main contributions of the thesis

This thesis addresses several open problem in the control of drinking water networks and in stochastic optimal control in general. In particular

1. We demonstrated the effectiveness of MPC in Chapter 2 combining forecasting models for the upcoming water demands and accounting for the worst-case prediction errors by shrinking the system constraints. This work lead to the publication (SGS+14).
2. In Chapter 3 we proposed a dual accelerated proximal gradient method for multistage stochastic optimal control problems which can be massively parallelised across all nodes at each stage of the scenario tree and solved on general purpose GPUs. Simulations demonstrate high speed-ups and fast convergence. These results appeared in (SSBP15).
3. The proposed parallelisable algorithm was tailored to the solution of stochastic optimal control problems related to the control of drinking water networks. In Chapter 4 we provide evidence of computational tractability showing that the proposed method is suitable for the control of DWNs and we further close the loop with the proposed stochastic MPC controller and report on the performance of the controlled system. These results have been provisionally accepted for publication in IEEE Control Systems Technology and a preprint is available on arXiv (SSBP16).
4. In Chapter 5 we presented a framework for multi-time-scale hierarchical and decentralised control where a lower control layer stabilises the local systems and an upper control layer commands set-points at a lower rate in such a way that the state-input constraints are not violated. These results appeared in (SSB16).
5. In Chapter 6 we present a methodology which we developed recently and leads to further acceleration for the fast solution of stochastic optimal control problems applying the LBFGS algorithm on a smooth merit function of dual optimisation problem — the forward-backward envelope. We show

7.2 Future research directions

In this thesis we proposed stochastic MPC based controller integrated with demand forecasts for management of drinking water networks. We also proposed a parallelisable algorithm to solve the large-scale optimisation problem that is associated with this formulation. There are many open issues which could be a future direction for research:

- **Quasi-Newton method applied for the control of DWN**. In Chapter 6 we proposed a quasi-Newton method to solve the stochastic optimisation problem and demonstrated faster convergence than APG. This method also exploit the structure and amenable to parallelisation. The next step is to implement this algorithm on a GPU and subsequently apply it to solve the scenario-based stochastic MPC problem for DWNs.
- **Novel parallelisable optimisation algorithms**. Several novel optimisation algorithms have recently emerged based on the forward-backward envelope function demonstrating high convergence rates and precision (STP16; TSP16). These methods are amenable to parallelisation and can be used to solve large-scale scenario-based stochastic optimal control problems.
- **Risk-averse stochastic optimal control.** In this thesis the randomness in the cost function is quantified with the expectation operator leading to a so-called *risk-neutral* formulation. In a pioneering paper Artzner *et al.* (ADEH99) developed an axiomatic framework for risk measures[1] which turned out to be suitable for the formulation of risk-averse multistage optimal control problems (SDR09; Sha09; GR11; Sha12) and have been used in various applications such as power systems (ZG13) and finance (AMRU01). When these measures are introduced in stochastic programming models, this lead to lead to a new class of problems –*risk-averse optimisation* (Sha09; GR11; Sha12).

  In two recent papers Asamov and Ruszczyński (AR15) and Collado *et al.* (CPR12) proposed methods to solve stochastic *linear* problems which however do not scale up well with the number of scenarios and the prediction horizon. Risk-averse problems, however, are significantly more complex than their risk-neutral counterparts and there are no algorithms to solve them fast enough for them to qualify for typical control applications — even for medium-scale problems. Developing optimisation methods for risk-averse optimisation problems in control applications is an interesting research direction.

[1] Artzner *et al.* postulated four regularity assumptions for risk measures $\rho : \mathcal{L}_p(\Omega, \mathfrak{F}, \mathrm{P}) \to \mathbb{R}$: (i) sub-additivity, (ii) positive homogeneity, (iii) monotonicity and (iv) translation invariance. These turn out to be *desiderata* for the formulation of a sound risk-averse optimal control problem.

- **Distributionally robust control.** In practice, the actual probability distribution — the probability measure P — of the involved stochastic process(es) is not known but it is rather estimated from measurements. What is the same, this probability measure may be time-varying. Then it is assumed that P is contained in a set of probability measures (DY10; PDHM14). Van Parys *et al.* formulated a control problem which accommodates the worst-case probability distribution for linear stochastic systems (PKGM16). In that work as well as in the framework presented in (ZKR13) we may further include first- and second-order information. It is an open problem though how to address the problem of distributionally robust control in the context of scenario-based multistage optimal control.
- **Stochastic economic MPC.** Often, the cost function of an MPC problem is chosen to reflect certain *economic* requirements or otherwise *requirements* for the operation of the problem rather than to enforce (robust) stability or other control theoretic properties for the closed-loop system. In such cases we are interested in (i) the performance of the closed loop — we ask: does the closed-loop operation eventually lead to the desired *economic operation* and (ii) the properties of the controlled system — are the constraints satisfied? is the system stable? This discourse has motivated the emergence of a branch of MPC known as *economic MPC* (DAR11; AAR12; MAA13; SSE$^+$14; EDC14). However, with the exception of (CL15), the performance of economic MPC under uncertainty is little understood. The economic MPC concept has been employed for the control of drinking water networks in (GOMP$^+$14)[2], however, without considering the inevitable inherent uncertainty.
- **Multi-rate control of DWN.** In the Chapter 5 we developed the theory for designing a multi-rate hierarchical decentralised controller and tested for Johansson's system. Now designing these decentralised controllers for a complete DWN is another extension which would have direct industrial impact.
- **Pressure model for DWN.** In this thesis we considered a flow-based model for the operation of DWN which lead to a linear

[2]In (GOMP$^+$14) the authors propose an MPC formulation with a periodic constraint and an terminal equality constraint.

dynamical model and polyhedral (box) constraints. This model neglects the pressure drops and including them would result in a nonconvex problem formulation (SKN$^+$15). The zero-FPR algorithm (TSP16) mentioned before is suitable for nonconvex problems and makes a possible research direction. On the other hand, an approximation method of such nonlinear constraints has been proposed in the literature using a separate *constraints satisfaction problem* (CSP) which can be employed to deliver a complete solution for the control of DWNs combining CSP with GPU-accelerated scenario-based stochastic MPC (CCPC14).

- **Other applications.** The developments presented in this thesis allow the application of stochastic MPC to adjacent fields of study such as power dispatch in micro-grids and management of smart-grids (PTB11; ZH14; HSB$^+$15), sewer networks (JDOMC14), inventory control (SM16) and more. Especially in management of smart-grids, there is uncertainty from consumer demand and in generation from renewable sources. The response time with smart-grids is in the order of few minutes which makes these networks challenging. Extending the quasi-Newton based methods for SMPC to manage these systems would be an interesting problems with a lot of industrial relevance.

References

[AAR12] D. Angeli, R. Amrit, and J. B. Rawlings. On average performance and stability of economic model predictive control. *IEEE Transactions on Automatic Control*, 57(7):1615–1626, July 2012. 114, 151

[ABB11] Alessandro Alessio, Davide Barcelli, and Alberto Bemporad. Decentralized model predictive control of dynamically coupled linear systems. *Journal of Process Control*, 21(5):705 – 714, 2011. Special Issue on Hierarchical and Distributed Model Predictive Control. 116

[ABCJC06] H. Algre, J. Baptista, E.C. Cabrera Jr, and F. Cubillo. *Performance indicators for water supply services*. Manuals of best practice series. IWA Publishing, London, 2006. 108

[ADEH99] Philippe Artzner, Freddy Delbaen, Jean-Marc Eber, and David Heath. Coherent measures of risk. *Mathematical Finance*, 9(3):203–228, 1999. 150

[ALdlP+11] I. Alvarado, D. Limon, D. Muoz de la Pea, J.M. Maestre, M.A. Ridao, H. Scheu, W. Marquardt, R.R. Negenborn, B. De Schutter, F. Valencia, and J. Espinosa. A comparative analysis of distributed MPC techniques applied to the hd-mpc four-tank benchmark. *Journal of Process Control*, 21(5):800 – 815, 2011. Special Issue on Hierarchical and Distributed Model Predictive Control. 124, 127

[AMRU01] Fredrik Andersson, Helmut Mausser, Dan Rosen, and Stanislav Uryasev. Credit risk optimization with conditional value-at-risk criterion. *Math. Program.*, 89(2):273–291, 2001. 150

[AR15] Tsvetan Asamov and Andrzej Ruszczyński. Time-consistent approximations of risk-averse multistage stochastic optimization problems. *Mathematical Programming*, 153(2):459–493, 2015. 150

[Bak08] Lubomr Bakule. Decentralized control: An overview. *Annual Reviews in Control*, 32(1):87 – 98, 2008. 114, 115

[BB07] D. Bertsimas and D.B. Brown. Constrained stochastic LQC: A tractable approach. *Automatic Control, IEEE Transactions on*, 52(10):1826–1841, Oct 2007. 86

[BB09] Davide Barcelli and Alberto Bemporad. Decentralized model predictive control of dynamically-coupled linear systems: Tracking under packet loss. *IFAC Proceedings Volumes*, 42(20):204 – 209, 2009. 116

[BB12] D. Bernardini and A. Bemporad. stabilizing model predictive control of stochastic constrained linear systems. *Automatic Control, IEEE Transactions on*, 57(6):1468–1480, June 2012. 86

[BBB10] D. Barcelli, D. Bernardini, and A. Bemporad. Synthesis of networked switching linear decentralized controllers. In *49th IEEE Conference on Decision and Control (CDC)*, pages 2480–2485, Dec 2010. 119

[BBMJ+13] A.M. Bagirov, A.F. Barton, H. Mala-Jetmarova, A. Al Nuaimat, S.T. Ahmed, N. Sultanova, and J. Yearwood. An algorithm for minimization of pumping costs in water distribution systems using a novel approach to pump scheduling. *Mathematical and Computer Modelling*, 57(34):873 – 886, 2013. 12, 85

[BBR10] D. Barcelliy, A. Bemporad, and G. Ripaccioliy. Hierarchical multi-rate control design for constrained linear systems. In *49th IEEE Conference on Decision and Control (CDC)*, pages 5216–5221, Dec 2010. 116

[BC05] Diana Barro and Elio Canestrelli. Time and nodal decomposition with implicit non-anticipativity constraints in dynamic portfolio optimization. Ge, growth, math methods, EconWPA, 2005. 67

[BC11] Heinz H. Bauschke and Patrick L. Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. Springer Publishing Company, Incorporated, 1st edition, 2011. 55

[BCM97] A. Bemporad, A. Casavola, and E. Mosca. Nonlinear control of constrained linear systems via predictive reference management. *IEEE Transactions on Automatic Control*, AC-42(3):340–349, 1997. 114

[BEFB94] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear matrix inequalities in system and control theory*, volume 15 of *Studies in Applied Mathematics*. SIAM, Philadelphia, PA, June 1994. 65

[Ber99] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, USA, 2nd edition, 1999. 65, 105

[Ber00] Dimitri P. Bertsekas. *Dynamic programming and optimal control*. Athena Scientific, 2nd edition, 2000. 66, 101

[BF12] Stephen Becker and Mohamed-Jalal Fadili. A quasi-newton proximal splitting method. In *NIPS*, pages 2627–2635, 2012. 133

[BFS13] G. Betti, M. Farina, and R. Scattolini. Decentralized predictive control for tracking constant references. In *52nd IEEE Conference on Decision and Control, IEEE*, pages 5228 – 5233, Firenze, Italy, December 2013. 115

[BJR94] G. E. P. Box, G. M. Jenkins, and G. C. Reisnel. *Time series analysis, forecasting and control*. Prentice-Hall International, Inc., 1994. 17

[BL02] Jrgen Blomvall and Per Olov Lindberg. A riccati-based primal interior point solver for multistage stochastic programming. *European Journal of Operational Research*, 143(2):452 – 461, 2002. 106

[BLNZ95] R.H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput.*, 16(5):1190–1208, sep 1995. 133

[BMLL+04] C. Biscos, M.V. Mulholland, M.V. Le Lann, C.A. Buckley, and C.J. Broukaert. Optimal operation of water distribution networks by predictive control using MINLP. *Water SA*, 29(4):393–404, 2004. 13

[BMTVG12] R. Benenson, M. Mathias, R. Timofte, and L. Van Gool. Pedestrian detection at 100 frames per second. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2903–2910, June 2012. 87

[BOMPB10] Davide Barcelli, Carlos Ocampo-Martínez, Vicen Puig, and Alberto Bemporad. Decentralized model predictive control of drinking water networks using an automatic subsystem decomposition approach. *IFAC Proceedings Volumes*, 43(8):572 – 577, 2010. 13, 26

[Boy09] S. Boyd. Linear dynamical systems, Winter Quarter 2008–2009. EE363 lecture notes. 71

[Bra10] A. Bradley. *Algorithms for equilibration of matrices and their application to limited-memory quasi-Newton methods*. PhD thesis, Stanford University, 2010. 65, 104

[Bro70] C. G. Broyden. The convergence of a class of double-rank minimization algorithms: 2. the new algorithm. *IMA Journal of Applied Mathematics*, 6(3):222–231, 1970. 132

[BS78] D.P. Bertsekas and S.E. Shreve. *Stochastic Optimal Control: The Discrete-Time Case*. Athena Scientific, 1978. 46

[BT09a] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Img. Sci.*, 2(1):183–202, March 2009. 58

[BT09b] Amir Beck and Marc Teboulle. Gradient-based algorithms with applications to signal-recovery problems. In Daniel P. Palomar and Yonina C. Eldar, editors, *Convex Optimization in Signal Processing and Communications*, pages 42–88. Cambridge University Press, 2009. Cambridge Books Online. 59

[BU94] M. A. Brdys and B. Ulanicki. *Operational control of water systems: structures, algorithms and applications*. Prentice Hall International, 1994. 12, 15

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. 43

[BVP+13] M. Bakker, J. H. G. Vreeburg, L. J. Palmen, V. Sperber, G. Bakker, and L. C. Rietveld. Better water quality and higher energy efficiency by using model predictive flow control at water supply systems. *Journal of Water Supply: Research and Technology - Aqua*, 62(1):1–13, 2013. 86

[CBA07] Eduardo F. Camacho and Carlos Bordons Alba. *Model predictive control*. Springer-Verlag London, 2nd edition, 2007. 25

[CC06] G.C. Calafiore and M.C. Campi. The scenario approach to robust control design. *Automatic Control, IEEE Transactions on*, 51(5):742–753, May 2006. 86

[Ç11] Erhan Çinlar. *Probability and Stochastics*, volume 261 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1 edition, 2011. 42

[CCCJ11] C. Chih-Chung and L. Chih-Jen. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May 2011. 22

[CCPC14] S. Cong Cong, S. Puig, and G. Cembrano. Combining CSP and MPC for the operational control of water networks: application to the Richmond case study. In *19th IFAC World Congress*, pages 6246–6251, Cape Town, 2014. 91, 152

[CGP09] Marco C. Campi, Simone Garatti, and Maria Prandini. The scenario approach for systems and control design. *Annual Reviews in Control*, 33(2):149 – 157, 2009. 86

[CGT91] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. Convergence of quasi-newton matrices generated by the symmetric rank one update. *Mathematical Programming*, 50(1):177–195, 1991. 132

[CGT14] A. Casavola, E. Garone, and F. Tedesco. Improved feed-forward command governor strategies for constrained discrete-time linear systems. *IEEE Transactions on Automatic Control*, 59(1):216–223, 2014. 114

[CHL10] G. Chaloulos, P. Hokayem, and J. Lygeros. Distributed hierarchical mpc for conflict resolution in air traffic control. In *Proceedings of the 2010 American Control Conference*, pages 3945–3950, June 2010. 114

[CKW09] M. Cannon, B. Kouvaritakis, and Xingjian Wu. Probabilistic constrained MPC for multiplicative and additive stochastic uncertainty. *Automatic Control, IEEE Transactions on*, 54(7):1626–1632, July 2009. 49, 86

[CL15] D. Chatterjee and J. Lygeros. On stability and performance of stochastic predictive control techniques. *IEEE Transactions on Automatic Control*, 60(2):509–514, Feb 2015. 49, 151

[CLZ16] Yankai Cao, Carl D. Laird, and Victor M. Zavala. Clustering-based preconditioning for stochastic programs. *Computational Optimization and Applications*, 64(2):379–406, 2016. 104

[CMP04] A. Casavola, E. Mosca, and M. Papini. Control under constraints: an application of the command governor approach to an inverted pendulum. *IEEE Transactions on Control Systems Technology*, 12(1):193–204, 2004. 114

[Com11] Jean-Christophe Combettes, Patrick L.and Pesquet. *Proximal splitting methods in signal processing*, pages 185–212. Springer New York, New York, NY, 2011. 56

[Con09] G. A. Constantinides. Tutorial paper: Parallel architectures for model predictive control. In *Control Conference (ECC), 2009 European*, pages 138–143, Aug 2009. 39

[CP10] P.L Combettes and J-C. Pesquet. Proximal splitting methods in signal processing, 2010. arXiv:0912.3522v4. 96, 104

[CPR12] Ricardo A. Collado, Dávid Papp, and Andrzej Ruszczyński. Scenario decomposition of risk-averse multistage stochastic programming problems. *Annals of Operations Research*, 200(1):147–170, 2012. 150

[Cre98] J.D. Creasy. Pump scheduling in water supply: More than a mathematical problem. *Computer Applications in Water Supply*, 2:279–289, 1998. 85

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. 22

[CWZ14] W. Chen, Z. Wang, and J. Zhou. Large-scale L-BFGS using MapReduce. In *Advances in Neural Information Processing Systems 27*, pages 1332–1340. Curran Associates, Inc., 2014. 133

[DAR11] M. Diehl, R. Amrit, and J. B. Rawlings. A lyapunov function for economic optimizing model predictive control. *IEEE Transactions on Automatic Control*, 56(3):703–707, March 2011. 151

[DBB13] S. Di Cairano, M. Brand, and S. A. Bortoff. Projection-free parallel quadratic programming for linear model predictive control. *International Journal of Control*, 86(8):1367–1385, 2013. 39

[DGKR03] J. Dupačová, N. Gröwe-Kuska, and W. Römisch. Scenario reduction in stochastic programming. *Mathematical Programming*, 95(3):493–511, 2003. 94

[DLHS10] A. M. De Livera, R. J. Hyndman, and R. D. Snyder. Forecasting time series with complex seasonal patterns using exponential smoothing. Technical report, Monash University, Department of Econometrics and Business Statistics, Oct. 2010. 22

[DY10] Erick Delage and Yinyu Ye. Distributionally robust optimization under moment uncertainty with application to data-driven problems. *Operations Research*, 58(3):595–612, 2010. 151

[DZZ+12] A. Domahidi, A. Zgraggen, M.N. Zeilinger, M. Morari, and C.N. Jones. Efficient Interior Point Methods for Multistage Problems Arising in Receding Horizon Control. In *IEEE Conference on Decision and Control*, pages 668 – 674, Maui, HI, USA, December 2012. 106

[EDC14] Matthew Ellis, Helen Durand, and Panagiotis D. Christofides. A tutorial review of economic model predictive control methods. *Journal of Process Control*, 24(8):1156–1178, 2014. Economic nonlinear model predictive control. 151

[FBB+80] W. Findeisen, F.N. Bailey, M. Brdys, K. Malinowski, P. Tatjewski, and A. Wozniak. *Control and coordination in hierarchical systems*, volume 9. International Series on Applied Systems Analysis . John Wiley & Sons, Chichester, U.K., 1980. 114

[FKP+14] H.J. Ferreau, C. Kirches, A. Potschka, H.G. Bock, and M. Diehl. qpOASES: A parametric active-set algorithm for quadratic programming. *Math. Progr. Comp.*, 6(4):327–363, 2014. 79

[GÖ92] O. Güler. New proximal point algorithms for convex minimization. *SIAM J. Optim.*, 2(4):649–664, nov 1992. 63

[GB14] Pontus Giselsson and Stephen Boyd. Diagonal scaling in douglas-rachford splitting and ADMM. In *53rd IEEE Conference on Decision and Control*, pages 5033–5039, Los Angeles, CA, USA, 2014. 65

[GB15] Pontus Giselsson and Stephen Boyd. Metric selection in fast dual forwardbackward splitting. *Automatica*, 62:1 – 10, 2015. 65, 104

[GDP09] Alexander Guzhva, Sergey Dolenko, and Igor Persiantsev. *Artificial Neural Networks – ICANN 2009: 19th International Conference, Limassol, Cyprus, September 14-17, 2009, Proceedings, Part I*, chapter Multifold acceleration of neural network computations using GPU, pages 373–380. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. 87

[GK99] Elmer G. Gilbert and Ilya Kolmanovsky. Fast reference governors for systems with state and control constraints and disturbance inputs. *International Journal of Robust and Nonlinear Control*, 9(15):1117–1141, 1999. 114

[GMOMP14] J.M. Grosso, J.M. Maestre, C. Ocampo-Martínez, and V. Puig. On the assessment of tree-based and chance-constrained predictive control approaches applied to drinking water networks. In *19th IFAC Conference*, pages 6240–6245, Cape town, South Africa, August 2014. 85, 86, 108

[GN14] A.P. Goryashko and A.S. Nemirovski. Robust energy cost optimization of water distribution system with uncertain demand. *Automation and Remote Control*, 75(10):1754–1769, 2014. 86

[GNDJ14] Nicolai Fog Gade-Nielsen, Bernd Dammann, and John Bagterp Jrgensen. *Interior point methods on GPU with application to model predictive control*. PhD thesis, Technical University of Denmark, Lyngby, Denmark, 2014. 39

[GOMP13] J.M. Grosso, C. Ocampo-Martnez, and V. Puig. Learning-based tuning of supervisory model predictive control for drinking water networks. *Engineering Applications of Artificial Intelligence*, 26(7):1741–1750, 2013. 12

[GOMP+14] J. M. Grosso, C. Ocampo-Martínez, V. Puig, D. Limon, and M. Pereira. Economic mpc for the management of drinking water networks. In *Control Conference (ECC), 2014 European*, pages 790–795, June 2014. 151

[GOMPJ14] J.M. Grosso, C. Ocampo-Martnez, V. Puig, and B. Joseph. Chance-constrained model predictive control for drinking water networks. *Journal of Process Control*, 24(5):504 – 516, 2014. 17, 85, 86, 89, 91, 105

[GP11] L. Grüne and J. Pannek. *Nonlinear model predictive control*. Springer-Verlag London, 2nd edition, 2011. 25

[GR11] Ch Pflug Georg and Werner Römisch. *Measuring multi-period risk*, pages 115–173. WORLD SCIENTIFIC, 2011. 150

[GRP07] Francesco Giordano, Michele La Rocca, and Cira Perna. Forecasting nonlinear time series with neural network sieve bootstrap. *Computational Statistics & Data Analysis*, 51(8):3871 – 3884, 2007. 22

[Gül91] Osman Güler. On the convergence of the proximal point algorithm for convex minimization. *SIAM Journal on Control and Optimization*, 29(2):403–419, 1991. 56

[Gur10] Gurobi Optimization Inc. Gurobi optimizer version 6.0, April 2010. Software program. 33

[Han03] Deren Han. A new hybrid generalized proximal point algorithm for variational inequality problems. *Journal of Global Optimization*, 26(2):125–140, 2003. 56

[HLndlPn+11] Mohsen Heidarinejad, Jinfeng Liu, David Mu noz de la Peña, James F. Davis, and Panagiotis D. Christofides. Multirate lyapunov-based distributed model predictive control of nonlinear uncertain systems. *Journal of Process Control*, 21(9):1231 – 1242, 2011. 115

[HOS01] M. Hegland, M.R. Osborne, and J. Sun. Parallel interior point schemes for solving multistage convex programming. *Annals of Operations Research*, 108(1):75–85, 2001. 106

[HR03] Holger Heitsch and Werner Römisch. Scenario reduction algorithms in stochastic programming. *Computational Optimization and Applications*, 24(2):187–206, 2003. 52, 94

[HR09] Holger Heitsch and Werner Römisch. Scenario tree modeling for multistage stochastic programs. *Mathematical Programming*, 118(2):371–406, 2009. 87, 94

[HSB+15] C.A. Hans, P. Sopasakis, A. Bemporad, J. Raisch, and C. Reincke-Collon. Scenario-based model predictive operation control of islanded microgrids. In *54 IEEE Conf. Decision and Control*, Osaka, Japan, Dec 2015. 132, 152

[HTS14] V. Havlena, P. Trnka, and B. Sheridan. Management of complex water networks. In T. Samad and A.M. Annaswamy, editors, *The Impact of Control Technology*. IEEE Control Systems Society, 2nd edition, 2014. available at www.ieeecss.org. 3

[HW01] Kjetil Høyland and Stein W. Wallace. Generating scenario trees for multistage decision problems. *Management Science*, 47(2):295–307, 2001. 93

[HY12] Bingsheng He and Xiaoming Yuan. An accelerated inexact proximal point algorithm for convex minimization. *Journal of Optimization Theory and Applications*, 154(2):536–548, 2012. 56

[ILO09] ILOG, Inc. Cplex 9.0 user manual, 2009. Software program. 33

[IY06] T. Ishii and K. Yasuda. Hierarchical decentralized autonomous control in super-distributed energy system. In *2006 IEEE International Conference on Systems, Man and Cybernetics*, volume 1, pages 746–751, Oct 2006. 114

[JDOMC14] B. Joseph-Duran, C. Ocampo-Martínez, and G. Cembrano. Output-feedback model predictive control of sewer networks through moving horizon estimation. In *53rd IEEE Conference on Decision and Control*, pages 1061–1066, Dec 2014. 152

[Joh00] K. H. Johansson. The quadruple-tank process: a multivariable laboratory process with an adjustable zero. *IEEE Transactions on Control Systems Technology*, 8(3):456–465, May 2000. 124

[JPJ08] Honghoon Jang, Anjin Park, and Keechul Jung. Neural network implementation using CUDA and openmp. In *Digital Image Computing: Techniques and Applications (DICTA), 2008*, pages 155–161, Dec 2008. 87

[JW01] Zhong-Ping Jiang and Yuan Wang. Input-to-state stability for discrete-time nonlinear systems. *Automatica*, 37(6):857 – 869, 2001. 122

[KCRC10] Basil Kouvaritakis, Mark Cannon, Saša V. Raković, and Qifeng Cheng. Explicit use of probabilistic distributions in linear predictive control. *Automatica*, 46(10):1719 – 1724, 2010. 49

[KK13] U. Kalabić and I. Kolmanovsky. Decentralized constraint enforcement using reference governors. In *52nd IEEE Conference on Decision and Control*, pages 6415–6421, Dec 2013. 114

[LHS11] Alysha M. De Livera, Rob J. Hyndman, and Ralph D. Snyder. Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American Statistical Association*, 106(496):1513–1527, 2011. 13, 21

[LMO08] Adrian S. Lewis, Michael, and L. Overton. Behavior of BFGS with an exact line search on nonsmooth examples. technical report. 2008. 133

[LN89] D.C. Liu and J. Nocedal. On the limited memory BFGS method for large-scale optimization. *Mathematical Programming*, 56(1–3):503–528, 1989. 133

[LNC+11] Quoc V. Le, Jiquan Ngiam, Adam Coates, Ahbik Lahiri, Bobby Prochnow, and Andrew Y. Ng. On optimization methods for deep learning. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 265–272, 2011. 133

[LO13] Adrian S. Lewis and Michael L. Overton. Nonsmooth optimization via quasi-newton methods. *Mathematical Programming*, 141(1):135–163, 2013. 144

[LPBC14] J. Liu, H. Peyrl, A. Burg, and G. A. Constantinides. FPGA implementation of an interior point method for high-speed model predictive control. In *2014 24th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–8, Sept 2014. 39

[LZNDS10] S. Leirens, C. Zamora, R.R. Negenborn, and B. De Schutter. Coordination in urban water supply networks using distributed model predictive control. In *American Control Conference (ACC), 2010*, pages 3957–3962, Baltimore, USA, June 2010. 86

[MAA13] M. A. Müller, D. Angeli, and F. Allgöwer. Economic model predictive control with transient average constraints. In *52nd IEEE Conference on Decision and Control*, pages 5119–5124, Dec 2013. 151

[McC07] M.D. McCool. Signal processing and general-purpose computing and GPUs [exploratory dsp]. *Signal Processing Magazine, IEEE*, 24(3):109–114, May 2007. 87

[MF84] S. Miyaoka and M. Funabashi. Optimal control of water distribution systems by network flow theory. *Automatic Control, IEEE Transactions on*, 29(4):303–311, Apr 1984. 88

[MP04] G. McCormick and R.S. Powell. Derivation of near-optimal pump schedules for water distribution by simulated annealing. *Journal of the Operational Research Society*, 55:728–736, July 2004. 12, 85

[MS06] L. Magni and R. Scattolini. Stabilizing decentralized model predictive control of nonlinear systems. *Automatica*, 42(7):1231–1236, 2006. 115

[MXAM12] J.F.C Mota, J.M.F Xavier, P.M.Q Aguiar, and Püschel M. Distributed ADMM for model predictive control and congestion control. In *IEEE 51st Conference on Decision and Control (CDC)*, pages 5110–5115, Dec 2012. 39

[NBGS08] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with CUDA. *Queue - GPU Computing*, 6(2):40–53, March 2008. 76, 79

[Nes83] Y. Nesterov. A method of solving a convex programming problem with convergence rate $\mathcal{O}(1/k^2)$. *Soviet Mathematics Doklady*, 72(2):372–376, 1983. 58, 62, 97, 132

[Nes12] Yu. Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, dec 2012. 63

[NS06] Arkadi Nemirovski and Alexander Shapiro. Convex approximations of chance constrained programs. *SIAM Journal on Optimization*, 17(4):969–996, 2006. 85, 86

[NVI12] NVIDIA Corporation. cuBLAS library user guide, October 2012. 69, 76

[NW06] Jorge Nocedal and Steve J. Wright. *Numerical optimization*. Springer Series in Operations Research and Financial Engineering. Springer, Berlin, 2006. NEOS guide http://www-fp.mcs.anl.gov/otc/Guide/. 132, 133, 143, 144

[OJM08] F. Oldewurtel, C.N. Jones, and M. Morari. A tractable approximation of chance constrained stochastic MPC based on affine disturbance feedback. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 4731–4736, Cancún, Mexico, Dec 2008. 49

[OL94] L. Ormsbee and K. Lansey. Optimal control of water supply pumping systems. *Journal of Water Resources Planning and Management*, 120(2):237–252, 1994. 12

[OMBP11] C. Ocampo-Martínez, S. Bovo, and V. Puig. Partitioning approach oriented to the decentralised predictive control of large-scale systems. *Journal of Process Control*, 21(5):775 – 786, 2011. Special Issue on Hierarchical and Distributed Model Predictive Control. 34

[OMBPB12a] C. Ocampo-Martínez, D. Barcelli, V. Puig, and A. Bemporad. Hierarchical and decentralised model predictive control of drinking water networks: Application to barcelona case study. *IET Control Theory Applications*, 6(1):62–71, January 2012. 28

[OMBPB12b] C. Ocampo-Martínez, D. Barcelli, V. Puig, and A. Bemporad. Hierarchical and decentralised model predictive control of drinking water networks: Application to barcelona case study. *IET Control Theory Applications*, 6(1):62–71, January 2012. 114

[OMFBP10] C. Ocampo-Martínez, V. Fambrini, D. Barcelli, and V. Puig. Model predictive control of drinking water networks: A hierarchical and decentralized approach. In *American Control Conference (ACC), 2010*, pages 3951–3956, Baltimore, USA, June 2010. 17, 86, 88, 89, 91

[OMPB11] Carlos Ocampo-Martínez, Vicen Puig, and Samuele Bovo. Decentralised MPC based on a graph partitioning approach applied to the barcelona drinking water network. *IFAC Proceedings Volumes*, 44(1):1577 – 1583, 2011. 18th IFAC World Congress. 114

[OMPC+09] C. Ocampo-Martínez, V. Puig, G. Cembrano, R. Creus, and M. Minoves. Improving water management efficiency by using optimization-based control strategies: The barcelona case study. *Water Science and Technology: Water Supply*, 9(5):565–575, 2009. 13, 15, 85, 86, 88

[OMPG+14] C. Ocampo-Martínez, V. Puig, J.M. Grosso, G. Cembrano, P. Sopasakis, A.K. Sampathirao, P. Patrinos, D. Bernardini, G.S. Gnecco, and A. Bemporad. D2.2 model predictive control algorithms for operational management of urban water networks. Deliverable, The EFFINET Project (EU FP7-318556), October 2014. `http://effinet.eu/download/Deliverables/`. 8

[OSB13] B. O'Donoghue, G. Stathopoulos, and S. Boyd. A splitting method for optimal control. *IEEE Transactions on Control Systems Technology*, 21(6):2432–2442, Nov 2013. 40, 71

[PB12] P. Patrinos and A. Bemporad. An accelerated dual gradient-projection algorithm for linear model predictive control. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 662–667, Dec 2012. 33

[PB13] P. Patrinos and A. Bemporad. Proximal newton methods for convex composite optimization. In *52nd IEEE Conference on Decision and Control*, pages 2358–2363, Dec 2013. 139

[PB14a] Neal Parikh and Stephen Boyd. Proximal algorithms. *Found. Trends Optim.*, 1(3):127–239, January 2014. 45, 53, 96

[PB14b] P. Patrinos and A. Bemporad. An accelerated dual gradient-projection algorithm for embedded linear model predictive control. *IEEE Trans. Aut. Contr.*, 59(1):18–33, 2014. 62, 64, 73

[PB14c] P. Patrinos and A. Bemporad. An accelerated dual gradient-projection algorithm for embedded linear model predictive control. *Automatic Control, IEEE Transactions on*, 59(1):18–33, Jan 2014. 98, 105

[PDHM14] K.S. Postek, Dick Den Hertog, and Bertrand Melenberg. Tractable counterparts of distributionally robust constraints on risk measures. Discussion Paper 2014-031, Tilburg University, Center for Economic Research, 2014. 151

[PGL12] M. Prandini, S. Garatti, and J. Lygeros. A randomized approach to stochastic model predictive control. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 7315–7320, Maui, Hawaii, USA, Dec 2012. 86

[PKGM16] Bart P. G. Van Parys, Daniel Kuhn, Paul J. Goulart, and Manfred Morari. Distributionally robust control of constrained stochastic systems. *IEEE Trans. Automat. Contr.*, 61(2):430–442, 2016. 151

[PSA14] C.G. Petra, O. Schenk, and M. Anitescu. Real-time stochastic optimization of complex energy systems on high-performance computers. *Computing in Science Engineering*, 16(5):32–42, 2014. 39

[PSB14] P. Patrinos, L. Stella, and A. Bemporad. Forward-backward truncated Newton methods for convex composite optimization. Technical report, 2014. `http://arxiv.org/abs/1402.6655`. 133

[PSS11a] P. Patrinos, P. Sopasakis, and H. Sarimveis. A global piecewise smooth newton method for fast large-scale model predictive control. *Automatica*, 47(9):2016 – 2022, 2011. 33

[PSS11b] Panagiotis Patrinos, Pantelis Sopasakis, and Haralambos Sarimveis. Stochastic model predictive control for constrained networked control systems with random time delay. *IFAC Proceedings Volumes*, 44(1):12626 – 12631, 2011. 18th IFAC World Congress. 39

[PSSB14] Panagiotis Patrinos, Pantelis Sopasakis, Haralambos Sarimveis, and Alberto Bemporad. Stochastic model predictive control for constrained discrete-time Markovian switching systems. *Automatica*, 50(10):2504 – 2514, 2014. 39, 132

[PTB11] P. Patrinos, S. Trimboli, and A. Bemporad. Stochastic MPC for real-time market-based optimal power dispatch. In *50th IEEE Conference on Decision and Control and European Control Conference*, pages 7111–7116, Dec 2011. 132, 152

[PVSC10] Bruno Picasso, Daniele De Vito, Riccardo Scattolini, and Patrizio Colaneri. An MPC approach to the design of two-layer hierarchical control systems. *Automatica*, 46(5):823 – 831, 2010. 115

[QB03] S.Joe Qin and Thomas A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733 – 764, 2003. 13

[QPC+06] J. Quevedo, V. Puig, G. Cembrano, J. Aguilar, C. Isaza, D. Saporta, G. Benito, M. Hedo, and A. Molina. Estimating missing and false data in flow meters of a water distribution network. *IFAC Proceedings Volumes*, 39(13):1181 – 1186, 2006. 6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes. 13

[RFFT13] S. Riverso, M. Farina, and G. Ferrari-Trecate. Plug-and-play decentralized model predictive control. *IEEE Transactions on Automatic Control*, 58(10):2608 – 2614, 2013. 115

[RFT12] Stefano Riverso and Giancarlo Ferrari-Trecate. Tube-based distributed control of linear constrained systems. *Automatica*, 48(11):2860 – 2865, 2012. 114

[RJM13] Stefan Richter, Colin Neil Jones, and Manfred Morari. Certification aspects of the fast gradient method for solving the dual of parametric convex programs. *Mathematical Methods of Operations Research*, 77(3):305–321, 2013. 70

[RK12] Paula Rocha and Daniel Kuhn. Multistage stochastic portfolio optimisation in deregulated electricity markets using linear decision rules. *European Journal of Operational Research*, 216(2):397 – 408, 2012. 39

[RM09] J. B. Rawlings and D. Q. Mayne. *Model predictive control: theory and design*. Madison: Nob Hill Publishing, 2009. 13, 25

[Roc72] R.T. Rockafellar. *Convex analysis*. Princeton university press, 1972. 60, 61, 97, 98, 140

[Roc76] R. Tyrrell Rockafellar. Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization*, 14(5):877–898, 1976. 55, 56

[Roc99] R. T. Rockafellar. Duality and optimality in multistage stochastic programming. *Ann. Oper. Res*, pages 1–19, 1999. 69

[RS13] J. Rogers and N. Slegers. Robust parafoil terminal guidance using massively parallel processing. *Journal of Guidance, Control, and Dynamics*, 36(5):1336–1345, 2013. 39

[RW91] R. T. Rockafellar and Roger J.-B. Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research*, 16(1):119–147, 1991. 68

[RW09] R.T. Rockafellar and J.B. Wets. *Variational analysis*. Springer-Verlag, Berlin, 3rd edition, 2009. 43, 45, 54, 61, 97, 138

[SB16] Pantelis Sopasakis and Alberto Bemporad. Advanced topics in control systems: Risk-averse optimisation, February 2016. Lecture notes. 51

[Sca09] Riccardo Scattolini. Architectures for distributed and hierarchical model predictive control a review. *Journal of Process Control*, 19(5):723 – 731, 2009. 114

[SDR09] A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on stochastic programming: Modeling and theory*. Society for Industrial and Applied Mathematics, Philadelphia, 2009. 49, 51, 68, 86, 135, 150

[SFP16] P. Sopasakis, N. Freris, and P. Patrinos. Accelerated reconstruction of a compressively sampled data stream. In *European Signal Processing Conference*, pages 1–5, Hungary, Budapest, aug 2016. 60

[SFS+14] P. Sopasakis, R. Farias, A.K. Sampathirao, S. Lopez-Martinez, D. Eliades, J.M. Mirats Tur, V. Bernardo, M. Milis, J.A. Lopez Merino, V. Puig, C. Ocampo-Martínez, and G. Cembrano. D2.3 economic and technical evaluation of operational management algorithms. Deliverable, The EFFINET Project (EU FP7-318556), Sept 2014. `http://effinet.eu/download/Deliverables/`. 8

[SGB14] Katya Scheinberg, Donald Goldfarb, and Xi Bai. Fast first-order methods for composite convex optimization with backtracking. *Foundations of Computational Mathematics*, 14(3):389–417, 2014. 63

[SGS+14] Ajay Kumar Sampathirao, Juan Manuel Grosso, Pantelis Sopasakis, Carlos Ocampo-Martínez, Alberto Bemporad, and Vicenç Puig. Water demand forecasting for the optimal operation of large-scale drinking water networks: The Barcelona case study. In *19th IFAC World Congress*, pages 10457–10462. Cape Town, South Africa, 2014. xv, 12, 39, 85, 86, 89, 90, 91, 105, 107, 149

[Sha09] Alexander Shapiro. On a time consistency concept in risk averse multistage stochastic programming. *Operations Research Letters*, 37(3):143 – 147, 2009. 150

[Sha12] Alexander Shapiro. Minimax and risk averse multistage stochastic programming. *European Journal of Operational Research*, 219(3):719–726, 2012. Feature Clusters. 150

[SKN+15] Gokul Siva Sankar, S. Mohan Kumar, Sridharakumar Narasimhan, Shankar Narasimhan, and S. Murty Bhallamudi. Optimal control of water distribution networks with storage facilities. *Journal of Process Control*, 32:127 – 137, 2015. 86, 152

[SM16] G. Schildbach and M. Morari. Scenario-based model predictive control for multi-echelon supply chain management. *European Journal of Operational Research*, 252(2):540 – 549, 2016. 132, 152

[SO96] Anne Shepherd and Leonard Ortolano. Water-supply system operations: critiquing expert-system approach. *Journal of Water Resources Planning and Management*, 122(5):348–355, 1996. 13

[Sol02] Skander Soltani. On the use of the wavelet decomposition for time series prediction. *Neurocomputing*, 48(14):267 – 277, 2002. 22

[Son13] D.-P Song. *Optimal control and optimization of stochastic supply chain systems*. Advances in Industrial Control. Springer, 2013. 39

[SPM10] SPM: Signal Processing Microelectronics. QPC: Quadratic programming in C version 2.0, 2010. Software program. 79

[SS94] R. Scattolini and N. Schiavoni. A multirate model based predictive controller. In *Decision and Control, 1994., Proceedings of the 33rd IEEE Conference on*, volume 1, pages 243–248 vol.1, Dec 1994. 115

[SS97] Hanif D. Sherali and Ernest P. Smith. A global optimization approach to a water distribution network design problem. *Journal of Global Optimization*, 11(2):107–132, 1997. 12

[SSB16] Ajay Kumar Sampathirao, Pantelis Sopasakis, and Alberto Bemporad. *Decentralised hierarchical multi-rate control of large-scale drinking water networks*, pages 45–56. Springer International Publishing, Cham, 2016. 113, 149

[SSBP15] A. K. Sampathirao, P. Sopasakis, A. Bemporad, and P. Patrinos. Distributed solution of stochastic optimal control problems on GPUs. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 7183–7188, Dec 2015. 38, 140, 149

[SSBP16] A.K. Sampathirao, P. Sopasakis, A. Bemporad, and P. Patrinos. GPU-accelerated stochastic predictive control of drinking water networks. `http://arxiv.org/abs/1604.01074`, 2016. 84, 149

[SSE+14] L.E. Sokoler, L. Standardi, K. Edlund, N.K. Poulsen, H. Madsen, and J.B. Jørgensen. A dantzigwolfe decomposition algorithm for linear economic model predictive control of dynamically decoupled subsystems. *Journal of Process Control*, 24(8):1225–1236, 2014. Economic nonlinear model predictive control. 151

[SSG+13] P. Sopasakis, A.K. Sampathirao, J.M. Grosso, C. Ocampo-Martínez, V. Puig, G. Cembrano, and A. Bemporad. D2.1 control-oriented modelling for operational management of urban water networks. Deliverable, The EFFINET Project (EU FP7-318556), January 2013. `http://effinet.eu/download/Deliverables/`. 8, 88

[SSG+14] P. Sopasakis, A.K. Sampathirao, J.M. Grosso, C. Ocampo-Martínez, V. Puig, G. Cembrano, Panagiotis Patrinos, D. Bernardini, and G.S. Gnecco. D2.3 economic and technical evaluation of operational management algorithms. Deliverable, The EFFINET Project (EU FP7-318556), Sept 2014. `http://effinet.eu/download/Deliverables/`. 8

[STP16] L. Stella, A. Themelis, and P. Patrinos. Forward-backward quasi-newton methods for nonsmooth optimisation problems. `https://arxiv.org/pdf/1604.08096v2.pdf`, 2016. 133, 139, 146, 150

[TB09] VuNam Tran and MietekA. Brdys. Optimizing control by robustly deasible model predictive control and application to drinking water distribution systems. In Cesare Alippi, Marios Polycarpou, Christos Panayiotou, and Georgios Ellinas, editors, *Artificial Neural Networks ICANN 2009*, volume 5769 of *Lecture Notes in Computer Science*, pages 823–834. Springer Berlin Heidelberg, 2009. 86

[Tse08] P. Tseng. On accelerated proximal gradient methods for convex-concave optimization, 2008. 59

[TSP16] A. Themelis, L. Stella, and P. Patrinos. Forward-backward envelope for the sum of two nonconvex functions: Further properties and nonmonotone line-search algorithms. `https://arxiv.org/pdf/1606.06256.pdf`, 2016. 150, 152

[TvBdW+03] U Thissen, R van Brakel, A.P de Weijer, W.J Melssen, and L.M.C Buydens. Using support vector machines for time series prediction. *Chemometrics and Intelligent Laboratory Systems*, 69(12):35 – 49, 2003. 22

[Uni12] United Nations Environment Programme. Measuring water use in a green economy. Technical report, International Resource Panel Working Group on Water Efficiency, 2012. `http://www.unep.org/resourcepanel-old/Portals/24102/Measuring_Water.pdf`. 1

[Uni15] United Nations. World urbanization prospects: The 2014 revision. Technical report, Department of Economic and Social Affairs Population Division, New York, 2015. `https://esa.un.org/unpd/wup/Publications/Files/WUP2014-Report.pdf`. 2

[vHB02] D.H. van Hessem and O.H. Bosgra. A conic reformulation of model predictive control including bounded and stochastic disturbances under state and input constraints. In *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, volume 4, pages 4643–4648 vol.4, Las Vegas, USA, Dec 2002. 86

[VLK09] M. Vaccarini, S. Longhi, and M.R. Katebi. Unconstrained networked decentralized model predictive control. *Journal of Process Control*, 19(2):328 – 339, 2009. 115

[VMK13] C. Vermillion, A. Menezes, and I. Kolmanovsky. Stable hierarchical model predictive control using an inner loop reference model and λ-contractive terminal constraint sets. *Automatica*, pages –, 2013. 115

[Š91] D. D. Šiljak. *Decentralised control of complex systems*. Academic Press, 1991. 119

[WM97] Jr. Watkins, D. and D. McKinney. Finding robust solutions to water resources problems. *Journal of Water Resources Planning and Management*, 123(1):49–58, 1997. 86

[WOPQ14] Ye Wang, Carlos Ocampo-Martínez, Vicenç Puig, and Joseba Quevedo. Gaussian-process-based demand forecasting for predictive control of drinking water networks. In *Critical Information Infrastructures Security - 9th International Conference, CRITIS 2014, Limassol, Cyprus, October 13-15, 2014, Revised Selected Papers*, pages 69–80, 2014. 89

[YPS94] G. Yu, R.S. Powell, and M.J.H. Sterling. Optimized pump scheduling in water distribution systems. *Journal of Optimization Theory and Applications*, 83(3):463–488, 1994. 85

[ZG13] Y. Zhang and G. B. Giannakis. Robust optimal power flow with wind integration using conditional value-at-risk. In *Smart Grid Communications (SmartGridComm), 2013 IEEE International Conference on*, pages 654–659, Oct 2013. 150

[ZH14] D. Zhu and G. Hug. Decomposed stochastic model predictive control for optimal dispatch of storage and generation. *IEEE Transactions on Smart Grid*, 5(4):2044–2053, July 2014. 152

[Zhu95] Ciyou Zhu. Asymptotic convergence analysis of the forward-backward splitting algorithm. *Mathematics of Operations Research*, 20(2):449–464, 1995. 58

[Zip00] P.H. Zipkin. *Foundations of inventory management*. McGrawHill, New York, 2000. 39

[ZKR13] Steve Zymler, Daniel Kuhn, and Berç Rustem. Distributionally robust joint chance constraints with second-order moment information. *Mathematical Programming*, 137(1):167–198, 2013. 151

[ZS89] U. Zessler and U. Shamir. Optimal operation of water distribution systems. *Journal of Water Resources Planning and Management*, 115(6):735–752, 1989. 12, 85

[ZSSM13] X. Zhang, G. Schildbach, D. Sturzenegger, and M. Morari. Scenario-based MPC for energy-efficient building climate control under weather and occupancy uncertainty. In *European Control Conference (ECC)*, pages 1029–1034, July 2013. 132