# Online optimization-based predictive flight control using first-order methods

Ferranti, Laura

**DOI**
10.4233/uuid:786608c2-38ce-4dbc-97d2-8a26080829ba

**Publication date**
2017

**Document Version**
Final published version

**Citation (APA)**
Ferranti, L. (2017). *Online optimization-based predictive flight control using first-order methods*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:786608c2-38ce-4dbc-97d2-8a26080829ba

**Important note**
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

# Online Optimization-Based Predictive Flight Control Using First-Order Methods

Ph.D. Thesis

Laura Ferranti

*Delft University of Technology, 2017*

ISBN 978-94-6186-838-1

Cover design by Laura Ferranti.
Printed by Ipskamp Printing.

Delft University of Technology

# ONLINE OPTIMIZATION-BASED PREDICTIVE FLIGHT CONTROL USING FIRST-ORDER METHODS

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof. Ir. K.Ch.A.M. Luyben;
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op
Woensdag 6, September 2017 om 10:00 uur

door

**Laura FERRANTI**

Master of Science in Control Engineering at University of Rome Tor Vergata, Italië
geboren te Rome, Italië

This dissertation has been approved by the
promotor: Prof. dr. ir. M. Verhaegen, and
copromotor: Dr. ir. T. Keviczky

Composition of the doctoral committee:

| | |
|---|---|
| Rector Magnificus, | chairperson |
| Prof. dr. ir. M. Verhaegen, | promotor |
| Dr. ir. T. Keviczky, | copromotor |

Independent members:

| | |
|---|---|
| Prof. dr. ir. Bart De Schutter, | Technische Universiteit Delft |
| Prof. dr. Etienne de Klerk, | Technische Universiteit Delft |
| Dr. Panagiotis Patrinos | Katholieke Universiteit Leuven |
| Dr. Colin N. Jones | École Polytechnique Fédérale de Lausanne |





This research was funded by the European Union's Seventh Framework Programme FP7/2007-2013 under grant agreement n. AAT-2012-RTD-2314544 (**RECONFIGURE**).


This dissertation has been completed in fulfillment of the requirements of the Dutch Institute of Systems and Control (DISC) for graduate study.

*"... the raison d'être for model predictive control is its ability to handle control problems where off-line computation of a control law is difficult or impossible."*

D. Q. Mayne et al, 1999.

# Acknowledgements

When I started my Ph.D. at Delft University of Technology, I knew that for the next four years of my life I would dedicate myself to the final objective of writing this dissertation. Here I am, after four years since I started pursuing that final goal. The road to this point was far from easy, but I was lucky to have the right persons helping me through the journey.

First and foremost, I would like to thank my supervisor, Tamás Keviczky, for his guidance and support during all these years. He motivated me to complete this journey and gave me the freedom to develop my own critical thinking and independently explore different exciting research directions. Furthermore, I would like to thank my promotor, Michel Verhaegen, for helping me find my research direction and for the interesting discussions over the years.

Next, I would like to express my gratitude to all the members of my Ph.D. thesis committee: Bart De Schutter, Etienne de Klerk, Panagiotis Patrinos, and Colin Jones. I felt honored to have them as reviewers of my dissertation. A special thanks to Colin Jones for hosting me in his research lab. It was an exciting experience collaborating with him and his Ph.D. students. I would also like to thank Panagiotis Patrinos for the great contributions to the field of online optimization and model predictive control. I learned a lot from his papers and conference presentations. Finally, I am extremely thankful to Bart De Schutter for the support during my years at DCSC and to Etienne de Klerk for accepting to be part of this process.

I would like to acknowledge the amazing people involved in the RECONFIGURE project. From the TU Delft side of the project, I would like to thank Yiming Wan, my office mate and coauthor, for the support during the years of RECONFIGURE, and Will van Geest for the technical support with the installation of the benchmark model. From the University of Cambridge side, I would like to thank Edward Hartley, Kestutis Sialuys, and Jan Maciejowski for the interesting discus-

sions we had during the meetings. From the Airbus side, a special thanks to Philippe Goupil and Josep Boada-Bauxell for the interesting discussions on the model and the challenges related to the use of model predictive control in such a framework. Finally, I would like to thank the people from Deimos for the project coordination and the support provided during the verification campaign.

A special mention goes to the coauthors of my publications: Ion Necoara (University Politehnica Bucharest), Ye Pu (UC Berkeley), Giorgos Stathopoulous (EPFL), and Yiming Wan (MIT). It was a pleasure to work with them. I learned a lot from all these collaborations and the final results of my thesis would have not been possible without their help.

DCSC has offered me full support and encouragement during the last four years. I was honored to be part of this department. There are so many that I would like to acknowledge. A special thanks goes to the secretariat team of the department and especially to Marieke, Heleen, Kiran, Mascha, and Kitty. I would like to thank my fellow Ph.D. students who shared their experiences with me. In particular, I would like to thank Renshi for all the support he gave me during these years, Farid for all the nice chats and the Iranian card games, Zhou and Anqui for the nice time we spent in the US, and Reinier for helping me with the Dutch summary of my thesis. Finally, a special thanks to Vahab for all the interesting discussions, the lunches, and his friendship.

A special acknowledgment goes to the people who helped me during my time in the LA lab at EPFL. In particular, I want to thank Altug, Andrea, Christophe, Diogo, Faran, Francisco, Giorgos, Harsh, Ivan, Luca, Predrag, René, Tafarel, Tomasz, and Ye. My time at EPFL was great thanks to all of them. I would also like to thank Calin, Maria, Raja, and Nimisha for the great support and the nice time we had together in Switzerland.

I would also like to extend my gratitude to my Amsterdam friends who provided the right amount of fun and support during my Ph.D. journey. In particular, I want to thank all the *"Djangonians"*: Albana, Christian, Dirk, baby Hana (the last-generation Djangonian), and Kaveh. They have been of great support sharing their Ph.D. experience with me in the past few years. Finally, I want to thank Marieke (and baby Lorèn) for her friendship and the great time spent together.

Last but not least, I would like to thank my family for supporting me all these years. My parents for all their trips back and forth to the Netherlands, for all the sleepless nights during my travels outside Europe, and for motivating me to finish this journey. My grandfather who taught me what it means to be dedicated to a job and my grandmother for her unconditional love. My beautiful little Nero that I miss every day. Finally, Cristiano for his guidance, care, and love during all the ups and downs of this journey.

Laura Ferranti
Delft, The Netherlands, September 2017

# Summary

The use of model predictive control (MPC) techniques for embedded systems with fast dynamics is still limited. In fields such as aerospace or automotive, the use of classical control methods such as PID is still significant. The presence of constraints, however, impacts on the performance of these controllers that are usually designed to avoid constraint saturation. MPC techniques are the obvious alternative to handle constraint saturation and fully exploit the operative range of the system. Furthermore, MPC can be used as a fault-tolerant controller to handle actuator faults and control reallocation in a modular and systematic way.

In this dissertation, we rely on MPC techniques to handle constraints and actuator faults, motivated by an aerospace application (i.e., the longitudinal control of an Airbus passenger aircraft). First, we show the improvements in terms of tracking performance when constraints saturate during maneuvers compared to a PID implementation. Second, we show how to handle actuator faults by actively performing control constraint reconfigurations in the MPC problem formulation. The proposed fault-tolerant strategy strongly relies on the MPC capability of handling constraints and directly controlling each actuator independently.

The advantages of MPC in terms of performance and fault tolerance, however, are shadowed by the computational requirements of this technique. For medium- and large-scale applications, MPC requires the online solution of a constrained optimization problem (often a constrained quadratic programming problem). The presence of an optimizer compromises the performance of the controller in terms of computation time and increases its requirements in terms of hardware and software. Furthermore, from an industrial perspective, the presence of the iterative optimizer affects also the reliability of the design. In order to address the concerns above for the aforementioned applications, the optimizer should be certifiable in terms of worst-case

computation time to return medium-accuracy solutions (which are in practice sufficient to ensure performance). Furthermore, the algebraic operations involved in the optimizer should be simple (e.g., the solver should avoid to perform matrix inversions or complex projections).

In this dissertation, we analyze the state of the art in terms of MPC-tailored optimizers suitable for online optimization. In particular, we focused on first-order solvers, such as proximal-gradient and splitting methods, which are suitable candidates for implementations on embedded platforms. In this respect our contribution is the following. First, we focus on how the use of a subclass of these solvers—the dual projected gradient methods—impacts on the (primal) feasibility of the MPC problem solution and on the stability of the controlled system. We propose to use a robust control technique (i.e., constraint tightening) to guarantee recursive feasibility of the MPC problem and closed-loop stability. Second, in order to improve the computation time of the solver, we combine the use of the proposed tightening scheme with splitting methods. These algorithms allow us to exploit the structure of the MPC problem and, if available, parallel hardware architectures. This approach aims to address both regulation and tracking problems. Third, if parallel hardware architectures are not available, we propose the use of a novel splitting algorithm that allows one to randomly update the decision variables of the MPC problem formulation. The use of these random updates aims to reduce the overall computation time compared to the update of the full set of decision variables at each iteration of the solver. For this algorithm, we provide a complete proof of convergence and two accelerated versions to further improve the computation time.

The proposed contributions aim to bring MPC closer to actual implementation on the next generation of aircraft.

# Samenvatting

De toepassing van model predictive control (MPC) technieken voor embedded systemen met snelle dynamiek is nog steeds beperkt. Bij toepassingen zoals lucht- en ruimtevaart is het gebruik van klassieke regelmethoden zoals PID nog steeds significant aanwezig. Het bestaan van randvoorwaarden heeft echter invloed op de prestaties van deze controllers, die gewoonlijk zijn ontworpen om randvoorwaardenverzadiging te vermijden. MPC-technieken zijn het voor de hand liggende alternatief om randvoorwaardenverzadiging aan te pakken en het operationele bereik van het systeem volledig te benutten. Bovendien kan MPC als fouttolerante controller gebruikt worden om actuatorfouten en regelactie-herverdeling op modulaire en systematische wijze te behandelen.

In dit proefschrift, gemotiveerd door een ruimtevaarttoepassing (dat wil zeggen de longitudinale controle van een Airbus-passagiersvliegtuig), vertrouwden we op MPC-technieken om randvoorwaarden en actuatorfouten aan te pakken. Ten eerste tonen we de verbeteringen aan in termen van het referentievolgend vermogen wanneer randvoorwaarden tijdens manoeuvres verzadigen, vergeleken met een PID-implementatie. Ten tweede tonen we aan hoe met de fouten in actuatoren moet worden omgegaan door actief de randvoorwaarden in het MPC-probleem te herformuleren. De voorgestelde fouttolerante strategie berust sterk op het vermogen van een MPC-regelaar om randvoorwaarden te handhaven en onafhankelijk elke actuator aan te sturen.

De voordelen van MPC in termen van prestatie en fouttolerantie worden echter overschaduwd door de vereiste rekenkracht voor deze techniek. Voor middelen grootschalige toepassingen vereist MPC het online oplossen van een optimalisatieprobleem met randvoorwaarden (vaak een kwadratisch programmeringsprobleem

met randvoorwaarden). De aanwezigheid van optimaliseringssoftware compromitteert de prestaties van de controller met betrekking tot de rekentijd en verhoogt de eisen ten aanzien van hardware en software. Vanuit industrieel oogpunt beïnvloedt de iteratieve optimaliseringssoftware ook de betrouwbaarheid van het ontwerp. Om de bovengenoemde zorgen voor de bovengenoemde toepassingen aan te pakken, dient de optimaliseringssoftware te worden gecertificeerd in termen van de langst mogelijke rekentijd om oplossingen van gemiddelde nauwkeurigheid op te leveren (die in de praktijk voldoende zijn om prestaties te waarborgen). Bovendien moeten de algebraïsche bewerkingen die betrokken zijn bij de optimaliseringssoftware eenvoudig zijn (de solver moet bijvoorbeeld voorkomen dat matrix-inversies of ingewikkelde projecties worden uitgevoerd).

In dit proefschrift hebben we de stand van de techniek geanalyseerd voor optimaliseringssoftware specifiek voor MPC, die geschikt zijn voor online optimalisatie. In het bijzonder richten we ons op "first-order" solvers, zoals "proximal-gradient"- en splitsingsmethoden, die geschikt zijn voor implementatie op embedded platforms. Met betrekking tot dit is onze bijdrage het volgende. Ten eerste richten we ons op hoe het gebruik van een subklasse van deze solvers—de duale "projected-gradient" methoden—invloed heeft op de vervulbaarheid van de randvoorwaarden in het MPC-probleem en op de stabiliteit van het geregelde systeem. We stellen voor om een robuuste regeltechniek (dat wil zeggen, een aanscherping van de randvoorwaarden) te gebruiken om recursieve vervulbaarheid van de MPC-randvoorwaarden en de stabiliteit van het system en regelaar in een gesloten lus te garanderen. Ten tweede combineren we het gebruik van de voorgestelde aanscherping van de randvoorwaarden met splitsingsmethoden om de rekentijd van de solver te verbeteren. Deze algoritmen stellen ons in staat om de structuur van het MPC-probleem uit te buiten en, indien beschikbaar, parallelle hardware-architecturen aan te wenden. Deze aanpak kan zowel voor regels- als referentievolgproblemen worden gebruikt. Ten derde stellen we voor indien parallelle hardware-architecturen niet beschikbaar zijn, om een nieuw splitsingsalgoritme te gebruiken waarmee men de beslissingsvariabelen van de MPC-probleemformulering willekeurig kan updaten. Het gebruik van deze willekeurige gekozen updates vermindert de totale berekeningstijd in vergelijking met de update van de volledige set beslissingsvariabelen bij elke iteratie van de solver. Voor dit algoritme hebben we een compleet bewijs geleverd van convergentie en twee versnelde versies van het algoritme voorgesteld om de rekentijd verder te verbeteren.

De voorgestelde bijdragen willen MPC dichter bij werkelijke implementatie brengen op de eerstvolgende generatie vliegtuigen.

# Contents

# List of Figures

# List of Tables

# 1 Introduction

Model Predictive Control (MPC) is a popular control technique, especially suited to handle multiple-input multiple-output (MIMO) systems with state, input, and output constraints. In order to compute an optimal control action for the plant, the MPC controller solves, recursively, an optimization problem based on the current plant measurements and returns the optimal control sequence for the predicted evolution of the model over a (generally) finite time frame, called prediction horizon. Then, the controller only applies the first element of that optimal control sequence to the plant. At the next sampling instant (i.e., when new measurements are available from the plant), the MPC controller solves another optimization problem based on the new measurements, in a receding horizon fashion [69; 6; 103].

This dissertation focuses on the application of MPC to a flight control problem, that is the longitudinal control of an Airbus passenger aircraft [36; 37]. Having an MPC controller implemented on the flight control unit can lead to significant improvements in terms of the aircraft performance. This is illustrated by an example in Chapter 3 that shows how tracking performance can be improved in the presence of constraints. Furthermore, MPC formulations can be used to develop a fault-tolerant controller (as proposed in Chapter 5) and to improve the detection and the diagnosis of jamming faults on the aircraft actuators. In this scheme, the MPC controller performs a sequence of control reconfigurations that aim to actively diagnose the root-cause of the fault once it has been detected. The reconfigurations are possible thanks to the MPC capability of directly enforcing constraints and computing the optimal reference trajectory to handle the faults.

The main difficulty when using MPC in such applications, however, is related to the presence of an optimizer. In the past, the need to solve a new computationally demanding optimization problem at each problem instance has confined the use of MPC to applications with slow dynamics, such as the chemical industry. Nev-

ertheless, the growing availability of powerful, embedded computing hardware architectures has increased the interest for MPC in applications with fast dynamics as well, such as automotive or aerospace. In recent years, many researchers have focused their efforts on developing algorithms tailored to MPC for applications with fast dynamics with the following main points of attention:

- Offline optimization (i.e., the explicit MPC [4]).
- Second-order methods (i.e., *interior-point* or *active-set* methods).
- First-order methods (i.e., *proximal-gradient* or *splitting* methods).

Concerning the approaches based on explicit MPC, the possibility to move the computational effort offline is the most appealing for applications with hard real-time constraints. The main drawback of these techniques, however, is that their use is still limited to small scale problems due to the large number of regions (and resulting memory requirements) that represent the multiparametric solution of the underlying optimization problem.

Concerning the approaches based on second-order methods, these algorithms are known to provide high-accuracy solutions in a sufficiently small amount of time. Among the second-order methods, active-set methods are practically efficient to solve small/medium scale problems (some fast active set strategies for MPC have been recently proposed in [28]). These methods, however, to the best of the author's knowledge, do not provide any certification in terms of worst-case execution time, which is an important requirement for the industrial clearance of a controller.

Among the second-order methods, interior-point solvers have gained most of the attention for the class of problems considered in this dissertation. In this respect, in the late 90s, the authors of [135; 101] presented one of the first interior-point methods tailored to MPC applications. In particular, they proposed a modified version of the interior-point method presented in [73], known as predictor-corrector algorithm, aiming to improve its performance in terms of computational costs. Their interior-point method *exploits the structure* of the MPC problem and solves the linear system that derives from the KKT conditions using a discrete-time Riccati recursion. As a result, the computational cost of their algorithm grows linearly with the length $N$ of the prediction horizon instead of cubically, as in standard versions of the interior-point method, such as [73]. Almost ten years later, the authors of [131; 141] built on those results and developed novel interior-point-based algorithms that allow for fast online computation of the MPC solution. First, the authors of [131] showed that their proposed second-order solver (a modified version of the algorithm of [101] with warm starting and early termination) was a suitable candidate for systems with fast dynamics. As underlined by the same authors in [131], however, a stability analysis was missing as well as a study on primal feasibility and optimality of the solution returned by their algorithm. These issues were addressed by the authors of [141]. In particular, they provided a competitive algorithm based on an interior-point method that guarantees feasibility and closed-loop stability (the authors provided also a fast

robust MPC design based on the tube MPC concept for uncertain systems). The key to guaranteeing stability in their approach was in the warm start and early termination techniques. The results presented in [141] were based on the observation that if the suboptimal solution is feasible and the cost associated with the current problem instance is lower than the cost associated with the shifted solution (used for the warm starting of the solver), then suboptimality is sufficient to ensure stability (feasibility implies stability [118]). The final result was an interior-point-based algorithm tailored to MPC that guarantees feasibility and closed-loop stability in a certified number of iterations.

The main drawback of MPC designs that rely on interior-point methods is that a second-order method is not well suited for control applications that will run in an embedded environment. In particular, these algorithms require advanced algebraic operations and a significant amount of memory. A valid alternative to overcome these issues is represented by first-order methods.

First-order methods, such as proximal and splitting methods, recently gained significant attention (to mention a few [81; 82; 88; 61; 108; 106; 90; 91; 33; 78; 35; 128; 96; 123; 7]). These algorithms are a promising alternative to interior-point solvers. They only require simple hardware and software and are relatively easy to certify in terms of worst-case computation time and optimality.

Among the proximal methods, an interesting class of solvers suitable for MPC applications is the class of projected-gradient methods. Depending on the complexity of the MPC feasible region, these algorithms can be used (directly) on the primal MPC problem (e.g., [61; 106]) or (indirectly) on the dual MPC problem (e.g., [33; 109; 91]). In particular, if the projection on the feasible region in the primal space cannot be computed efficiently, under some mild assumptions, the computation of the MPC problem solution can be moved to the dual space. This *trick* overcomes the issues related to the projection step in primal space, but it introduces some challenges in terms of feasibility and optimality of the primal solution. In this respect, several algorithms tailored to MPC have been proposed that provide *a-priori* upper bounds on primal infeasibility, primal suboptimality, and worst-case computation time. For example, the authors of [91] proposed a custom version of Nesterov's dual fast gradient (DFG) method [82], called Accelerated Dual Gradient Projection or GPAD. They also optimized the choice of the Lipschitz constant for the dual gradient (which is important for the convergence of this class of solvers given that it affects, for instance, their step size) and provided termination criteria for the algorithm. The authors of [33] presented a DFG algorithm to solve a distributed MPC problem. Furthermore, in [31], the author proposed a generalization of the fast gradient method that allows the use of a nonuniform quadratic upper bound on the negative dual function.

The aforementioned techniques certify the fast gradient method in terms of primal infeasibility, primal suboptimality, and worst-case computation time. Guarantees on recursive feasibility and closed-loop stability (i.e., guarantees from the control perspective), however, are missing. The authors in [13; 32; 112] addressed these aspects using constraint tightening techniques. In particular, these techniques con-

sider a modified primal MPC problem that differs from the original one in the definition of the primal feasible region. The feasible region of this modified primal MPC problem is *tightened* by a factor $\epsilon_c$ proportional to the suboptimality $\epsilon$ of the DFG algorithm. A *tailored* selection of the tightening parameter allows one to preserve primal feasibility, a specified level of suboptimality, and closed-loop stability of the original primal MPC problem.

Among the splitting methods, the alternating minimization algorithm (AMA [128; 35]) and the alternating direction method of multipliers (ADMM [8; 5]) emerged to play a significant role in solving MPC problems. These algorithms are particularly appealing for MPC applications because they allow one to exploit the structure of the problem.

AMA derives from the application of the proximal-gradient method to the dual problem. Compared to ADMM, the convergence analysis and certifications in terms of feasibility and suboptimality can be derived from the ones of the proximal-gradient method and guidelines are available for the selection of the tuning parameters of the algorithm. Applications of this algorithm to MPC can be found for example in [96; 97].

ADMM derives from the application of the Douglas-Rachford method to the dual problem. Compared to AMA, the algorithm does not require any strong convexity assumption (which is an interesting feature for control problems) and shows faster convergence in practice (see, for example, [12]). Applications of this algorithm to MPC can be found for example in [60; 63; 62; 121; 30].

This dissertation mainly focuses on first-order solvers, with a particular focus on proximal-gradient and splitting methods, for aerospace applications. These applications offer several challenges from the optimization point of view. In general, these applications require a long prediction horizon, have several state, input, and output constraints (preventing the use of primal first-order solvers), and the conditioning of the resulting MPC problem is poor (with impact on the convergence of the first-order solvers). Furthermore, the computation time is in general limited to few milliseconds and it is important that the controller guarantees a feasible solution within the available time. We worked to address the issues above in different directions. In this respect, compared to the state of the art:

- We propose a novel constraint-tightening strategy when a DFG solver is used to compute the solution of the MPC problem (Chapter 2). The proposed strategy leads to less conservative primal feasible solutions and guarantees recursive feasibility and closed-loop stability. This algorithm is tested on the longitudinal control of an F-16 aircraft [113].

- Building on the results proposed in Chapter 2, we propose an improved strategy that combines the DFG method with a decomposition of the original MPC (Chapter 3). This decomposition helps improve the numerical properties of the algorithm (leading to smaller computation time) and allows the algorithm to

select different tightening parameters along the length of the prediction horizon. Furthermore, we provide closed-loop stability and recursive feasibility guarantees for both MPC for regulation and tracking problems (tracking problems are not often considered but are interesting from the practical point of view). Finally, we show how the proposed algorithm leads to improved results in terms of computation time on the Airbus passenger aircraft [37].

- Motivated by the recent developments in the theory of stochastic proximal-gradient methods [138; 51; 136; 119; 120] and the relationship between proximal-gradient methods and AMA, we propose a stochastic AMA scheme with variance reduction (i.e., SVR-AMA discussed in Chapter 4 and its accelerated versions). Compared to AMA, this algorithm performs random updates of the decision variables along the length of the prediction horizon. The main advantage is that the algorithm can return medium accuracy solutions (that are in general sufficient for flight control applications) within a smaller amount of time (measured in terms of number of iterations) compared to AMA. The proposed algorithms are also tested on the Airbus passenger aircraft [37].

We believe that these contributions can bring MPC closer to becoming a valid, high-performance alternative in constrained flight control problems and enable their implementation on real on-board control units.

## Organization of the Dissertation

This dissertation makes several contributions, with results published in international journals and conferences (refer to page 143 for a complete list of publications). The remainder is organized as follows:

- Chapter 2 presents a novel MPC algorithm to deal with the early termination of the solver used for online optimization. The proposed algorithm guarantees recursive feasibility and closed-loop stability when using Nesterov's dual fast gradient scheme to solve the MPC problem online. Effectiveness of the proposed approach is tested on the longitudinal control problem of an F-16 aircraft. Chapter 2 appeared in the *Journal of Optimal Control Applications and Methods (OCAM)* [78].

- Chapter 3 presents an improved MPC algorithm to deal with the early termination of the solver used for online optimization. Compared to the results presented in the previous chapter, this algorithm relies on the use of splitting methods to exploit the structure of the MPC problem. The proposed algorithm guarantees recursive feasibility and closed-loop stability. Furthermore, the algorithm can be implemented on parallel hardware architectures. Finally, the performance improvements (in terms of computation time) are demonstrated on the longitudinal

control of an Airbus passenger aircraft. Chapter 3 appeared in the *AIAA Journal of Guidance, Control, and Dynamics (JGCD)* [22].

- Chapter 4 presents SVR-AMA, a novel first-order solver suitable for MPC problems, and its accelerated versions. Compared to the approach presented in the previous chapter, SVR-AMA performs random updates of the decision variables of the MPC problem (a useful feature to have more flexibility in the computation of the MPC problem solution when synchronized parallel hardware architectures are not available). The numerical results on the longitudinal control of an Airbus passenger aircraft show improvements (in terms of quality of the solution returned within the same number of iterations) compared to AMA. Chapter 4 has been submitted for review to the *IEEE Transactions on Automatic Control (TAC)* [26].

- Chapter 5 presents an algorithm for the detection and active diagnosis of elevator jamming faults (faults that can cause either temporary or permanent jamming of the aircraft elevators). Compared to traditional fault-tolerant control approaches, the proposed algorithm does not assume perfect knowledge of the fault. The interactions between the fault detection (FD) unit and the MPC controller are designed to allow the FD unit to identify the root cause of the actuator jamming (i.e., to identify whether the jamming is permanent or temporary) through a sequence of tailored reconfigurations in the MPC controller. This strategy, which strongly relies on features specific to MPC, is tested on the longitudinal control of an Airbus passenger aircraft. Chapter 5 has been submitted for review to the *International Journal of Robust and Nonlinear Control (IJRNC)* [27].

- Chapter 6 concludes the dissertation, summarizing key results, analyzing current limitations, and highlighting opportunities for future research directions.

# 2 An Adaptive Constraint Tightening Approach to Linear Model Predictive Control Based on Approximation Algorithms for Optimization


## Abstract

In this chapter, we propose a model predictive control scheme for discrete-time linear invariant systems based on inexact numerical optimization algorithms. We assume that the solution of the associated quadratic program produced by some numerical algorithm is possibly neither optimal nor feasible, but the algorithm is able to provide estimates on primal suboptimality and primal feasibility violation. By adaptively tightening the complicating constraints, we can ensure the primal feasibility of the approximate solutions generated by the algorithm. We derive a control strategy that has the following properties: the constraints on the states and inputs are satisfied, asymptotic stability of the closed-loop system is guaranteed, and the number of iterations needed for a desired level of suboptimality can be determined. The proposed method is illustrated using a simulated longitudinal flight control problem.

## 2.1 Introduction

Model predictive control (MPC) has become a popular advanced control technology due to its ability to handle hard input and state constraints. An MPC scheme consists of solving at each sampling time instant (i.e., when the controller receives new measurements from the plant) an optimization problem whose variables are given by the inputs and the states of the system over a finite time horizon. Once the optimal solution is computed, only the first input is injected to the system and then the whole procedure is repeated. MPC was first implemented in slow systems such as industrial processes [99], but due to the increase of computing power and data transmission capabilities of modern digital devices it has been extensively studied also in the context of controlling fast embedded systems and distributed control of networked systems. Recently there has been a growing interest in developing faster MPC schemes for embedded systems, by improving the computational efficiency and providing worst-case computational complexity certificates for the applied solution methods, making these schemes implementable on hardware with limited computational power [49; 54; 53; 75; 79; 101; 107; 108]. In large-scale networked system settings many decomposition methods have been proposed for the synthesis of distributed MPC schemes as well [9; 74; 77; 75; 76; 114; 125].

The two main classes of general-purpose iterative algorithms that have typically been used for solving the underlying linearly constrained convex quadratic programs in MPC problems are the active-set and interior point methods [2]. Recently, however, there has been interest in applying first-order (projected gradient) methods to the quadratic programming problems (QPs) arising in MPC. In particular, Nesterov's Fast Gradient Method (FGM) has been applied to MPC in [75; 79; 108; 145]. As well as being a very simple algorithm to implement, and being very parallelizable (the main computational load being a matrix-vector multiplication), and essentially division-free and matrix-inversion-free, the key benefit is that a tight bound can be found on the number of iterations required to achieve a given degree of solution accuracy [75; 79; 108]. The primal FGM is well suited to implementation using fixed-point arithmetic [145; 53]. Further recent developments include combination with the Alternating Direction of Multipliers Method (ADMM) [79; 107] to handle equality constraints (allowing the optimizer to use both the states and the control commands as decision variables), solution of the Lagrangian dual problem rather than the primal problem [75; 79; 91], and applications to distributed model predictive control problems [32; 75]. An online tool, FiOrDoS, [129] can automatically generate C-code for first-order methods, customized to specific problem structures.

Typical requirements for the practical implementation of real-time MPC include certification of the worst-case execution time, reduced memory usage, simple numerical iterations that can be easily implemented on cheap and/or certifiable hardware and software, and distributed computations. Classic approaches meeting these requirements such as explicit MPC [4], or methods based on interior point algorithms [101] can become inapplicable due to the large dimension of the problems or

complex iterations that involve matrix inversion. An alternative is provided by dual first-order methods [107; 76; 79; 75]. Although these methods are characterized by simple computations and offer tight worst-case bounds on the required number of iterations, they can ensure feasibility only at optimality [79]. In order to avoid this drawback, new dual methods based on constraint tightening have been proposed in [14; 32; 75; 112]. The authors of [32; 112] present suboptimal stable MPC schemes able to ensure also feasibility of the primal variables using a constraint tightening approach. In these schemes the tightening is applied to both the state and the input constraints, while the parameters measuring the suboptimality and the degree of tightening are fixed for all initial states of the MPC scheme. In [14], stability and feasibility of an MPC scheme is also ensured using a constraint tightening approach and suboptimality results are based on dual subgradient analysis. In [75] the authors derive a complete convergence rate analysis for two dual methods based on inexact gradient information and averaging that generate approximate primal solutions for smooth convex optimization problems. Further, they combine these methods with constraint tightening and apply this framework to MPC. In the present chapter we extend the main results from [75] on suboptimality and feasibility of a suboptimal MPC scheme by assuming that the suboptimal control inputs are computed using a generic inexact numerical optimization algorithm. Moreover, compared to [75], we show that by choosing adaptively the parameters of the suboptimal MPC scheme we can ensure closed-loop stability.

*Contribution:* The main contribution of this chapter is to propose a suboptimal MPC scheme that ensures both feasibility and stability with only a limited number of optimization iterations. Given that in many MPC schemes an approximate solution of the optimization problem that has to be solved online, that is, at each sampling time instant when new measurements are available from the plant, might not be feasible, we solve approximately an auxiliary problem obtained by tightening the constraints of the original one. We show that the approximate solution of the tightened problem is also a suboptimal feasible solution for our original optimization problem and thus we obtain an MPC scheme that ensures feasibility, suboptimality, and stability for the closed-loop system. Compared to the recent papers [112; 32], in our approach the tightening is applied only to the state constraints leaving the original control input limitations unaltered. Furthermore, the parameters measuring the suboptimality and the tightening are chosen adaptively, that is, depending on the initial state of the MPC scheme. This leads to a more flexible and potentially less conservative approach. The proposed MPC scheme can accommodate any QP solver in order to find an approximate solution of the problem. Further, in order to establish a bound on the number of iterations required to find a desired solution, we also provide a dual fast gradient method taken from [75], which is at least one order of magnitude faster than the one in [14].

Aerospace applications represent an area where embedded optimization-based control solutions have received increased attention. In particular, within the area of flight control, MPC has been used for diverse purposes including regulation and

tracking [57], control reconfiguration [59], flight envelope protection [58], gust load alleviation [39], and optimal control surface allocation [17]. In this chapter, we demonstrate the applicability of our proposed inexact MPC scheme on a medium-scale, nontrivial longitudinal flight control application problem involving two control inputs, 9 states, and around 1000 constraints.

*Chapter outline:* The chapter is organized as follows. In Section 2.2 we formulate the linear MPC problem and we also introduce a tightened problem that helps us to recover feasibility. In Section 2.3 we present a dual fast gradient algorithm for finding the approximate solution required by our MPC scheme and provide a rate analysis and estimates on the primal feasibility violation and suboptimality of the generated approximate primal solutions. In Section 2.4 we prove the feasibility, suboptimality, and stability of the proposed MPC scheme. Finally, in Section 2.5 we apply our scheme to a flight control problem involving altitude and airspeed regulation of an F-16 aircraft.

*Notation:* We work in the space $\mathbb{R}^n$ composed by column vectors. For $u, v \in \mathbb{R}^n$ we denote the standard Euclidean inner product $\langle u, v\rangle = \sum_{i=1}^n u_i v_i$, the Euclidean norm $\|u\| = \sqrt{\langle u, u\rangle}$, and the projection onto the nonnegative orthant $\mathbb{R}^n_+$ as $[u]_+$. We use the same notations $\langle\cdot,\cdot\rangle$, $\|\cdot\|$, and $[\cdot]_+$ for spaces of different dimension. Further, $Q \succ (\succeq)\, 0$ denotes a positive (semi)definite matrix $Q$. For a matrix $G$, we denote its spectral norm by $\|G\|$.

## 2.2 Condensed QP Formulation for Linear MPC

We consider discrete-time linear systems, defined by the following linear difference equation:

$$x(t+1) = Ax(t) + Bu(t), \tag{2.1}$$

where $x(t) \in \mathbb{R}^{n_x}$ and $u(t) \in \mathbb{R}^{n_u}$ represent the state and the input of the system at time $t$, respectively. We also impose state and input constraints:

$$x(t) \in X, \quad u(t) \in U \quad \forall t \geq 0, \tag{2.2}$$

where $X \subseteq \mathbb{R}^{n_x}$ is a polyhedral set and $U \subseteq \mathbb{R}^{n_u}$ is a simple convex set, i.e., the projection on this set can be computed efficiently (e.g. hyperbox, Euclidean ball, $\mathbb{R}^{n_u}$, etc). Moreover, we assume that both sets $X$ and $U$ contain the origin in their interior. For the system (2.1) we consider a quadratic convex stage cost:

$$\ell(x(t), u(t)) = \frac{1}{2}\|x(t)\|_Q^2 + \frac{1}{2}\|u(t)\|_R^2,$$

where we use the notation $\|x\|_Q^2 = x^{\mathrm{T}} Q x$. The following assumption is valid throughout the chapter:

**Assumption 2.2.1.** *The pair $(A, B)$ is stabilizable and the matrices $Q$ and $R$ are positive definite, i.e. $Q \succ 0$ and $R \succ 0$.*

Based on Assumption 2.2.1 we denote with $K \in \mathbb{R}^{n_u \times n_x}$ the gain associated with the infinite horizon linear quadratic regulator (LQR) defined by the matrices $A, B, Q$ and $R$ and with $P$ the solution of the algebraic Riccati equation associated with the LQR problem. We also introduce a terminal cost:

$$V^{\mathrm{f}}(x) = \frac{1}{2}\|x\|_P^2,$$

and a terminal polyhedral set $X^{\mathrm{f}}$. We assume that the terminal set $X^{\mathrm{f}}$ is $\mu$-contractive for the closed-loop system $x(t+1) = (A+BK)x(t)$, not necessarily maximal, with $\mu < 1$ (see e.g. [103] for a detailed discussion), i.e.:

$$\forall x \in X^{\mathrm{f}} \Rightarrow x \in X, Kx \in U \text{ and } (A+BK)x \in \mu X^{\mathrm{f}}. \tag{2.3}$$

For a prediction horizon of length $N$, the MPC problem for (2.1), with a given initial state $x \in \bar{X}_N$, where $\bar{X}_N$ denotes a region of attraction, can be formulated as (see e.g. [118; 103] for more details):

$$\begin{aligned} V^*(x) = & \min_{x(i),u(i)} \sum_{i=0}^{N-1} \ell(x(i),u(i)) + V^{\mathrm{f}}(x(N)) \\ \text{s.t: } & x(i+1) = Ax(i) + Bu(i),\ x(0) = x \\ & x(i) \in X,\ u(i) \in U,\ x(N) \in X^{\mathrm{f}} \quad \text{for } i = 0, \ldots, N-1. \end{aligned} \tag{2.4}$$

For the input trajectory of the system we use the notation:

$$\mathbf{u} = \left[u(0)^{\mathrm{T}} \cdots u(N-1)^{\mathrm{T}}\right]^{\mathrm{T}} \in \mathbb{R}^{Nn_u}.$$

By eliminating the states from the dynamics (2.1), the MPC problem (2.4) can be expressed as a condensed quadratic convex optimization problem [103; 75]:

$$\begin{aligned} V^*(x) = & \min_{\mathbf{u} \in \mathbf{U}} V_N(x, \mathbf{u}) \quad \left( = \frac{1}{2}\mathbf{u}^{\mathrm{T}}\mathbf{Q}\mathbf{u} + (\mathbf{W}x)^{\mathrm{T}}\mathbf{u} \right) \\ & \text{s.t: } \mathbf{G}\mathbf{u} + \mathbf{E}x + \mathbf{g} \leq 0, \end{aligned} \tag{$\mathbf{P}(x)$}$$

where $\mathbf{Q}$ is positive definite due to the assumption that $R$ is positive definite and the convex set $\mathbf{U}$ is the Cartesian product of the sets $U$ for $N$ times. Note that if the set $U$ is simple, then the Cartesian product set $\mathbf{U}$ is also a simple set. Further, the inequalities $\mathbf{G}\mathbf{u} + \mathbf{E}x + \mathbf{g} \leq 0$ are obtained by eliminating the states from the constraints $x(i) \in X$ and $x(N) \in X^{\mathrm{f}}$ and they are usually called *complicating constraints*. Here we consider $\mathbf{G} \in \mathbb{R}^{p \times Nn_u}$. In MPC, at each discrete time instant, given the initial state $x$, we need to solve the optimization problem (2.4) or equivalently optimization problem ($\mathbf{P}(x)$). We denote by $\mathbf{u}^*(x)$ the unique optimal solution and by $\lambda^*(x)$ an optimal Lagrange multiplier associated to the complicating constraints of problem ($\mathbf{P}(x)$). Further, we denote by $\mathbf{u}^{\mathrm{f}}(x)$ the LQR solution:

$$\mathbf{u}^{\mathrm{f}}(x) = [(Kx(0))^{\mathrm{T}} \cdots (Kx(N-1))^{\mathrm{T}}]^{\mathrm{T}},$$

where $x(0) = x$ and $x(i+1) = (A + BK)x(i)$ for all $i = 0, \ldots, N-1$. In any practical situations, e.g., when we have fast dynamics and hard real-time computational requirements, or when we need to perform distributed computations, finding the solution $\mathbf{u}^*(x)$ of ($\mathbf{P}(x)$) is difficult. Thus, we assume that we have available an optimization algorithm that can deliver in a computationally predictable way an approximate $\epsilon$-solution $\bar{\mathbf{u}}(x) = \text{Alg}((\mathbf{P}(x)), \epsilon)$, with $\epsilon > 0$, such that:

$$\bar{\mathbf{u}}(x) = \mathbf{u}^{\text{f}}(x) \quad \text{if } x \in X^{\text{f}} \tag{2.5}$$

or

$$\bar{\mathbf{u}}(x) \in \mathbf{U}, \ \|[\mathbf{G}\bar{\mathbf{u}}(x) + \mathbf{E}x + \mathbf{g}]_+\| \leq \epsilon \text{ and } |V_N(x, \bar{\mathbf{u}}(x)) - V^*(x)| \leq \epsilon. \tag{2.6}$$

We note that in this setting, the approximate solution $\bar{\mathbf{u}}(x)$ is indeed suboptimal for the MPC scheme but it may also be infeasible since the constraints $\mathbf{G}\bar{\mathbf{u}}(x) + \mathbf{E}x + \mathbf{g} \leq 0$ might not be satisfied. In many applications, such as the MPC problem, the constraints typically represent physical limitations of actuators, or safety limits and operating conditions of the controlled plant. Thus, ensuring the feasibility of the primal variables, that are, $\mathbf{u} \in \mathbf{U}$ and $\mathbf{G}\mathbf{u} + \mathbf{E}x + \mathbf{g} \leq 0$, becomes a critical requirement. We will see further how we can modify the original problem ($\mathbf{P}(x)$) in order to find an approximate optimal solution that is also feasible.

In our proposed approach, instead of solving the original problem ($\mathbf{P}(x)$), we consider a tightened version (similarly to [14; 32; 75; 112]). We introduce the following tightened problem using $\epsilon_{\text{c}} > 0$ associated with the original problem ($\mathbf{P}(x)$):

$$\begin{aligned} V^*_{\epsilon_{\text{c}}}(x) = \min_{\mathbf{u} \in \mathbf{U}} \ & V_N(x, \mathbf{u}) \quad \left( = \frac{1}{2}\, \mathbf{u}^{\text{T}}\mathbf{Q}\mathbf{u} + (\mathbf{W}x)^{\text{T}}\mathbf{u} \right) \\ \text{s.t:} \ & \mathbf{G}\mathbf{u} + \mathbf{E}x + \mathbf{g} + \epsilon_{\text{c}}\mathbf{e} \leq 0, \end{aligned} \tag{$\mathbf{P}_{\epsilon_{\text{c}}}(x)$}$$

where $\mathbf{e}$ denotes the vector with all entries 1. We state first the following assumption:

**Assumption 2.2.2.** *For any $x \in X_N \subseteq \bar{X}_N$ there exists a strictly feasible vector $\tilde{\mathbf{u}}(x)$ for problem ($\mathbf{P}(x)$), i.e. there exists $\tilde{\mathbf{u}}(x) \in \mathbf{U}$ satisfying $\mathbf{G}\tilde{\mathbf{u}}(x) + \mathbf{E}x + \mathbf{g} < 0$.*

Based on Assumption 2.2.2, we choose $\epsilon_{\text{c}}$ to satisfy e.g., the following inequality:

$$0 < \epsilon_{\text{c}} \leq \frac{1}{2} \min_{j=1,\ldots,p} \{ -(\mathbf{G}\tilde{\mathbf{u}}(x) + \mathbf{E}x + \mathbf{g})_j \}, \tag{2.7}$$

with $\tilde{\mathbf{u}}(x)$ being a strictly feasible vector for ($\mathbf{P}(x)$).

*Remark* 2.2.1. Note that for this range of $\epsilon_{\text{c}}$, the input sequence $\tilde{\mathbf{u}}(x)$ is also a strictly feasible vector for problem ($\mathbf{P}_{\epsilon_{\text{c}}}(x)$), so that Assumption 2.2.2 still holds for this problem. This observation is relevant to derive the results described in Section 2.4.

It is important to note that in our approach we apply the tightening only to the state constraints. Thus, our approach is usually less restrictive than the approaches

in [112; 32] where the tightening procedure is applied to both state and input constraints. It is straightforward to establish that both problems ($\mathbf{P}(x)$) and ($\mathbf{P}_{\epsilon_c}(x)$) are convex quadratic programs with a strongly convex objective function since the Hessian $\mathbf{Q}$ is positive definite. Thus, without loss of generality we assume that for finding an $\epsilon$-solution of problem ($\mathbf{P}_{\epsilon_c}(x)$) we can invoke the same algorithm as for finding $\bar{\mathbf{u}}(x)$, i.e., $\bar{\mathbf{u}}_{\epsilon_c}(x) = \text{Alg}((\mathbf{P}_{\epsilon_c}(x)), \epsilon)$.

If $x \in X^{\text{f}}$ we do not need to introduce the tightened problem ($\mathbf{P}_{\epsilon_c}(x)$) since in this case we have:

$$\bar{\mathbf{u}}(x) = \mathbf{u}^*(x) = \mathbf{u}^{\text{f}}(x), \tag{2.8}$$

which is feasible and optimal for problem ($\mathbf{P}(x)$). In this case, it is also known that the value of the cost function is equal to the value of the terminal cost, i.e.:

$$V_N(x, \bar{\mathbf{u}}(x)) = V^*(x) = V^{\text{f}}(x). \tag{2.9}$$

## 2.3 Dual Fast Gradient Algorithm for Convex Problems

In this section we present a dual fast gradient method that can be applied for finding an $\epsilon$-solution of the optimization problem ($\mathbf{P}(x)$) or ($\mathbf{P}_{\epsilon_c}(x)$). There are different versions of fast gradient methods, but in this chapter we consider Nesterov's scheme from [82; 75]. Since the algorithm can be applied to a wider class of problems we introduce first the following convex optimization problem:

$$V^* = \min_{\mathbf{u} \in \mathbf{U}} \left\{ V(\mathbf{u}) : \ \mathbf{G}\mathbf{u} + \mathbf{g} \leq 0 \right\}, \tag{2.10}$$

where $V : \mathbb{R}^{Nn_u} \to \mathbb{R}$ is a $\sigma_{\text{V}}$-strongly convex function (i.e., $V$ is a strongly convex function with convexity parameter $\sigma_{\text{V}} > 0$), $\mathbf{U} \subseteq \mathbb{R}^{Nn_u}$ is a simple convex set, as assumed in Section 2.2, $\mathbf{G} \in \mathbb{R}^{p \times Nn_u}$, and $\mathbf{g} \in \mathbb{R}^p$. We can notice that since $\mathbf{G}\mathbf{u} + \mathbf{g} \leq 0$ (the complicating constraints) is a general polyhedron, the projection on this set is hard to compute, but the set $\mathbf{U}$ is simple (e.g., hyperbox, Euclidean ball, $\mathbb{R}^{Nn_u}$, etc.), i.e., the projection on this set can be computed very efficiently. We note that problems ($\mathbf{P}(x)$) and ($\mathbf{P}_{\epsilon_c}(x)$) are particular cases of problem (2.10) with $V$ being a convex quadratic function, and in this case $\sigma_{\text{V}} = \lambda_{\min}(\mathbf{Q})$.
By moving the complicating constraints into the cost via Lagrange multipliers we define the dual function:

$$d(\lambda) = \min_{\mathbf{u} \in \mathbf{U}} \mathcal{L}(\mathbf{u}, \lambda), \tag{2.11}$$

where $\mathcal{L}(\mathbf{u}, \lambda) = V(\mathbf{u}) + \langle \lambda, \mathbf{G}\mathbf{u} + \mathbf{g} \rangle$ denotes the partial Lagrangian w.r.t. the complicating constraints $\mathbf{G}\mathbf{u} + \mathbf{g} \leq 0$. We also denote by $\mathbf{u}(\lambda)$ the optimal solution of the *inner problem*:

$$\mathbf{u}(\lambda) = \arg\min_{\mathbf{u} \in \mathbf{U}} \mathcal{L}(\mathbf{u}, \lambda). \tag{2.12}$$

**Algorithm 2.1** Dual fast gradient algorithm (DFG)

1: Set $\lambda^0 = 0$
2: **for** $k = 0$ to $\bar{k}$ **do**
3: Compute: $\mathbf{u}^k = \arg\min_{\mathbf{u}\in\mathbf{U}} \mathcal{L}(\mathbf{u}, \lambda^k)$
4: Compute: $\hat{\lambda}^k = \left[\lambda^k + \frac{1}{L_\mathrm{d}}(\mathbf{G}\mathbf{u}^k + \mathbf{g})\right]_+$
5: Compute: $\lambda^{k+1} = \frac{k+1}{k+3}\hat{\lambda}^k + \frac{2}{L_\mathrm{d}(k+3)}\left[\sum_{s=0}^{k} \frac{s+1}{2}(\mathbf{G}\mathbf{u}^s + \mathbf{g})\right]_+$,
6: **end for**

Since $V$ is strongly convex, it can be proven that the gradient of the dual function $d(\lambda)$ is given by:

$$\nabla d(\lambda) = \mathbf{G}\mathbf{u}(\lambda) + \mathbf{g}, \tag{2.13}$$

and it is Lipschitz continuous with constant $L_\mathrm{d} = \frac{\|\mathbf{G}\|^2}{\sigma_\mathrm{v}}$ (see [76; 75] for more general settings). If we assume that strong duality holds, we have for the *outer problem*:

$$V^* = \max_{\lambda\geq 0} d(\lambda), \tag{2.14}$$

for which we denote an optimal solution by $\lambda^*$. The next lemma provides bounds for function $d(\lambda)$ in terms of a linear and a quadratic model that use information of the dual function and of its gradient.

**Lemma 2.3.1** ([76; 75])**.** *Let $V$ be strongly convex and for a given $\lambda$ let $\mathbf{u}(\lambda)$ be the optimal solution of* (2.12)*. Then, the following inequalities are valid:*

$$0 \geq d(\mu) - [d(\lambda) + \langle \nabla d(\lambda), \mu - \lambda\rangle] \geq -\frac{L_d}{2}\|\mu - \lambda\|^2 \quad \forall \mu \in \mathbb{R}^p_+.$$

Algorithm 2.1 summarizes the dual fast gradient scheme we consider. Recall that $\nabla d(\lambda^k) = \mathbf{G}\mathbf{u}^k + \mathbf{g}$ in Algorithm 2.1. A complete analysis of this algorithm, in particular when the inner subproblems are solved inexactly with an initial iterate $\lambda^0 \neq 0$, can be found in [76; 75]. Note that we can start our algorithm also from a Lagrange multiplier $\lambda^0 \neq 0$ (see e.g., [76; 75]), but for simplicity of the exposition we present here Algorithm 2.1 for $\lambda^0 = 0$. Let us now define the following average sequence for the primal variables:

$$\hat{\mathbf{u}}^k = \sum_{s=0}^{k} \frac{2(s+1)}{(k+1)(k+2)}\mathbf{u}^s. \tag{2.15}$$

Our main goal in this section is to derive estimates on primal feasibility violation and suboptimality in the form (2.6) for our problem (2.10) using the averaged primal sequence $\hat{\mathbf{u}}^k$ obtained by the proposed Algorithm 2.1 (see [75] for a complete analysis of Algorithm 2.1).

**Theorem 2.3.1.** *Let $V$ be strongly convex, the sequences $\left(\mathbf{u}^k, \hat{\lambda}^k, \lambda^k\right)_{k\geq 0}$ be generated by Algorithm 2.1 and $\hat{\mathbf{u}}^k$ be given by* (2.15)*. Then, an estimate on the primal feasibility violation for the original problem* (2.10) *is given by:*

$$\|[\mathbf{G}\hat{\mathbf{u}}^k + \mathbf{g}]_+\| \leq \frac{8L_d R_d}{(k+1)^2},$$

*where $R_d = \|\lambda^*\|$. Moreover, an estimate on primal suboptimality is given by:*

$$-\frac{8L_d R_d^2}{(k+1)^2} \leq V(\hat{\mathbf{u}}^k) - V^* \leq 0.$$

*Proof.* From [76, Theorem 3.4], we have the following inequality which will help us to establish the convergence properties of our proposed Algorithm 2.1:

$$\frac{(k+1)(k+2)}{4} d(\hat{\lambda}^k) \geq$$
$$\max_{\lambda\geq 0}\left[-\frac{L_{\mathrm{d}}}{2}\|\lambda\|^2 + \sum_{s=0}^{k}\frac{s+1}{2}\left(d(\lambda^s) + \langle \nabla d(\lambda^s), \lambda - \lambda^s\rangle\right)\right] \quad \forall \lambda \in \mathbb{R}^p_+.$$

Rearranging the terms in the previous inequality and taking into account that $d(\lambda^s) = V(\mathbf{u}^s) + \langle \mathbf{G}\mathbf{u}^s + \mathbf{g}, \lambda^s\rangle$ and $\nabla d(\lambda^s) = \mathbf{G}\mathbf{u}^s + \mathbf{g}$ we get:

$$\begin{aligned} d(\hat{\lambda}^k) &\geq \max_{\lambda\geq 0}\left[-\frac{2L_{\mathrm{d}}}{(k+1)^2}\|\lambda\|^2\right] + \\ &\quad \max_{\lambda\geq 0}\left[\sum_{s=0}^{k}\frac{2(s+1)}{(k+1)(k+2)}\left(d(\lambda^s) - \langle \mathbf{G}\mathbf{u}^s + \mathbf{g}, \lambda^s\rangle + \langle \mathbf{G}\mathbf{u}^s + \mathbf{g}, \lambda\rangle\right)\right] \\ &= \max_{\lambda\geq 0}\left[-\frac{2L_{\mathrm{d}}}{(k+1)^2}\|\lambda\|^2 + \sum_{s=0}^{k}\frac{2(s+1)}{(k+1)(k+2)}V(\mathbf{u}^s) + \langle \mathbf{G}\hat{\mathbf{u}}^k + \mathbf{g}, \lambda\rangle\right] \\ &\geq \max_{\lambda\geq 0}\left[-\frac{2L_{\mathrm{d}}}{(k+1)^2}\|\lambda\|^2 + \langle \mathbf{G}\hat{\mathbf{u}}^k + \mathbf{g}, \lambda\rangle\right] + V(\hat{\mathbf{u}}^k), \end{aligned}$$

where in the last inequality we used convexity of $V$. Further, we can write:

$$\max_{\lambda\geq 0}\left[-\frac{2L_{\mathrm{d}}}{(k+1)^2}\|\lambda\|^2 + \langle \lambda, \mathbf{G}\hat{\mathbf{u}}^k + \mathbf{g}\rangle\right] \leq d(\hat{\lambda}^k) - V(\hat{\mathbf{u}}^k). \tag{2.16}$$

For the term in the right-hand side we have:

$$\begin{aligned} d(\hat{\lambda}^k) - V(\hat{\mathbf{u}}^k) &\leq d(\lambda^*) - V(\hat{\mathbf{u}}^k) = \min_{\mathbf{u}\in\mathbf{U}} V(\mathbf{u}) + \langle \lambda^*, \mathbf{G}\mathbf{u} + \mathbf{g}\rangle - V(\hat{\mathbf{u}}^k) \\ &\leq V(\hat{\mathbf{u}}^k) + \langle \lambda^*, \mathbf{G}\hat{\mathbf{u}}^k + \mathbf{g}\rangle - V(\hat{\mathbf{u}}^k) = \langle \lambda^*, \mathbf{G}\hat{\mathbf{u}}^k + \mathbf{g}\rangle \quad (2.17) \\ &\leq \langle \lambda^*, [\mathbf{G}\hat{\mathbf{u}}^k + \mathbf{g}]_+\rangle \leq \|\lambda^*\|\|[\mathbf{G}\hat{\mathbf{u}}^k + \mathbf{g}]_+\|, \end{aligned}$$

where we used that $\lambda^* \geq 0$ and the Cauchy-Schwartz inequality. By evaluating the left-hand side term in (2.16) and taking into account that $\langle [v]_+, v - [v]_+ \rangle = 0$ we obtain the following expression:

$$\max_{\lambda \geq 0} \left[ -\frac{2L_{\mathrm{d}}}{(k+1)^2} \|\lambda\|^2 + \langle \lambda, \mathbf{G}\hat{\mathbf{u}}^k + \mathbf{g} \rangle \right] = \frac{(k+1)^2}{8L_{\mathrm{d}}} \|[\mathbf{G}\hat{\mathbf{u}}^k + \mathbf{g}]_+\|^2. \quad (2.18)$$

Substituting now (2.17) and (2.18) into (2.16) we get the first statement of the theorem. In order to prove the left-hand side inequality in the primal suboptimality formula we can write:

$$\begin{aligned} V^* = d(\lambda^*) &= \min_{\mathbf{u} \in \mathbf{U}} V(\mathbf{u}) + \langle \lambda^*, \mathbf{G}\mathbf{u} + \mathbf{g} \rangle \leq V(\hat{\mathbf{u}}^k) + \langle \lambda^*, \mathbf{G}\hat{\mathbf{u}}^k + \mathbf{g} \rangle \\ &\leq V(\hat{\mathbf{u}}^k) + \left\langle \lambda^*, [\mathbf{G}\hat{\mathbf{u}}^k + \mathbf{g}]_+ \right\rangle \leq V(\hat{\mathbf{u}}^k) + \|\lambda^*\| \, \|[\mathbf{G}\hat{\mathbf{u}}^k + \mathbf{g}]_+\| \\ &= V(\hat{\mathbf{u}}^k) + R_{\mathrm{d}} \|[\mathbf{G}\hat{\mathbf{u}}^k + \mathbf{g}]_+\|, \end{aligned}$$

which together with the first part of the theorem leads to the result. In order to prove the right-hand side inequality we use (2.16):

$$V(\hat{\mathbf{u}}^k) - d(\hat{\lambda}^k) \leq -\max_{\lambda \geq 0} -\frac{2L_{\mathrm{d}}}{(k+1)^2} \|\lambda\|^2 + \langle \lambda, \mathbf{G}\hat{\mathbf{u}}^k + \mathbf{g} \rangle \overset{\lambda=0}{\leq} 0.$$

Taking now into account that $d(\hat{\lambda}^k) \leq V^*$ we get the result. □

An immediate consequence of the previous theorem is the following result: if we take $\bar{k} = \left\lfloor 2\sqrt{\frac{2L_{\mathrm{d}}R_{\mathrm{d}}}{\epsilon}} \right\rfloor$ iterations, we obtain the following estimates on feasibility violation and primal suboptimality:

$$\|[\mathbf{G}\hat{\mathbf{u}}^{\bar{k}} + \mathbf{g}]_+\| \leq \epsilon \quad \text{and} \quad -R_{\mathrm{d}}\epsilon \leq V(\hat{\mathbf{u}}^{\bar{k}}) - V^* \leq 0.$$

Thus, if we redefine $\epsilon = \max\{\epsilon, R_{\mathrm{d}}\epsilon\}$ we can conclude that $\hat{\mathbf{u}}^{\bar{k}}$ is an $\epsilon$-solution for problem (2.10) satisfying (2.6). In other words, Algorithm 2.1 can be used for finding an $\epsilon$-solution of MPC problem ($\mathbf{P}(x)$) or of the tightened MPC problem ($\mathbf{P}_{\epsilon_{\mathrm{c}}}(x)$). We will discuss further how we can recover the feasibility, suboptimality, and stability of our suboptimal MPC scheme in the case when $x \notin X^{\mathrm{f}}$.

## 2.4 Suboptimality, Feasibility, and Stability of the Model Predictive Control Scheme

As mentioned before, at each time step of the MPC scheme, given the initial state $x$, instead of applying the algorithm $\mathrm{Alg}((\mathbf{P}(x)), \epsilon)$ (e.g. Algorithm 2.1 of the previous section) to provide an $\epsilon$-solution $\bar{\mathbf{u}}(x)$ of the optimization problem ($\mathbf{P}(x)$) we apply $\mathrm{Alg}((\mathbf{P}_{\epsilon_{\mathrm{c}}}(x)), \epsilon)$ for finding an $\epsilon$-solution $\bar{\mathbf{u}}_{\epsilon_{\mathrm{c}}}(x)$ of the tightened problem ($\mathbf{P}_{\epsilon_{\mathrm{c}}}(x)$). However, since we are interested in obtaining an approximate primal solution that

may be suboptimal for the original problem ($\mathbf{P}(x)$) but certainly primal feasible, we need to find first a relation between optimal values $V^*_{\epsilon_c}(x)$ and $V^*(x)$. In order to find such a relation, let us denote by $\lambda^*_{\epsilon_c}(x)$ an optimal Lagrange multiplier associated with the complicating constraints in problem ($\mathbf{P}_{\epsilon_c}(x)$). Then, the following upper bound on $\|\lambda^*_{\epsilon_c}(x)\|$ can be established:

**Lemma 2.4.1.** *Let $\lambda^*_{\epsilon_c}(x)$ denote an optimal Lagrange multiplier associated with the inequality constraints $\mathbf{G}\mathbf{u} + \mathbf{E}x + \mathbf{g} + \epsilon_c\mathbf{e} \leq 0$ in ($\mathbf{P}_{\epsilon_c}(x)$). Then, the following upper bound on $\|\lambda^*_{\epsilon_c}(x)\|$ can be established:*

$$\|\lambda^*_{\epsilon_c}(x)\| \leq 2\mathcal{R}_d,$$

*where*

$$\mathcal{R}_d = \frac{V_N(x, \tilde{\mathbf{u}}) - d(\tilde{\lambda})}{\min\limits_{j=1,\ldots,p}\{-\left(\mathbf{G}\tilde{\mathbf{u}} + \mathbf{E}x + \mathbf{g}\right)_j\}}, \tag{2.19}$$

*and $\tilde{\mathbf{u}}$ denotes a strictly feasible vector for problem ($\mathbf{P}(x)$) (see Assumption 2.2.2), $d(\cdot)$ denotes the dual function of ($\mathbf{P}(x)$) with respect to the inequality constraints $\mathbf{G}\tilde{\mathbf{u}} + \mathbf{E}x + \mathbf{g} \leq 0$ and $\tilde{\lambda} \in \mathbb{R}^p_+$.*

*Proof.* First, let us denote by $\mathcal{L}(x, \mathbf{u}, \lambda)$ and $\mathcal{L}_{\epsilon_c}(x, \mathbf{u}, \lambda)$ the partial Lagrangian with respect to the complicating constraints in problem ($\mathbf{P}(x)$) and ($\mathbf{P}_{\epsilon_c}(x)$), respectively. Using Lemma 1 in [80] we can write:

$$\begin{aligned}
\|\lambda^*_{\epsilon_c}(x)\| &\leq \frac{V_N(x, \tilde{\mathbf{u}}) - \min_{\mathbf{u}\in\mathbf{U}} \mathcal{L}_{\epsilon_c}(x, \mathbf{u}, \tilde{\lambda})}{\min\limits_{j=1,\ldots,p}\{-\left(\mathbf{G}\tilde{\mathbf{u}} + \mathbf{E}x + \mathbf{g} + \epsilon_c\right)_j\}} \\
&= \frac{\left[V_N(x, \tilde{\mathbf{u}}) - \min_{\mathbf{u}\in\mathbf{U}} \mathcal{L}(x, \mathbf{u}, \tilde{\lambda})\right] - \left\langle \tilde{\lambda}, \epsilon_c\mathbf{e}\right\rangle}{\min\limits_{j=1,\ldots,p}\{-\left(\mathbf{G}\tilde{\mathbf{u}} + \mathbf{E}x + \mathbf{g}\right)_j\} - \epsilon_c} \\
&\leq 2\mathcal{R}_d \qquad \forall x \in X_N,
\end{aligned} \tag{2.20}$$

where in the last inequality we used (2.7) and the fact that both $\tilde{\lambda}$ and $\epsilon_c$ are nonnegative. □

Note that for computing the bound $\mathcal{R}_d$ we have the freedom of choosing $\tilde{\lambda} \in \mathbb{R}^p$ and $\tilde{\mathbf{u}}$. Thus, we can obtain a small enough bound on the norm of Lagrange multipliers $\lambda^*_{\epsilon_c}(x)$.

Taking now into account that

$$\{\mathbf{u} : \mathbf{G}\mathbf{u} + \mathbf{E}x + \mathbf{g} + \epsilon_c\mathbf{e} \leq 0\} \subseteq \{\mathbf{u} : \mathbf{G}\mathbf{u} + \mathbf{E}x + \mathbf{g} \leq 0\},$$

we have on the one hand:

$$V^*_{\epsilon_c}(x) \geq V^*(x). \tag{2.21}$$

On the other hand, from the the dual formulation of the tightened problem ($\mathbf{P}_{\epsilon_c}(x)$) we have:

$$\begin{aligned} V^*_{\epsilon_c}(x) &= \min_{\mathbf{u}\in\mathbf{U}} V_N(x,\mathbf{u}) + \left\langle \lambda^*_{\epsilon_c}(x), \mathbf{G}\mathbf{u}+\mathbf{E}x+\mathbf{g}+\epsilon_c\mathbf{e}\right\rangle \\ &\leq \max_{\lambda\geq 0}\min_{\mathbf{u}\in\mathbf{U}} V_N(x,\mathbf{u}) + \langle \lambda, \mathbf{G}\mathbf{u}+\mathbf{E}x+\mathbf{g}\rangle + \sqrt{p}\epsilon_c\|\lambda^*_{\epsilon_c}(x)\| \\ &\leq V^*(x) + 2\sqrt{p}\mathcal{R}_d\epsilon_c. \end{aligned} \tag{2.22}$$

We will see further how we can use relations (2.21) and (2.22) to recover the primal suboptimality for the original problem ($\mathbf{P}(x)$) from the suboptimality of the tightened problem ($\mathbf{P}_{\epsilon_c}(x)$). We assume now that $\bar{\mathbf{u}}_{\epsilon_c}(x) = \text{Alg}((\mathbf{P}_{\epsilon_c}(x)),\epsilon)$, i.e., an $\epsilon$-suboptimal solution for problem ($\mathbf{P}_{\epsilon_c}(x)$), where the accuracy $\epsilon$ is chosen such that

$$\epsilon \leq \frac{1}{4}\min_{j=1,\ldots,p}\{-(\mathbf{G}\tilde{\mathbf{u}}+\mathbf{E}x+\mathbf{g})_j\}, \tag{2.23}$$

and the tightening parameter is chosen e.g., as

$$\epsilon_c = 2\epsilon, \tag{2.24}$$

which satisfies (2.7). Thus, from (2.6) we have:

$$\bar{\mathbf{u}}_{\epsilon_c}(x)\in\mathbf{U}, \|[\mathbf{G}\bar{\mathbf{u}}_{\epsilon_c}(x)+\mathbf{E}x+\mathbf{g}+\epsilon_c\mathbf{e}]_+\|\leq\epsilon \text{ and } |V_N(x,\bar{\mathbf{u}}_{\epsilon_c}(x)) - V^*_{\epsilon_c}(x)|\leq\epsilon.$$

Further, using (2.24) we can write:

$$\left\|[\mathbf{G}\bar{\mathbf{u}}_{\epsilon_c}(x)+\mathbf{E}x+\mathbf{g}+\epsilon_c\mathbf{e}]_+\right\| \leq \epsilon = \frac{\epsilon_c}{2} < \epsilon_c,$$

which implies that for all $j=1,\ldots,p$ we have: $[\mathbf{G}_j\bar{\mathbf{u}}_{\epsilon_c}(x)+\mathbf{E}_jx+\mathbf{g}_j+\epsilon_c]_+ < \epsilon_c$. Since $\mathbf{G}_j\bar{\mathbf{u}}_{\epsilon_c}(x)+\mathbf{E}_jx+\mathbf{g}_j+\epsilon_c \leq [\mathbf{G}_j\bar{\mathbf{u}}_{\epsilon_c}(x)+\mathbf{E}_jx+\mathbf{g}_j+\epsilon_c]_+$ we can conclude that $\mathbf{G}\bar{\mathbf{u}}_{\epsilon_c}(x)+\mathbf{E}x+\mathbf{g}<0$ and therefore the feasibility of $\bar{\mathbf{u}}_{\epsilon_c}(x)$ for the original MPC problem ($\mathbf{P}(x)$) is guaranteed.

Further, since $\bar{\mathbf{u}}_{\epsilon_c}(x)$ is feasible for ($\mathbf{P}(x)$), we have on the one hand that $0 \leq V_N(x,\bar{\mathbf{u}}_{\epsilon_c}(x)) - V^*(x)$. On the other hand, using (2.22) we can write:

$$0 \leq V_N(x,\bar{\mathbf{u}}_{\epsilon_c}(x)) - V^*(x) \leq (1+4\sqrt{p}\mathcal{R}_d)\epsilon, \tag{2.25}$$

and thus $\bar{\mathbf{u}}_{\epsilon_c}(x)$ is a feasible approximate solution of the original problem ($\mathbf{P}(x)$).

We are interested now in proving stability of the proposed MPC scheme. For this purpose, we introduce first the following notation for the feasible suboptimal solution $\bar{\mathbf{u}}_{\epsilon_c}(x)$:

$$\bar{\mathbf{u}}_{\epsilon_c}(x) = \left[(\bar{\mathbf{u}}^0_{\epsilon_c}(x))^{\mathrm{T}}\cdots(\bar{\mathbf{u}}^{N-1}_{\epsilon_c}(x))^{\mathrm{T}}\right]^{\mathrm{T}}.$$

Using this notation, the next state in our MPC scheme is then given by:

$$x^+ = Ax + B\bar{\mathbf{u}}^0_{\epsilon_c}(x). \tag{2.26}$$

For the tightened problem with initial state $x^+$, we will also use the notations $\epsilon_{\mathrm{c}}^+$ and $\mathcal{R}_{\mathrm{d}}^+$ for the tightening parameter and the upper bound given in Lemma 2.4.1, respectively. The following result helps us to construct a strictly feasible vector $\tilde{\mathbf{u}}^+$ for the tightened problem $(\mathbf{P}_{\epsilon_{\mathrm{c}}}(x))$.

**Lemma 2.4.2.** *Let $x^+$ be computed according to* (2.26) *and $\bar{\mathbf{u}}_{\epsilon_c}(x)$ be an $\epsilon$-solution of problem $(\mathbf{P}_{\epsilon_{\mathrm{c}}}(x))$. Then, a strictly feasible vector of problem $(\boldsymbol{P}(x^+))$ is given by:*

$$\tilde{\mathbf{u}}^+ = \left[(\bar{\mathbf{u}}_{\epsilon_c}^1(x))^T \cdots (\bar{\mathbf{u}}_{\epsilon_c}^{N-1}(x))^T (Kx(N))^T\right]^T. \tag{2.27}$$

*Proof.* First, let us note that $\bar{\mathbf{u}}_{\epsilon_{\mathrm{c}}}(x) \in \mathbf{U}$, which together with (2.3) leads to $\tilde{\mathbf{u}}^+ \in \mathbf{U}$. Further, let us recall that the first $N-1$ block inequalities in $\mathbf{G}\mathbf{u} + \mathbf{E}x + \mathbf{g} \leq 0$ are obtained from the state constraints $x(t) \in X$ for $t = 1, \ldots, N-1$ while the last block is deduced from $x(N) \in X^{\mathrm{f}}$. It is straightforward to observe that the first $N-1$ block inequalities in $\mathbf{G}\tilde{\mathbf{u}}^+ + \mathbf{E}x^+ + \mathbf{g}$ are strictly satisfied, since the first $N-2$ blocks coincide with blocks $\{2, \ldots, N-1\}$ of $\mathbf{G}\bar{\mathbf{u}}_{\epsilon_{\mathrm{c}}}(x) + \mathbf{E}x + \mathbf{g}$, while the new block $N-1$ is deduced from $x(N) \in X$ instead of the original $x(N) \in X^{\mathrm{f}}$, which is strictly satisfied due to $X^{\mathrm{f}} \subseteq X$. Also from the $\mu$-contractive property of the set $X^{\mathrm{f}}$ (see (2.3)) we can also deduce that the last block inequality, obtained from $x(N+1) \in X^{\mathrm{f}}$ is also strictly satisfied. Thus, we can conclude that $\tilde{\mathbf{u}}^+$ is a strictly feasible vector of problem $(\mathbf{P}(x^+))$. □

Therefore, in the MPC problem for the next state $x^+$ we update the strictly feasible vector as explained above, and consequently, if $\epsilon_{\mathrm{c}}^+$ satisfies (2.7), according to Remark 2.2.1 we have that $\tilde{\mathbf{u}}^+$ is also strictly feasible for the tightened problem $(\mathbf{P}_{\epsilon_{\mathrm{c}}^+}(x^+))$. It is well-known in the MPC framework (e.g., [118; 75; 103]) that if Assumption 2.2.1 is satisfied and $K$ and $P$ are computed according to Section 2.2, then the following relation holds:

$$V_N(x^+, \tilde{\mathbf{u}}^+) \leq V_N(x, \bar{\mathbf{u}}_{\epsilon_{\mathrm{c}}}(x)) - \|x\|_Q^2 \quad \forall x \in X_N. \tag{2.28}$$

In order to prove the asymptotic stability of the MPC scheme for all $x \in X_N$ we use similar arguments as in [118; 75] by showing that $V_N(x, \bar{\mathbf{u}}_{\epsilon_{\mathrm{c}}}(x))$ is a Lyapunov function for the closed-loop system:

$$\begin{aligned} V_N(x^+, \bar{\mathbf{u}}_{\epsilon_{\mathrm{c}}^+}(x^+)) &\overset{(2.25)}{\leq} V^*(x^+) + (1 + 4\sqrt{p}\mathcal{R}_{\mathrm{d}}^+)\epsilon^+ \\ &\leq V_{\epsilon_{\mathrm{c}}}^*(x^+) + (1 + 4\sqrt{p}\mathcal{R}_{\mathrm{d}}^+)\epsilon^+ \\ &\leq V_N(x^+, \tilde{\mathbf{u}}^+) + (1 + 4\sqrt{p}\mathcal{R}_{\mathrm{d}}^+)\epsilon^+ \\ &\overset{(2.28)}{\leq} V_N(x, \bar{\mathbf{u}}_{\epsilon_{\mathrm{c}}}(x)) - \|x\|_Q^2 + (1 + 4\sqrt{p}\mathcal{R}_{\mathrm{d}}^+)\epsilon^+. \end{aligned}$$

From (2.23) and previous discussion we have that by choosing e.g.:

$$\epsilon^+ \leq \min\left\{ \frac{1}{2(1 + 4\sqrt{p}\mathcal{R}_{\mathrm{d}}^+)} \|x\|_Q^2,\ \frac{1}{4} \min_{j=1,\ldots,p} \{ -\left(\mathbf{G}\tilde{\mathbf{u}}^+ + \mathbf{E}x^+ + \mathbf{g}\right)_j \} \right\}, \tag{2.29}$$

**Algorithm 2.2** MPC scheme with adaptive constraint tightening

1: Given $A, B, Q, R, X, U, X^{\mathrm{f}}, N$
2: Compute offline $K, P, \mathbf{Q}, \mathbf{W}, \mathbf{w}, \mathbf{G}, \mathbf{E}, \mathbf{g}$
3: Measure initial state $x$ at time $t = 0$.
4: Compute initial strictly feasible vector $\tilde{\mathbf{u}}$ for ($\mathbf{P}(x)$).
5: Compute accuracy $\epsilon$ according to (2.23).
6: Compute tightening parameter $\epsilon_{\mathrm{c}}$ from (2.24).
7: **for** $t = 0$ to $\infty$ **do**
8: Measure current state $x$.
9: **if** $x \in X^{\mathrm{f}}$ **then**
10: Compute $u = Kx$.
11: Implement control input $u$.
12: **else**
13: Compute $\epsilon$-solution $\bar{\mathbf{u}}_{\epsilon_{\mathrm{c}}}(x) = \mathrm{Alg}((\mathbf{P}_{\epsilon_{\mathrm{c}}}(x)), \epsilon)$.
14: Compute $u = \bar{\mathbf{u}}^0_{\epsilon_{\mathrm{c}}}(x)$.
15: Update strictly feasible vector $\tilde{\mathbf{u}} \leftarrow \tilde{\mathbf{u}}^+$ according to (2.27).
16: Update accuracy $\epsilon \leftarrow \epsilon^+$ according to (2.29).
17: Update tightening parameter $\epsilon_{\mathrm{c}}$ using (2.24).
18: Implement control input $u$.
19: **end if**
20: **end for**

we get asymptotic stability of the closed-loop system.

We can conclude that choosing the accuracy $\epsilon$ and the tightening parameter $\epsilon_{\mathrm{c}}$ according to (2.29) and (2.24), respectively, the proposed MPC scheme generates a sequence of inputs that ensures feasibility, suboptimality, and stability. Also, we can observe from (2.29) and (2.24) that both $\epsilon$ and $\epsilon_{\mathrm{c}}$ are chosen adaptively, i.e., depending on the initial state of each step of the MPC scheme. More specifically, if for instance the norm of the initial state is big enough, i.e., the system is far from the origin, then the accuracy required for an approximate solution can be less stringent. Thus, our approach can be less restrictive than the approach in [112; 32] where the accuracy is fixed for all initial states. Conversely, if we are sufficiently close to the origin, i.e., $x \in X^{\mathrm{f}}$, we do not have to apply the algorithm for finding the $\epsilon$-solution since the optimal solution is given by $\mathbf{u}^{\mathrm{f}}(x)$.

In Algorithm 2.2 we present an algorithmic description for the proposed adaptive constraint-tightening based MPC scheme with feasibility, suboptimality, and stability guarantees.

Note that in Step 4 of the proposed scheme we can compute the initial strictly

feasible vector $\tilde{\mathbf{u}}$ for ($\mathbf{P}(x)$) by solving the following linear program offline:

$$\begin{aligned} &\max_{\gamma \geq 0, \mathbf{u} \in \mathbf{U}} \gamma \\ &\text{s.t.: } \mathbf{G}\mathbf{u} + \mathbf{E}x + \mathbf{g} + \gamma \leq 0. \end{aligned} \tag{2.30}$$



## 2.5 Numerical Flight Control Simulation Example

This section presents an application of Algorithm 2.2 to the altitude/airspeed regulation of an F-16 aircraft in the presence of actuator saturation, which is a common problem in flight control. Under normal operating conditions, the longitudinal states and the actuators of the aircraft are subject to the constraints shown in Table 2.1. However, during an agile maneuver, the control surfaces of the aircraft might come to a stall due to the aerodynamic pressure. In this situation, the controller is confronted with reduced actuators bounds to stabilize the plane. Our ultimate goal is to apply the techniques presented in the previous sections to stabilize the aircraft at a predefined altitude and velocity. In the following, we rely on the equations of motion and the basic aerodynamic model of the F-16 aircraft presented in [124] and on a publicly available F-16 simulation environment [113].

### 2.5.1 Model Predictive Control Problem Formulation

Our implementation focuses on the stabilization of the longitudinal motion of the aircraft in the presence of actuator constraint saturation. The prediction model of the MPC considers only the longitudinal states $x_{\text{long}}$ of the aircraft, namely altitude $(h)$, pitch angle $(\vartheta)$, velocity $(V)$, angle of attack $(\alpha)$, and pitch rate $(q)$. The original six-degree-of-freedom F-16 model presented in [124] has four first-order actuator models for thrust, elevator, aileron, and rudder. Since the F-16 longitudinal dynamics are associated with the thrust and elevator control inputs we use the reduced equations of motion, i.e., $\dot{x}_{\text{long}} = (x_{\text{long}}, \delta_{\text{th}}, \delta_e)$.

In order to stabilize the aircraft in the presence of actuator constraints, we consider the following optimization problem, which follows from the formulation given in [69]:

$$\begin{aligned} V^*(x) = &\min_{x, \Delta u} \sum_{i=0}^{N-1} \left( \|x(i)\|^2_{Q_{h,V}} + \|\Delta u(i)\|^2_R \right) + V^{\text{f}}(x(N)) \\ \text{s.t.: } &x(i+1) = Ax(i) + B\Delta u(i), \;\; x(0) = x \\ &x(i) \in X \subseteq \mathbb{R}^9, \;\; \Delta u(i) \in U \subseteq \mathbb{R}^2, \;\; \text{for } i = 0, \ldots, N-1 \\ &x(N) \in X^{\text{f}} \subseteq \mathbb{R}^9, \end{aligned} \tag{2.31}$$

where

- $N$ is the prediction and the control horizon;

| | Lower Bound | Upper Bound |
|---|---|---|
| Altitude [ft] | 5,000 | 40,000 |
| Velocity [ft/sec] | 300 | 900 |
| Thrust [lbs] | 1,000 | 19,000 |
| Elevator [deg] | -25 | 25 |
| Thrust rate [lbs/sec] | -10,000 | 10,000 |
| Elevator rate [deg/sec] | -60 | 60 |

**Table 2.1:** Constraints on states and actuators of the F-16 aircraft.

- $x(i)$ is the $i$-step ahead prediction of the state of the augmented system (2.34) that comprises the longitudinal states $x_{\text{long}}$, the aircraft actuators dynamics $u_{\text{in}}$, and the past MPC command signal;
- $\Delta u(t)$ is the control increment sequence;
- $Q_{h,V}$ and $R$ are suitable weighting matrices. In particular, $Q_{h,V}$ penalizes only the altitude $h$ and the velocity $V$, which are the only bounded longitudinal states in this setup, as also shown in Table 2.1;
- $V^{\text{f}}(x(N)) = \frac{1}{2}\left(x(N)\right)^{\text{T}} P x(N)$ is the associated to Problem 2.31;
- $A$ and $B$ are matrices that describe the dynamics of the system;
- $X$ is the admissible region where the constraints on the state of the augmented system are satisfied;
- $U$ is the admissible region where the constraints on the control increment are satisfied;
- $X^{\text{f}}$ is the terminal set.

Solving the optimization problem above requires an appropriate prediction model. We obtained a linearized continuous-time model using the aircraft simulator in [113] and trimming the aircraft at some specified $h = h_{\text{trim}}$ and $V = V_{\text{trim}}$ values. The following linearized discrete-time longitudinal model is then obtained using a sampling period $T_s$:

$$\begin{aligned} x_{\text{long}}(t+1) &= A_{\text{trim}}\, x_{\text{long}}(t) \;+\; B_{\text{trim}}\, u_{\text{in}}(t) \\ u_{\text{in}}(t+1) &= A_u u_{\text{in}}(t) \;+\; B_u u(t), \end{aligned} \tag{2.32}$$

where $u_{\text{in}}(t) = [\delta_{\text{th}}\ \delta_e]^{\text{T}}$, $A_{\text{trim}}$, $A_u$, $B_{\text{trim}}$, and $B_u$ are matrices of proper dimensions. Furthermore, we use a linear quadratic controller $K_e$ to stabilize the inner-loop (IL)

dynamics. Thus, the resulting IL dynamics is given by:

$$\overbrace{\begin{bmatrix} x_{\text{long}}(t+1) \\ u_{\text{in}}(t+1) \end{bmatrix}}^{x_{\text{IL}}(t+1)} = \overbrace{\begin{bmatrix} A_{\text{trim}} & B_{\text{trim}} \\ \begin{bmatrix} 0 \\ B_{\delta_e} K_e \end{bmatrix} & A_u \end{bmatrix}}^{A_{\text{IL}}} \overbrace{\begin{bmatrix} x_{\text{long}}(t) \\ u_{\text{in}}(t) \end{bmatrix}}^{x_{\text{IL}}(t)} + \overbrace{\begin{bmatrix} 0 \\ B_u \end{bmatrix}}^{B_{\text{IL}}} u(t), \tag{2.33}$$

where $B_{=}[B_{\delta_{\text{th}}} \ B_{\delta_e}]^{\text{T}}$. Subsequently, we augment the linearized IL model following the approach presented in [57]. In particular, we add two integrators (one for each actuator) and define new incremental inputs $\Delta u$. The augmented model has the following form:

$$\overbrace{\begin{bmatrix} x_{\text{IL}}(t+1) \\ u(t) \end{bmatrix}}^{x(t+1)} = \overbrace{\begin{bmatrix} A_{\text{IL}} & B_{\text{IL}} \\ 0 & I_m \end{bmatrix}}^{A} \overbrace{\begin{bmatrix} x_{\text{IL}}(t) \\ u(t-1) \end{bmatrix}}^{x(t)} + \overbrace{\begin{bmatrix} B_{\text{IL}} \\ I_m \end{bmatrix}}^{B} \Delta u(t). \tag{2.34}$$

Finally, we compute the prediction model over the prediction horizon $N$ by iteratively performing substitutions starting from the equations (2.34). In detail, for given $x(0) = x$, we define $\mathbf{x} := [x(1)^{\text{T}} \dots x(N)^{\text{T}}]^{\text{T}}$ as the vector described by the following equation:

$$\mathbf{x} = \Psi x + \Theta \boldsymbol{\Delta}\mathbf{u}, \tag{2.35}$$

where $\boldsymbol{\Delta}\mathbf{u} := [\Delta u(0)^{\text{T}} \dots \Delta u(N-1)^{\text{T}}]^{\text{T}}$, $\Psi$ and $\Theta$ are defined as follows:

$$\Psi := \begin{bmatrix} A \\ A^2 \\ \vdots \\ \vdots \\ A^N \end{bmatrix}, \ \Theta := \begin{bmatrix} B & 0 & 0 & \dots & 0 \\ AB & B & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \dots & \dots & B \end{bmatrix}. \tag{2.36}$$

Using the predicted state (2.35), we can reformulate the optimization problem (2.31) in a condensed form as a function of the control increment sequence $\boldsymbol{\Delta}\mathbf{u}$ as follows:

$$\begin{aligned} V(x)^* = \min_{\boldsymbol{\Delta}\mathbf{u} \in \mathbf{U}} \ & \underbrace{\frac{1}{2}\boldsymbol{\Delta}\mathbf{u}^{\text{T}}\mathbf{Q}\boldsymbol{\Delta}\mathbf{u} + (\mathbf{W}x)^{\text{T}}\boldsymbol{\Delta}\mathbf{u} + \text{ const}}_{V_N(x,\ \boldsymbol{\Delta}\mathbf{u})} \\ \text{s.t.:} \quad & \mathbf{G}\boldsymbol{\Delta}\mathbf{u} + \mathbf{E}x + \mathbf{g} \leq 0, \end{aligned} \tag{2.37}$$

where $\mathbf{U} = \overbrace{U \times \dots \times U}^{N \text{ times}}$, $\mathbf{Q} := 2\Theta^{\text{T}}\bar{Q}\Theta + R$, $\mathbf{W} := 2\Theta^{\text{T}}\bar{Q}\Psi$, const $:= x^{\text{T}}\Psi^{\text{T}}Q\Psi x$, $p$ denotes the number of constraints, and $\bar{Q}$ is defined as $\bar{Q} := \text{diag}(\underbrace{Q, \ \dots Q,}_{(N-1) \text{ times}} P)$.

Notice that the constraints on the predictor state in (2.37) are written as a function of $\boldsymbol{\Delta}\mathbf{u}$. Furthermore, the optimization problem (2.37) is a quadratic programming (QP) problem – similarly to ($\mathbf{P}(x)$) – with decision variable $\boldsymbol{\Delta}\mathbf{u}$.

In order to solve the above QP with Algorithm 2.2, we need to introduce the tightened problem associated with (2.37), namely:

$$V_{\epsilon_c}(x)^* = \min_{\mathbf{\Delta u} \in \mathbf{U}} \ \frac{1}{2}\mathbf{\Delta u}^{\mathrm{T}}\mathbf{Q}\mathbf{\Delta u} + (\mathbf{W}x)^{\mathrm{T}}\mathbf{\Delta u} + \text{ const}$$
$$\text{s.t.:} \quad \mathbf{G}\mathbf{\Delta u} + \mathbf{E}x + \mathbf{g} + \epsilon_c \mathbf{e} \leq 0_p. \tag{2.38}$$

### 2.5.2 Simulation Results

Based on the problem formulation introduced in the previous section, we considered the following simulation scenario:

- Trim conditions $[h_{\text{trim}}\ V_{\text{trim}}]^{\mathrm{T}} = [10,000 \text{ ft}\ \ 579,12 \text{ ft/sec}]^{\mathrm{T}}$;
- Sample time $T_s$ equal to $0.05$ sec;
- Prediction horizon $N$ equal to $8$ samples (i.e., $0.40$ sec);
- Initial velocity (with respect to the trim condition) $V_0 = -4.5$ ft/sec;
- Initial altitude (with respect to the trim condition) $h_0 = 82$ ft;
- Reduced actuators bounds, i.e.,

$$[-4000 \ \ -1]^{\mathrm{T}} \leq u_{\text{in}}(t) \leq [14000 \ \ 1.5]^{\mathrm{T}}$$

- Constraints on the optimization variable, i.e., the change of control input

$$[-50 \ \ -3]^{\mathrm{T}} \leq \Delta u(t) \leq [50 \ \ 3]^{\mathrm{T}}.$$

Our problem setup requires computing a control-invariant terminal set $X^{\mathrm{f}}$. For this purpose, we relied on the Multi-Parametric Toolbox (MPT) v3.0 [47] for MATLAB. Our computations resulted in an overall number of $p = 481$ constraints.

We initialize the algorithm as follows. First, we compute a strictly feasible vector by solving the linear program (2.30) for $\mathbf{\Delta \tilde{u}}$ offline. Next, we select the initial accuracy of Algorithm 2.1 by choosing $\epsilon$ according to (2.23), where $\tilde{\mathbf{u}} = \mathbf{\Delta \tilde{u}}$. Finally, we consider an initial number of iterations $\bar{k} = \left\lfloor 2\sqrt{\frac{4\mathcal{R}_{\mathrm{d}}L_{\mathrm{d}}}{\epsilon}} \right\rfloor = 67812$, where $L_{\mathrm{d}} = 703.15$ and $\mathcal{R}_{\mathrm{d}} = 18745$ was obtained by (2.19).

Figure 2.1 and Figure 2.2 illustrate the typical behavior of Algorithm 2.1 presented in Section 2.3 for one selected problem instance. In particular, Figure 2.1 depicts the evolution of the cost functions $V(x)$ (solid grey line) and $V_{\epsilon_c}(x)$ (solid black line) associated to problems (2.37) and (2.38), respectively. In addition, the figure shows the associated optimal cost functions $V^*(x)$ (dashed grey line) and $V^*_{\epsilon_c}(x)$ (dashed black line), respectively. Figure 2.2 presents the maximum value of the estimated gradient $\max_i \ [\nabla d(\lambda)]_i \ \ (i = 1, \ldots, p)$ associated to Problem (2.37) (solid grey line) and Problem (2.38) (solid black line), respectively. Furthermore,

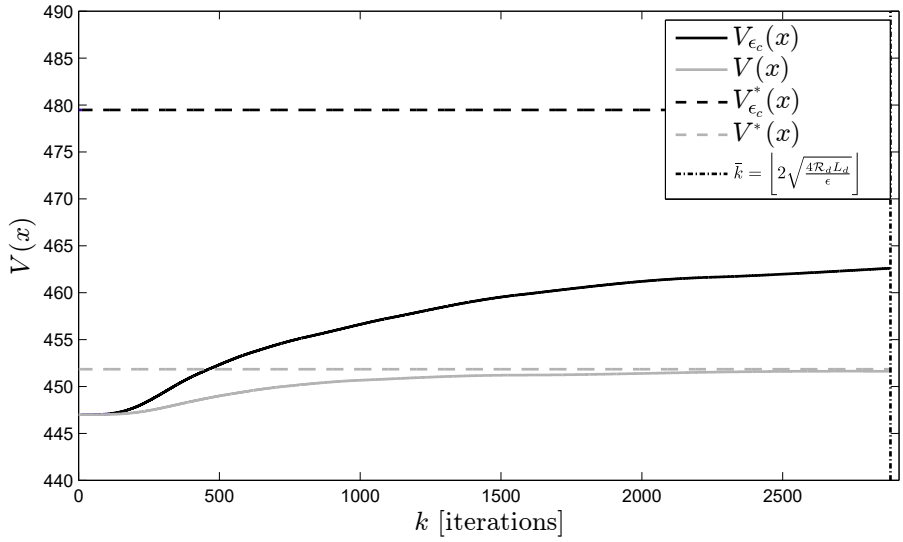


**Figure 2.1:** Comparison between the evolution of the cost function value using Algorithm 2.1 with tightening (solid black lines) and without tightening (solid grey lines) of the constraints for one illustrative problem instance using accuracy $\epsilon = 0.0324$.

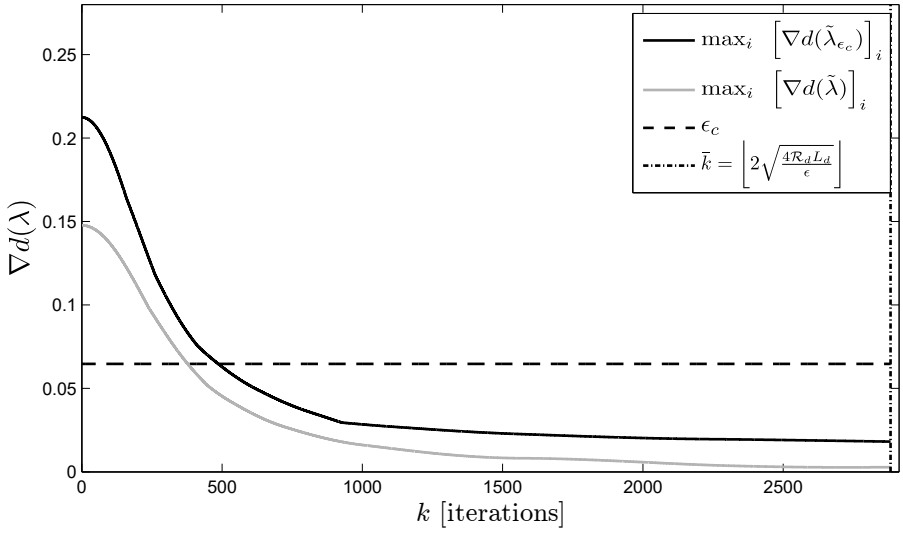


**Figure 2.2:** Comparison between the evolution of the peak gradient value $\max_i \ [\nabla d(\lambda)]_i$ in $(2.13)$ using Algorithm 2.1 with tightening (solid black lines) and without tightening (solid grey lines) of the constraints for one illustrative problem instance using accuracy $\epsilon = 0.0324$. The dashed black line highlights the value of tightening parameter $\epsilon_c$.

the figure highlights the tightening parameter $\epsilon_c$ (dashed black line). As Figure 2 shows, within the predicted number of iterations, the residuals of Problem (2.37) converge to zero asymptotically, while the residuals of Problem (2.38) converge below the bound $\epsilon_c$, resulting in a feasible solution for the the original problem. Note that Algorithm 2.1 already returns a feasible solution for the tightened problem after approximately 500 iterations, while the predicted number of iterations is larger. In addition, note that the solution iterates with constraint tightening are suboptimal for the original Problem (2.37). Moreover, notice that Algorithm 2.1 explores infeasible iterates before converging to the final solution due to its dual approach, as both Figure 2.1 and Figure 2.2 demonstrate.

We compare the performance of our algorithm using adaptive tightening to the one proposed in [112], which uses a fixed tightening parameter $\bar{\epsilon}_c$ for both the state and the input constraints. In order to have a fair comparison, we chose the fixed tightening parameter $\bar{\epsilon}_c$ in a way to obtain a terminal set and a terminal cost that are of similar size and of the same order of magnitude of our problem. From our calculations, the resulting value of the fixed tightening parameter is $\bar{\epsilon}_c = 2 \cdot 10^{-5}$. Furthermore, we use Algorithm 2.1 to compute the approximate solution for the resulting tightened problem. To illustrate the performance of the two approaches, we performed closed-loop simulations (using the Algorithm 2.1 to solve Step 13) where the MPC solutions were implemented on the discrete-time linearized F-16 model. In our experiments, we considered a maneuver that results in the elevator $\delta_e$ – a state of the augmented system (2.34) – hitting a saturation limit. Figure 2.3 depicts the behavior of $\delta_e$ obtained using the proposed adaptive constraint-tightening approach (solid black line), the behavior of $\delta_e$ obtained using the fixed constraint-tightening approach (dashed grey line), and the behavior of $\delta_e$ calculated by solving the original problem using MATLAB's `quadprog` (solid red line). Furthermore, Figure 2.3 highlights the setpoint (dashed black line), the elevator's limits (dotted red lines), and the instant when the state enters the terminal set (dash-dotted vertical black line). Figure 2.4 compares the mismatch on the evolution of $\delta_e$ obtained using the adaptive-tightening approach (solid black line) to the one obtained using the fixed-tightening approach (dashed grey line). Note that the adaptive-tightening difference to the original problem constraints is smaller, especially when $\delta_e$ approaches the limits. The resulting closed-loop altitude and velocity trajectories are depicted in Figure 2.5. In particular, Figure 2.5 highlights in red the trajectories computed using the exact solution. Notice that the closed-loop behaviors obtained using the inexact solutions (the adaptive and the fixed tightening approaches) are comparable to ones obtained using the exact solution.

*Remark* 2.5.1. As Figure 2.4 highlights, the mismatch between the approximate solutions and the exact solution is small. This result is a directly related to the simulation scenario that we are considering. This scenario offers many challenges. In this respect, to keep the number of constraints small, the algorithms use a relatively short horizon to enter the terminal set around the reference altitude and velocity.

Furthermore, the reduced bounds on the elevator input (due to the fault scenario) and the selected initial conditions leave little degree of freedom on the choice of an admissible, strictly feasible input sequence. As a direct consequence, the suboptimality level and the amount of constraint tightening selected by the algorithm are only at infinitesimal levels. Nevertheless, Figure 2.4 well illustrates the qualitative benefits that can derive from the use of our adaptive tightening approach in a real-life application. Potential remedies to this observed behavior are the following. Reference tracking MPC (with target calculator) can be exploited to alleviate the issues related to the short control horizon and terminal constraint feasibility. Furthermore, considering less challenging scenarios (e.g., increased feasible domain, no active constraints before entering the terminal set) allows larger levels of suboptimality.

Figure 2.6 presents the evolution of the normalized level of suboptimality, i.e., $(V_{\epsilon_c}(x) - V^*(x))/V^*(x)$, computed using the proposed adaptive-tightening approach. The figure shows that despite the constraint tightening the cost stays very close to the optimal value during the simulation. The cost value is very similar using the fixed-tightening approach. Nevertheless, a direct and illustrative comparison of the costs is difficult due to the small numerical differences associated to the terminal cost. Figure 2.7 shows the evolution of the tightening parameter $\epsilon_c$, whose value is adapted over time, and remains small. In particular, notice that the adaptive-tightening parameter becomes smaller than the fixed-tightening parameter when the elevator approaches is saturation limits. Note that once the state enters the terminal set, i.e., $x \in X^{\mathrm{f}}$, the algorithm automatically switches to the LQ terminal controller $K$, which occurs at $0.65$ sec into the simulation, therefore the time axis of Figure 2.6 and Figure 2.7 is only displayed until this time. The solution computed with adaptive-tightening approach is closer to the exact solution when the state of the system approaches its limits, as the presented results highlight.

## 2.6 Conclusions

In this chapter we have proposed a model predictive control scheme for discrete-time linear time-invariant systems based on the general framework of inexact numerical optimization algorithms. We have developed our main results on stability and feasibility of a suboptimal MPC scheme using the framework from [75], and computed suboptimal control inputs using fast dual gradient algorithms. We have derived a control strategy that has the following properties: the constraints on the states and inputs are satisfied, asymptotic stability of the closed-loop system is guaranteed and the number of iterations needed for a certain level of suboptimality can be determined. We have implemented our proposed approach in a flight control scenario in order to illustrate its behavior, and showed that the performance degradation due to the constraint tightening remains very small for this problem. Furthermore, we have compared the presented approach with alternative techniques from [112].

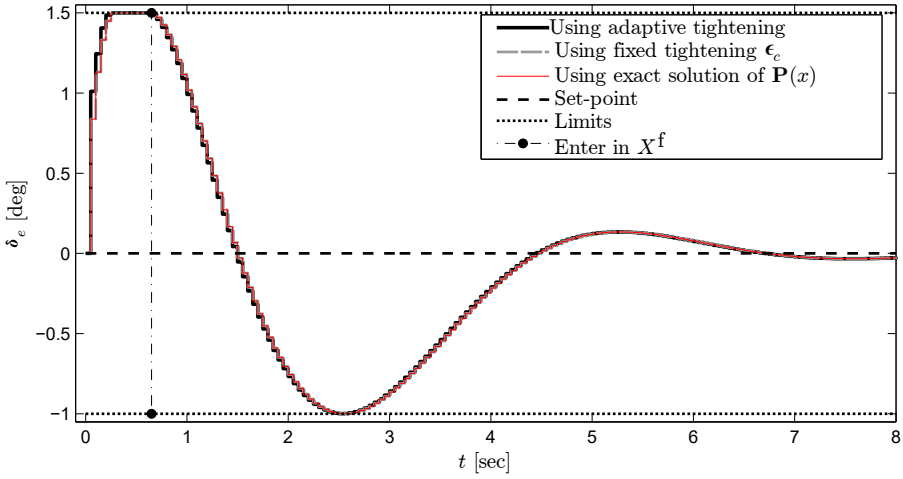


**Figure 2.3:** Elevator control input comparison between the adaptive constraint-tightening Algorithm 2.2 (solid black line) and the fixed constraint tightening (dashed grey line) in [112].

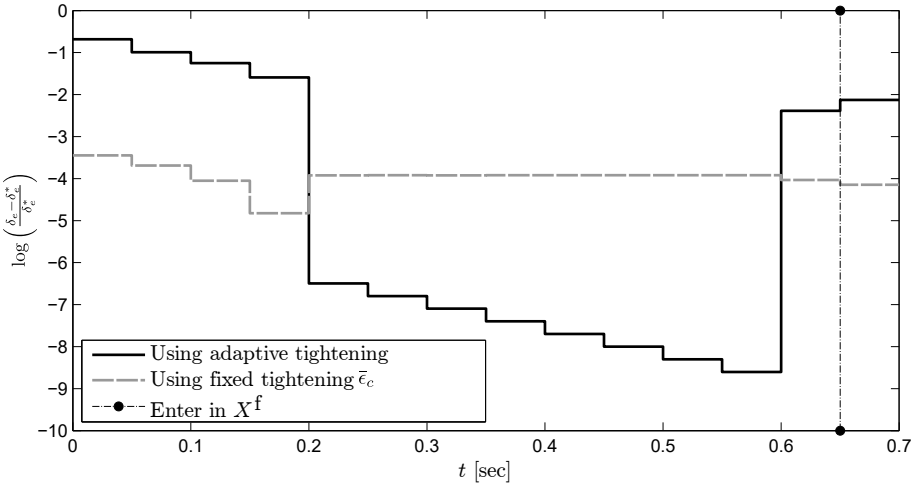


**Figure 2.4:** Elevator control error comparison (with respect to the exact solution) between the adaptive constraint-tightening Algorithm 2.2 (solid black line) and the fixed constraint tightening (dashed grey line) in [112].

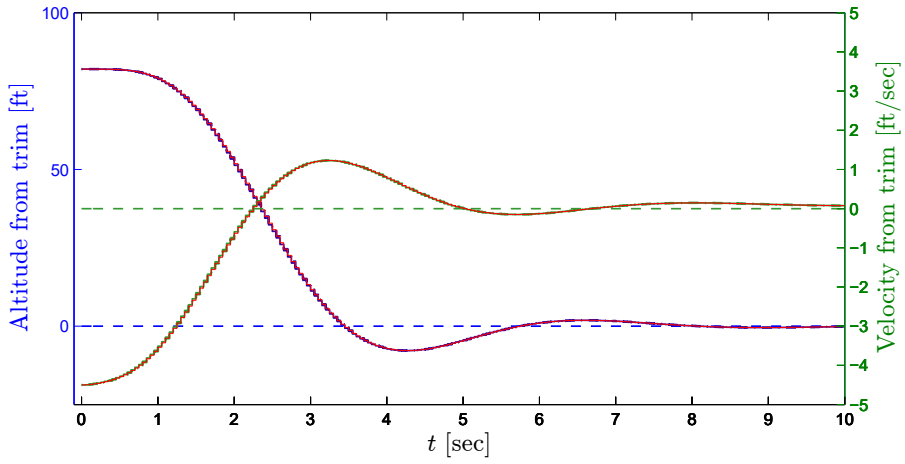


**Figure 2.5:** Closed-loop altitude and velocity trajectories obtained using the adaptive constraint-tightening Algorithm 2.2 (solid blue and green lines) and the fixed-constraint tightening in [112] (dot-dashed blue and green lines). The solid red lines highlight the output trajectories obtained using the exact solutions and the dashed lines highlight the setpoint.

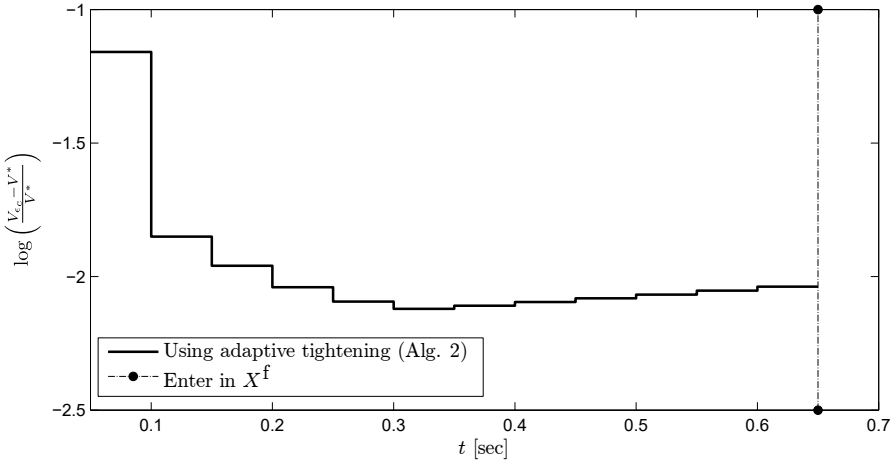


**Figure 2.6:** Evolution of the normalized level of suboptimality defined as $(V_{\epsilon_c}(x) - V^*(x))/V^*(x)$.

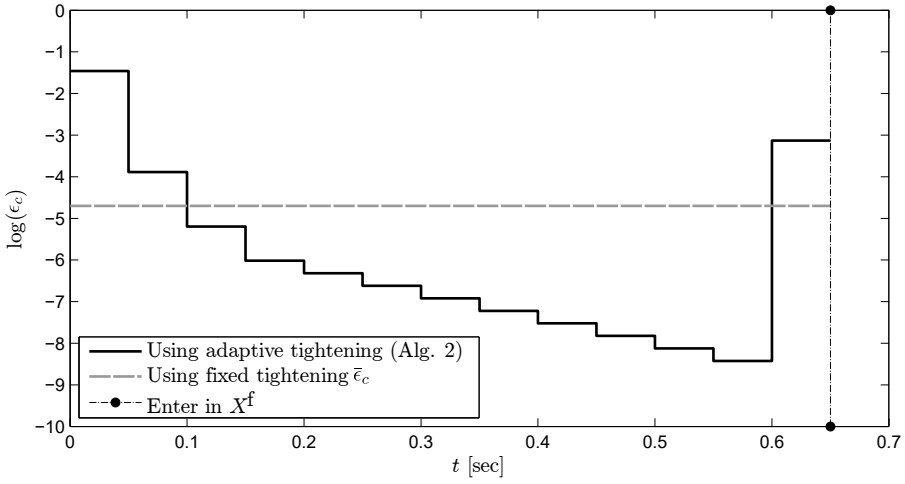


**Figure 2.7:** Evolution of the tightening parameter $\epsilon_c$.

Chapter 2

# 3

# Operator-Splitting and Gradient Methods for Real-Time Predictive Flight Control Design




## Abstract

Despite their ability to operate on the limits of performance, handle multivariable and nonlinear systems, and offer online adaptation and reconfiguration capabilities, model predictive control (MPC) approaches to aerospace applications suffer from limitations related to the online computational burden and complexity of the underlying optimization problem. In this chapter we focus on quadratic programming (QP) formulations that represent certain types of predictive flight control problems, and propose a parallelizable QP solver based on operator-splitting and fast gradient methods. The presented methodology and solution approach promise real-time implementation of QP-based predictive flight control schemes on future embedded platforms. We also provide formal analysis and guidelines on how to reshape the feasible region of the MPC problem st each time instant to ensure recursive feasibility and closed-loop stability. Finally, simulation results for the longitudinal control of an Airbus passenger aircraft are presented to show how the obtained computable certificates can be simplified in practice and to bring the proposed approach a step closer to future on-board real-time implementation.

## 3.1 Introduction

Model predictive control (MPC) is a consolidated control technique that has been studied since the mid '60s [95]. Initially, its industrial application has been limited mainly to systems with slow dynamics [104; 29]. In recent years, however, this technique has gained increasing attention also for the control of systems with fast dynamics in areas such as aerospace and automotive applications [57; 42; 50; 43; 38]. MPC is very appealing for its ability to easily formulate the control objectives and handle the constraints on the plant via a finite-horizon optimal control problem formulation. In particular, every time new measurements are available, a model predictive controller solves an optimization problem (very often a quadratic programming problem) that includes the control objectives (formulated in the cost function) and the information concerning the constraints and the dynamics of the plant. Then, as soon as the optimizer returns a solution for the MPC problem, i.e., a control sequence based on the predicted evolution of the system (over a predefined time window called prediction horizon) satisfying the constraints and the control objectives, the MPC implements only the first element of the returned solution as the closed-loop control law, in a receding horizon fashion [72].

This chapter focuses on one major limitation that is preventing a more extensive use of MPC in various fields with hard real-time constraints, such as aerospace, i.e., the *online optimization*. In particular, solving the optimization problem associated with the MPC online, i.e., every time new measurements are available, might lead to issues such as *(i)* unpredictable computation times, *(ii)* increasing hardware requirements (in terms of CPUs, memory, FPUs, etc.), *(iii)* infeasible control commands, and *(iv)* unstable closed-loop behavior.

Solutions have been proposed to avoid the online optimization, such as [4], in which equivalent explicit control laws are computed offline and stored in a look-up table. The control law is then computed online by searching on the look-up table, considerably reducing the online computation burden. The main drawback of this approach is that it can only handle small/medium scale problems due to the complexity and related memory requirements of the look-up tables. In recent years, the solvers used for online optimization have received increased attention to overcome the limitation of explicit MPC controllers [4] related to the problem size. Two main families of solvers have received particular attention in the MPC community: second-order and first-order solvers. The first category includes interior-points methods [101; 134] and active-set methods [28], for example. Several algorithms have been proposed that focus on the real-time computation issue, such as [131; 141; 52]. The second category includes gradient methods [34; 82; 33; 3; 61; 86; 90] and operator splitting methods, such as alternating direction methods of multipliers (ADMM) or alternating minimization algorithms (AMA) [8; 5; 128; 63; 35; 96]. This second category of solvers is the focus of our work in this chapter for the following reasons. These solvers are more suitable for embedded platforms (e.g., [53; 92]), since they only rely on very simple mathematical functions and require little memory (hence

simpler hardware and software) compared to the second-order solvers. Furthermore, detailed convergence analysis (e.g., [82; 46; 35]) and worst-case computation time to reach a desired level of suboptimality are available (e.g., [82; 35; 90]).

In order to overcome the issues *(i)-(iv)* listed above, we propose a novel optimization algorithm: a parallel dual fast-gradient (PDFG) solver [20], based on operator-splitting techniques [121] and projected dual fast-gradient methods [82] to solve the MPC optimization problem online. Thanks to the use of operator-splitting methods, the algorithm solves smaller independent subproblems in parallel, whose sizes do not increase with the length of the horizon and are characterized by a better condition number compared to the original condensed QP problem. The PDFG relies on the projected dual fast-gradient method [82] and can be used to solve MPC problems with state, input, and output constraints. The main steps of the algorithm involve the computation of the solution of an unconstrained least-squares problem (in the primal variables), a consensus step, and a linear update of the dual variables. In particular, our solver operates on the *dual* of the subproblems to compute a primal feasible and suboptimal solution, given that the complexity of the primal feasible region prevents an efficient direct computation of the projection step of the gradient method in the primal subproblems. While solving the projection step is very efficient using the dual formulation, early termination after a fixed number of iterations of the solver (driven by real-time constraints) might lead to *primal infeasible* solutions (given that the fast-gradient method converges to the solution of the QP problem only asymptotically). Nevertheless, an upper-bound on the primal infeasibility is available [78] and can be exploited to formally recover primal feasibility and suboptimality. In this respect, we propose an adaptive strategy to reshape the (primal) feasible region of each independent subproblem in order to guarantee recursive primal feasibility (of the original MPC problem) and closed-loop stability of the system controlled by the proposed MPC controller. Finally, we describe how our theoretical findings can be adapted and simplified in practice to apply our approach to the real-time longitudinal control of an Airbus passenger aircraft [21].

The chapter is structured as follows. Section 3.2 introduces our application, an Airbus passenger aircraft, used as motivating example for this study, while Section 3.3 introduces the PDFG solver for a general convex problem. Section 3.4 details the MPC problem formulation and the steps needed to reformulate the original MPC problem for the PDFG. Then, Section 3.5 describes the strategy to adaptively update the feasible region of the MPC problem in order to formally ensure recursive feasibility and closed-loop stable trajectories, and the strategy to ensure real-time computation in practical MPC applications. Section 3.6 presents the simulation results obtained by using the proposed control algorithm. Section 3.7 concludes the chapter. Finally, the proofs of the Lemmas and Theorems in this chapter can be found in Appendices A-F.

The following notation is used in the remainder of the chapter. For $u \in \mathbb{R}^n$, $\|u\| = \sqrt{\langle u, u \rangle}$ is the Euclidean norm and $[u]_+$ is the projection onto the nonnegative orthant $\mathbb{R}^n_+$. Given a matrix $A$, $[A]_i$ denotes the $i$-th row of A and $[A]_{i,j}$ the

entry $(i, j)$ of A. Furthermore, $\mathbf{1}_n$ is the vector of ones in $\mathbb{R}^n$ and $I_n$ the identity matrix in $\mathbb{R}^{n\times n}$. In addition, $\text{eig}_{\max}(A)$ and $\text{eig}_{\min}(A)$ denote the largest and the smallest (modulus) eigenvalues of the matrix $A^{\mathrm{T}}A$, respectively. $P \in \mathbb{S}^n_{>0}$ denotes that $P \in \mathbb{R}^{n\times n}$ is positive definite.

## 3.2 Motivating Example

In this section we consider control of the longitudinal dynamics of an Airbus passenger aircraft [37], as a motivating example, and compare the behavior of the baseline controller (provided with the aircraft simulator) and a traditional MPC design (refer to [23] for more details on the control architecture). The purpose of this introductory comparison is twofold. On one hand, we show how the on-board Guidance & Control unit can benefit in terms of performance by using an MPC controller. On the other hand, we show the limitations (in terms of computation time) that currently prevent the use of MPC for these tasks.

Our benchmark model is an Airbus civilian aircraft simulator [36; 37] and the control of the longitudinal dynamics of the aircraft is investigated. In this respect, the controller can rely on the tail control surfaces. Furthermore, the control design has to take into account the constraints, not only on the control surfaces, but also on the states and on the outputs of the system. A sampling time of $T_s = 0.04$ sec is available to solve the MPC problem online.

The control goal is to track a desired reference signal generated by a pilot stick command on the vertical load factor $n_z$ that forces the system to reach the upper saturation limit (as depicted in Figure 3.1). The same scenario will be considered in Section 3.6 to show the potential of our proposed solver for this application from the computational point of view.

For now, our aim is to illustrate that MPC has the potential to improve the performance of the aircraft in this scenario, albeit with increased and in general unpredictable computation time due to online optimization. Figure 3.1 shows the tracking performance of the two controllers. In particular, the solid line is associated with the MPC controller, while the dashed line with the baseline controller. The reference trajectory is the dash-dotted line and the upper-bound on the vertical load factor is the dotted line. As Figure 3.1 shows, the MPC controller improves the tracking performance of the baseline controller, mostly because it is able to accurately predict the aircraft behavior and operate against the constraint limits.

The improvement of performance depicted in Figure 3.1 comes at the cost of a large computation time $t_{\text{opt}}$ to solve the constrained MPC optimization problem online, as depicted in Figure 3.2. In particular, Figure 3.2 depicts the computation time required for online optimization $t_{\text{opt}}$ obtained using a second-order solver (dashed line) and the average computation time (dotted line) compared to the sampling time of the system (solid line). Note the behavior of the computation time that is typical of a second-order solver. We show, in Section 3.6, how this behavior can be

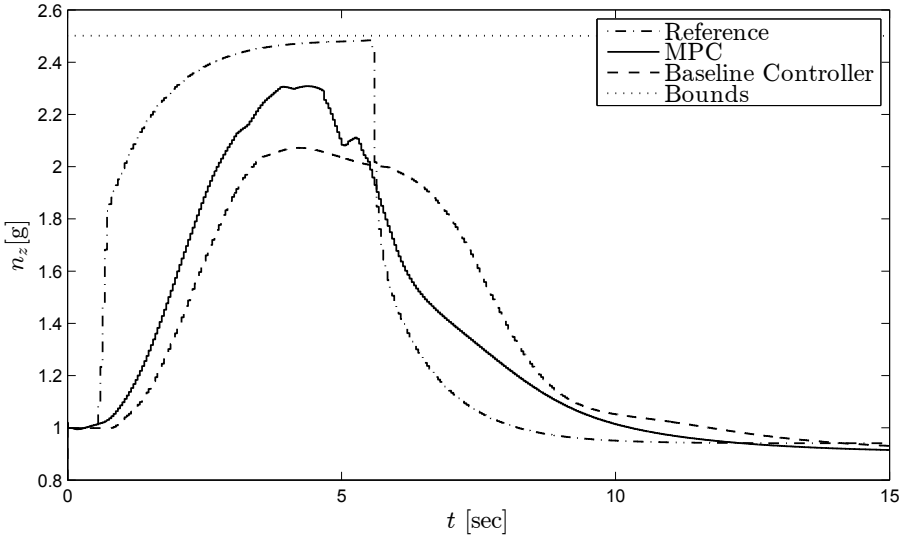


**Figure 3.1:** Vertical load factor.

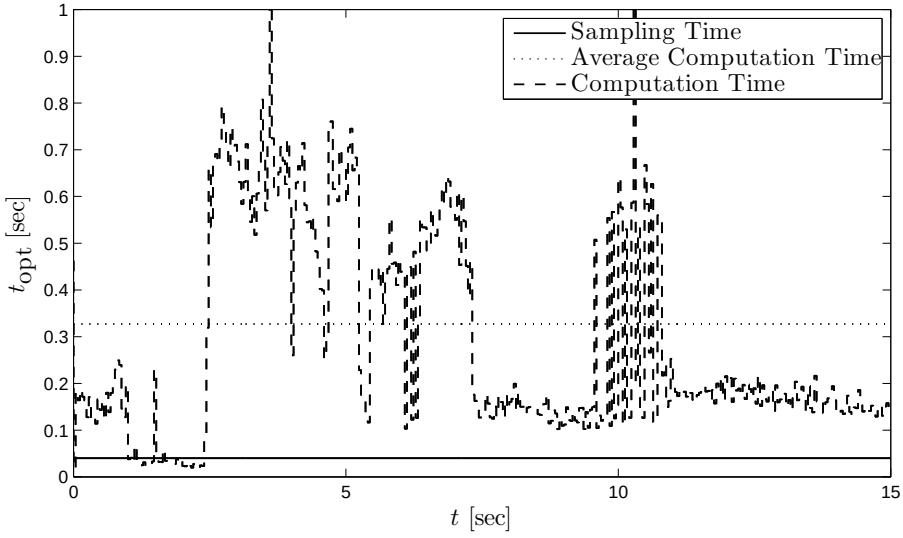


**Figure 3.2:** Computation time.

improved by using our proposed first-order solver.

*Remark* 3.2.1. In the context of this work, we focus on the computational aspects related to MPC. For more examples and details on other design challenges that this application offers in the context of MPC (e.g., how to handle model mismatches, how to perform control reconfiguration when actuator faults occur during the flight, etc.) we refer the reader to [23]. An overview of the most recent techniques for robust

MPC (to handle disturbances and model mismatches) can be found in Chapter 16 of [6].

Motivated by these results, the next section focuses on a QP solver, a parallel dual fast gradient method (PDFG), based on operator-splitting and first-order methods, that can be used to solved the MPC optimization problem online in real-time using only simple mathematical operations.

## 3.3 The Parallel Dual Fast Gradient Algorithm

This section introduces the solver we rely on throughout the developments in the chapter, i.e., the parallel dual fast gradient method (or PDFG) proposed in [20].

Consider the following problem:

$$\underset{\phi_t}{\text{minimize}} \sum_{t=0}^{N} \bar{\mathbf{V}}_t(\phi_t) \tag{3.1a}$$

$$\text{subject to: } H_1\phi_{t+1} - H_2\phi_t = 0 \quad t = 0, \ldots, N-1 \tag{3.1b}$$

$$G\phi_t + g \leq 0 \qquad t = 0, \ldots, N, \tag{3.1c}$$

where the $\phi_t \in \mathbb{R}^{n_\phi}$ ($n_\phi$ indicates the dimension of $\phi_t$) are coupled in the constraints (3.1b), where $H_1 \in \mathbb{R}^{n_z \times n_\phi}$ and $H_2 \in \mathbb{R}^{n_z \times n_\phi}$, and $G \in \mathbb{R}^{c \times n_\phi}$. We consider the following assumptions:

**Assumption 3.3.1.** *The functions $\bar{\mathbf{V}}_t(\phi_t)$ ($t = 0, \ldots, N$) are convex functions.*

**Assumption 3.3.2.** *$H_1$ and $H_2$ have full row rank.*

### 3.3.1 Operator Splitting

Our goal is to simplify the structure of Problem (3.1) and improve the calculation of its solution (in terms of computation time). In this respect, we aim to decompose the problem to allow one to compute each $\phi_t$ independently using $N+1$ independent (parallel) workers[1] $\Pi_t$. Hence, one possible way to exploit the structure of Problem (3.1) is to remove the coupling (3.1b) by introducing $N$ *consensus* variables $z_t \in \mathbb{R}^{n_z}$ ($t = 1, \ldots, N$) as follows (according to the splitting strategy proposed in [121] for MPC):

$$z_t = H_1\phi_t = H_2\phi_{t-1}. \tag{3.2}$$

[1] A worker is a process performing computations. We used the same terminology of the MATLAB's Parallel Computing Toolbox.

This allows Problem (3.1) to be written as the following sum of independent subproblems that can be solved by $N+1$ independent workers $\Pi_t$:

$$\underset{\phi_t}{\text{minimize}} \sum_{t=0}^{N} \bar{\mathbf{V}}_t(\phi_t) \tag{3.3a}$$

$$\text{subject to: } H_z z_{t+1} - \bar{H}_2 \phi_t \leq 0 \quad t = 0, \ldots, N-1 \tag{3.3b}$$

$$H_z z_t - \bar{H}_1 \phi_t \leq 0 \qquad t = 1, \ldots, N \tag{3.3c}$$

$$G_t \phi_t + g_t \leq 0 \qquad t = 0, \ldots, N, \tag{3.3d}$$

where $H_z^{\mathrm{T}} = [I \; -I]$, $\bar{H}_1^{\mathrm{T}} = [H_1^{\mathrm{T}} \; -H_1^{\mathrm{T}}]$, $\bar{H}_2^{\mathrm{T}} = [H_2^{\mathrm{T}} \; -H_2^{\mathrm{T}}]$, $G_t = G$, $g_t = g$ are used to reformulate the equality constraints (3.1b) as inequality constraints.

*Remark* 3.3.1. We reformulate the consensus constraints (3.2) as (3.3b)-(3.3c) for theoretical purposes only. In particular, the Lagrange multipliers associated with the inequality constraints (3.3b)-(3.3c) have a known sign (i.e., they must be greater than or equal to zero) and we can exploit this information to derive upper bounds on the dual variables in the Lemmas and Theorems that follow.

The PDFG is detailed in Algorithm 3.1. The PDFG is a dual solver, i.e., a solver that operates on the dual of Problem (3.3), which is given by:

$$\underset{\mu_t \geq 0}{\text{maximize}} \sum_{t=0}^{N} d_t(\mu_t), \tag{3.4}$$

where $\mu_t \geq 0$ is the vector of Lagrange multipliers associated with the inequality constraints in Problem (3.3). Furthermore, the dual function $d_t(\mu_t)$ is defined as follows:

$$d_t(\mu_t) := \underset{\xi_t}{\text{minimize}} \, \mathcal{L}_t(\mu_t, \xi_t), \tag{3.5}$$

where $\xi_t^{\mathrm{T}} := [\phi_t^{\mathrm{T}} z_t^{\mathrm{T}} z_{t+1}^{\mathrm{T}}]$ and $\mathcal{L}_t(\mu_t, \xi_t)$ is defined as follows:

$$\mathcal{L}_t(\mu_t, \xi_t) := \mathbf{V}_t(\xi_t) + \langle \mu_t, \bar{G}_t \xi_t + \bar{g}_t \rangle, \tag{3.6}$$

given $\rho > 0$, $\mathbf{V}_t(\xi_t) := \bar{\mathbf{V}}_t(\phi_t) + \frac{\rho}{2}\|H_z z_{t+1} - \bar{H}_2 \phi_t\|_2^2 + \frac{\rho}{2}\|H_z z_t - \bar{H}_1 \phi_t\|_2^2$, and $\bar{G}$, $\bar{g}$ follow from the partitioning of the constraints (3.3b)-(3.3d) according to the definition of $\xi_t$. Note that, for $t = 0$ and $t = N$, $\|H_z z_t - \bar{H}_1 \phi_t\|_2^2$ and $\|H_z z_{t+1} - \bar{H}_2 \phi_t\|_2^2$ do not appear in the definition of $\mathcal{L}_0(\mu_0, \xi_0)$ and $\mathcal{L}_N(\mu_N, \xi_N)$, respectively (together with the associated inequality constraints (3.3c) and (3.3b)). Furthermore, if Assumptions 3.3.1 and 3.3.2 hold, the $\mathbf{V}_t$ are strongly convex functions with convexity parameter $\sigma_{\mathbf{V}_t}$.

*Remark* 3.3.2. The size of the subproblems obtained from the decomposition remains unaffected if $N$ increases. Intuitively, this *modularity* is an additional feature that can be exploited to preserve some favorable numerical properties of the problem (e.g., conditioning, Lipschitz constant, etc.) even when the algorithm is running in serialized mode [130].

**Algorithm 3.1** Parallel Dual Fast Gradient Method.

Given $\xi_t^0, \mu_t^0, \bar{G}_t, \bar{g}_t, L_t$, and $\bar{k}$ for each $\Pi_t$ $(t=0,\ldots,N)$
**while** $k \leq \bar{k}$ or termination criteria are not met **do**

1. $\Pi_t$ computes $\phi_t^{k+1} = \underset{\phi_t}{\operatorname{argmin}}\ \mathcal{L}_t(\mu_t, \xi_t)$.
2. $\Pi_t$ receives $\phi_{t-1}^{k+1}$ from $\Pi_{t-1}$, sends $\phi_t^{k+1}$ to $\Pi_{t+1}$.
3. $\Pi_t$ updates $z_t^{k+1}$ according to (3.7).
4. $\Pi_t$ receives $z_{t-1}^{k+1}$ from $\Pi_{t-1}$, sends $z_t^{k+1}$ to $\Pi_{t+1}$.
5. $\Pi_t$ computes
$$\hat{\mu}_t^{k+1} = \left[\mu_t^k + L_t^{-1}\nabla_{\mu_t}^{\mathrm{T}} d_t(\mu_t)\right]_+ .$$
6. Define: $a := \frac{k+1}{k+3} I$, $b_t := L_t^{-1}\frac{2}{(k+3)}$.
7. $\Pi_t$ computes:
$$\mu_t^{k+1} = a\hat{\mu}_t^{k+1} + b_t \left[\sum_{s=0}^{k} \frac{s+1}{2}\nabla_{\mu_t}^{\mathrm{T}} d_t(\mu_t)\right]_+ .$$

**end while**

*Remark* 3.3.3. We introduced a quadratic penalty in the cost of the form $\rho/2(\|\bar{H}_1\phi_t - H_z z_t\|^2 + \|\bar{H}_2\phi_t - H_z z_{t+1}\|^2)$, according to the ADMM strategy [8]. This penalty has no impact on the cost of Problem (3.1), if the consensus constraints (3.2) are satisfied. As an alternative, we could have used the Lagrangian that does not include the quadratic penalty according to the AMA strategy [35]. In this context, we preferred the ADMM alternative to relax the assumption on the cost $\bar{\mathbf{V}}_t$ (Assumption 3.3.1), given that AMA requires these functions to be strongly convex (this assumption might be too restrictive for some MPC applications).

Algorithm 3.1 relies on Nesterov's DFG method in which we propose the inner problem (i.e., the minimization of the augmented Lagrangian (3.5)) to be solved in an ADMM fashion, as explained below. Specifically, as Figure 3.3 depicts, at each iteration of the algorithm, $\Pi_t$ computes a minimizer $\xi_t^k$ for $\mathcal{L}_t(\mu_t, \xi_t)$ (steps ①-④), i.e., the algorithm returns a solution for each inner subproblem (3.5). In particular, our algorithm, in compliance with the ADMM strategy, first minimizes $\mathcal{L}_t(\mu_t, \xi_t)$ with respect to $\phi_t$ in parallel for each subproblem (step ①). Then, using the information received by $\Pi_{t-1}$, that is, the updated value of $\phi_{t-1}^{k+1}$ (synchronization step ②), our algorithm computes—in parallel for each subproblem—the value of the global variable $z_t$ according to the following rule (step ③):

$$z_t^{k+1} = \frac{1}{2}\left(H_1\phi_t^{k+1} + H_2\phi_{t-1}^{k+1}\right). \tag{3.7}$$

Then, (synchronization step ④) $\Pi_t$ receives (sends) the updated value of $z_{t-1}^{k+1}$ ($z_t^{k+1}$) from $\Pi_{t-1}$ (to $\Pi_{t+1}$), respectively. Finally, the worker $\Pi_t$ computes the new values

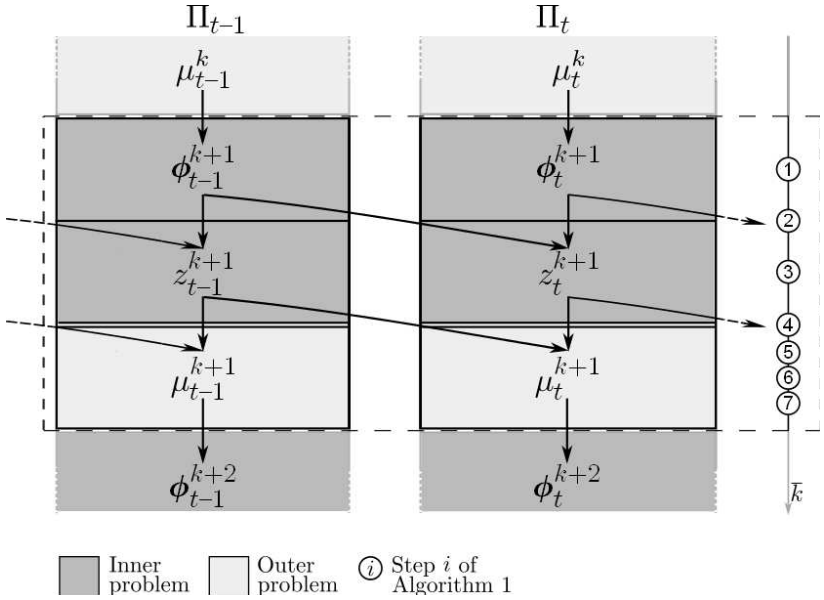


**Figure 3.3:** Iteration $k$ of Algorithm 3.1.

of the multipliers $\mu_t^{k+1}$ (steps ⑤-⑦).

We compute offline (for each subproblem) the Lipschitz constant $L_t$ associated with $\nabla_{\mu_t} d_t(\mu_t)$ to perform the multipliers' update as $L_t := \|\bar{G}_t\|_2^2 \sigma_{\mathbf{V}_t}{}^{-1}$. Furthermore, the solver iterates for $\bar{k}$ iterations or until a given termination criterion is met (e.g., $\|\mu_t^k - \mu_t^{k-1}\|_2^2 \leq \eta_t$, where $\eta_t$ is the desired accuracy).

Each $\Pi_t$ can be assigned to a different processor to parallelize the execution of Algorithm 3.1. The synchronization overhead (steps ② and ④) can have impact on the performance of the solver. Nevertheless, the workers only exchange vectors of small dimension (matrices are not involved in the exchange). Hence, the operations should be performed efficiently without largely affecting the performance of the algorithm.

The proposed algorithm converges to the optimal solution of Problem (3.1) asymptotically, that is, for $k \to \infty$ (Theorem 3.3.1). In practice, we have to take into account that the execution of the solver terminates in finite time. Using an argument similar to the one of Theorem 1 in [78], we can compute the primal feasibility violation and the level of suboptimality of the solution of Problem (3.3) returned by Algorithm 3.1.

**Theorem 3.3.1** ([78])**.** *Suppose that Assumptions 3.3.1 and 3.3.2 hold. Then, let the sequences* $(\boldsymbol{\xi}_t^k, \hat{\mu}_t^k, \mu_t^k)$ *be generated by Algorithm 3.1, and* $\hat{\boldsymbol{\xi}}_t^k := \sum_{s=0}^{k} \frac{2(s+1)}{(k+1)(k+2)} \boldsymbol{\xi}^s$. *Then, an estimate on the primal feasibility violation for the subproblem* (3.3) *is given*

*by the following:*

$$\|[\nabla^T_{\mu_t} d_t(\hat{\boldsymbol{\xi}}^k_t)]_+\| \leq \frac{8R_t \max_{t=0,\dots,N}\{L_t\}}{(k+1)^2} =: \eta_t, \tag{3.8}$$

*where* $R_t := \|\mu^*_t\|$ *. Moreover, an estimate on primal suboptimality is given by the following:*

$$0 \leq \mathbf{V}^*_t - \mathbf{V}_t(\hat{\boldsymbol{\xi}}_t) \leq R_t \eta_t. \tag{3.9}$$

Algorithm 3.1 terminates after a fixed number of iterations that depends on $\eta_t$ and $R_t$ [78]:

$$\bar{k} = \underset{t=0,\dots,N}{\text{maximize}}\, \bar{k}_t := \left\lfloor \sqrt{\frac{8R_t \max_{t=0,\dots,N}\{L_t\}}{\eta_t}} \right\rfloor. \tag{3.10}$$

The upper bound on the number of iterations (3.10) is proportional to the level of suboptimality $\eta_t$. Hence, we can either decide to fix $\eta_t$ to derive $\bar{k}$ or, if the number of iterations is imposed by the available processors/sampling time, fix $\bar{k}$ to compute the achievable guaranteed level of suboptimality $\eta_t$.

According to Theorem 3.3.1 an averaged sequence $\hat{\boldsymbol{\xi}}_t$ should be implemented. In practice, however, the use of $\xi^{\bar{k}}_t$ (i.e., the last element of the primal sequence computed by Algorithm 3.1) has a more positive impact on the convergence of the solver [89].

*Remark* 3.3.4. In [60], ADMM is used in combination with Nesterov's fast gradient methods. At each iteration, the algorithm proposed in [60], first computes the exact minimizer $\phi_t$ by solving a constrained QP problem and then updates the multipliers associated with the consensus constraints (3.3b)-(3.3c). Our PDFG combines an unconstrained least-squares problem and a consensus step to solve the inner problem (3.5), which can be solved more efficiently, compared to the constrained QP in [60], and then updates the multipliers associated with the inequality constraints in Problem (3.3) using the acceleration strategy proposed in [82], accelerating the convergence of the algorithm compared to [60]. The proposed algorithm is also different from the one proposed in [121] to solve similar problem formulations. In particular, the workers exchange the necessary pieces of information before the update of the Lagrange multipliers and none of the dual variables are exchanged between the neighboring workers, as Figure 3.3 highlights, simplifying the study of the convergence of the algorithm. Furthermore, the information exchange between neighboring workers is unidirectional, i.e., $\Pi_{t-1}$ sends the updated information to $\Pi_t$, but $\Pi_t$ does not send any updated information to $\Pi_{t-1}$. Finally, we focuses on the use of first-order solvers to solve the inner problem (3.5), while the focus of [121] was on the novelty of the proposed decomposition technique.

In the following, we show how to reshape the feasible region of Problem (3.1) to anticipate possible constraints violations when using the PDFG. In particular, Section 3.3.2 introduces the equality relaxed (ER) subproblems that rely on a relaxed

consensus policy, while Section 3.3.3 introduces a preliminary set of inequality-tightened (IT) subproblems that rely on a tightened feasible region to compensate for the early termination of the PDFG. Section 3.5 shows how to improve the formulation of the subproblems, from the MPC perspective, to formally ensure recursive feasibility and closed-loop stability.

### 3.3.2 Equality Constraint Relaxation

The decomposition presented in the previous section is possible thanks to the introduction of the consensus constraints (3.2) that are, equivalently, reformulated as inequality constraints (3.3b)-(3.3c) in the definition of the independent subproblems (3.3). The iterative nature of the proposed solver and its asymptotic convergence properties, however, might prevent that the constraints (3.3b)-(3.3c) are actually satisfied at the equality. This observation motivates the introduction of ER parameters $\epsilon_{z_t}$,$\epsilon_{z_{t+1}}>0$ to relax the consensus constrained as follows. For each subproblem (3.3), the former equality constraints (3.2) are replaced by the following inequality constraints:

$$\bar{H}_1\phi_t - H_z z_t \leq \epsilon_{z_t}\,\mathbf{1}_{2n_z}, \tag{3.11a}$$
$$\bar{H}_2\phi_t - H_z z_{t+1} \leq \epsilon_{z_{t+1}}\,\mathbf{1}_{2n_z}\,. \tag{3.11b}$$

Thus, for each subproblem, we can realistically consider a feasible region defined as follows:

$$w_t : [\,\bar{H}_1 \;\; -H_z \;\; 0]\;\xi_t - \epsilon_{z_t}\mathbf{1}_{2n_z} \;\leq 0, \tag{3.12a}$$
$$v_{t+1} : [\,\bar{H}_2 \;\; 0 \;\; -H_z]\;\xi_t - \epsilon_{z_{t+1}}\mathbf{1}_{2n_z} \leq 0, \tag{3.12b}$$
$$\lambda_t : [\,G_t \;\; 0 \;\; 0]\;\xi_t + \;g_t \;\; \leq 0, \tag{3.12c}$$

or, in a more compact notation:

$$G_{\xi_t}\xi_t + g_{\xi_t} \leq 0, \tag{3.13}$$

where $G_{\xi_t} \in \mathbb{R}^{p_{\xi_t}\times(n_\phi+2n_z)}$, $p_{\xi_t} := p+4n_z$, and $w_t$, $v_{t+1}$, $\lambda_t$ are the Lagrange multipliers associated with the constraints (3.12a), (3.12b), and (3.12c), respectively. Hence, let $\mu_t^{\mathrm{T}} := [\lambda_t^{\mathrm{T}}\; {w_t}^{\mathrm{T}}\; {v_{t+1}}^{\mathrm{T}}] \in \mathbb{R}_+^{p_{\xi_t}}$ be the Lagrange multipliers associated with the new set of inequality constraints defined by (3.13). The *equality relaxed* (ER) subproblems can be defined as follows:

$$\mathbf{V}_t^* = \underset{\xi_t}{\text{minimize}}\; \mathbf{V}_t(\xi_t) \;\text{ subject to: }\; G_{\xi_t}\xi_t + g_{\xi_t} \leq 0,\; t=0,\ldots,N. \tag{3.14}$$

These subproblems, however, only anticipate the possible violation of the consensus constraints when the PDFG terminates in a finite number of iterations. In the next section, we show how to further modify the feasible region of these subproblems to ensure that their solution is also primal feasible.

### 3.3.3 Tightening of the Original Inequality Constraints

We introduce $N+1$ auxiliary subproblems, namely the *inequality tightened* (IT) subproblems, which differ from the ER subproblems (3.14) in the definition of the feasible region, in order to guarantee the primal feasibility of each subproblem using Algorithm 3.1. In particular, each IT subproblem can be defined as follows:

$$\mathbf{V}^*_{\epsilon_t} = \min_{\xi_t} \mathbf{V}_t(\xi_t) \text{ s.t.: } G_{\xi_t}\xi_t + g_{\xi_t} + \epsilon_t \mathbf{1}_{p_{\xi_t}} \leq 0, \tag{3.15}$$

where $\epsilon_t \geq 0$ is the tightening parameter, which depends on the suboptimality level $\eta_t$ that the proposed algorithm can reach within $\bar{k}$ iterations (3.10). According to [78], solving the IT subproblem (3.15) using Algorithm 3.1 ensures, with a proper choice of $\epsilon_t$, that the solution of the IT subproblem (3.15) is primal feasible and suboptimal for subproblem (3.14).

We must compute an upper bound for the optimal Lagrange multiplier, namely $\mu^*_{t,\epsilon_t}$, associated with the IT subproblem (3.15), to define an $\epsilon_t$ similar to the one introduced in [78]. We use an argument similar to the one of Lemma 1 in [80]. In particular, we compute the aforementioned upper bound for $\mu^*_{t,\epsilon_t}$ according to the following lemma.

**Lemma 3.3.1.** *Assume that there exists a Slater vector $\tilde{\phi}_t \in \mathbb{R}^{n_\phi}$ that satisfies the consensus constraints* (3.2) *and $G_t\tilde{\phi}_t + g_t < 0$. Then, there exists $\epsilon_t \geq 0$, $\epsilon_t < \min_{j=1,\dots,p_t}\{-(G_t\tilde{\phi}_t + g_t)_j\}$, and $\epsilon_{z_t}, \epsilon_{z_{t+1}} > \epsilon_t$, such that the upper bound for $\mu^*_{t,\epsilon_t}$ is given by*

$$\|\mu^*_{t,\epsilon_t}\| \leq 2R_{d_t} := 2\frac{\mathbf{V}_t(\tilde{\xi}_t) - d_t(\tilde{\mu}_t)}{\min\limits_{j=1,\dots,p_{\xi_t}}\{[\Gamma_t]_j\}}, \tag{3.16}$$

*where $\Gamma_t := \big[[-(G_t\tilde{\phi}_t+g_t)^T - \epsilon_t \mathbf{1}^T_{p_t}][(\epsilon_{z_t}-\epsilon_t)\mathbf{1}^T_{2n_z}][(\epsilon_{z_{t+1}}-\epsilon_t)\mathbf{1}^T_{2n_z}]\big]^T \in \mathbb{R}^{p_{\xi_t}}$, and $d_t(\tilde{\mu}_t)$ is the dual function for the original subproblem* (3.5) *evaluated at $\tilde{\mu}_t \in \mathbb{R}^{p_{\xi_t}}$.*

*Proof.* See Appendix A. □

Note that $\tilde{\xi}_t := [\tilde{\phi}_t^{\mathrm{T}} (H_1\tilde{\phi}_t)^{\mathrm{T}} (H_1\tilde{\phi}_{t+1})^{\mathrm{T}}]^{\mathrm{T}}$, i.e., $\tilde{\xi}_t$ is computed by using $\tilde{\phi}_t$ and assuming that the consensus constraints (3.2) are satisfied.

*Remark* 3.3.5. Lemma 3.3.1 does not only provide an upper bound for $\|\mu^*_{t,\epsilon_t}\|$, but it also provides guidelines to select the values of $\epsilon_{z_t}$ and $\epsilon_{z_{t+1}}$ as a function of $\min_{j=1,\dots,p_t}\{-(G_t\tilde{\phi}_t + g_t)_j\}$, which only depends on the primal variable $\tilde{\phi}_t$. An alternative way to determine the relaxation parameters is to include $\epsilon_{z_t}$ and $\epsilon_{z_{t+1}}$ in the set of decision variables and penalize them in the cost function as it is usually done to handle soft constraints. This will, however, increase the number of decision variables in the problem formulation and it will have an impact on the original cost. Otherwise, in a more practical implementation of the solver, the ER parameters can be selected offline to be sufficiently small.

These results show how to reformulate Problem (3.1) to return feasible solutions for the independent subproblems (3.3) when using the PDFG. In the remainder of the chapter, we focus on how these changes impact the MPC problem (and on the controlled plant) when the MPC solution is applied in closed-loop. Hence, in the next section, we show how a general MPC problem can be reformulated as Problem (3.1) and how to improve the choice of the tightening parameters to ensure recursive feasibility and closed-loop stability.

## 3.4 General MPC Formulation



Consider the discrete-time linear time-invariant (LTI) system described below:

$$x(t+1) = Ax(t) + Bu(t) \qquad \forall t \geq 0, \tag{3.17a}$$
$$y(t) = Cx(t) + Du(t) \qquad \forall t \geq 0, \tag{3.17b}$$

where $x(t) \in \mathcal{X} \subseteq \mathbb{R}^n$ denotes the states of the system, $u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$ denotes the control inputs, and $y(t) \in \mathcal{Y} \subseteq \mathbb{R}^r$ denotes the outputs of the system. The sets $\mathcal{X}$, $\mathcal{U}$, and $\mathcal{Y}$ are simple proper convex sets (i.e., convex sets that contain the origin in their interior).

In the remainder of the chapter, we consider the following assumption:

**Assumption 3.4.1.** *The pair $(A, B)$ is stabilizable.*

Our goal is to steer the output of system (3.17) to a desired reference value denoted by $y_{\text{ref}}$. Furthermore, we have to take into account the constraints acting on $x$, $u$, and $y$. Hence, we rely on a modified version of the MPC for tracking formulation proposed in [66; 19]. In particular, we can formulate our so-called *original* MPC problem as follows:

$$\mathcal{V}^*(y_{\text{ref}}, x_{\text{init}}) := \underset{x,u,\theta}{\text{minimize}} \sum_{t=0}^{N} l_t(y_{\text{ref}}, x_t, u_t, \theta_t) \tag{3.18a}$$
$$\text{subject to: } Ax_t + Bu_t = x_{t+1}, \quad t = 0, \dots, N, \tag{3.18b}$$
$$\begin{bmatrix} \hat{x}_t \\ \hat{u}_t \end{bmatrix} = M_\theta \theta_t, \quad t = 0, \dots, N, \tag{3.18c}$$
$$\bar{C}x_t + \bar{D}u_t + g \leq 0 \quad t = 0, \dots, N, \tag{3.18d}$$
$$\bar{C}\hat{x}_t + \bar{D}\hat{u}_t + g \leq 0 \quad t = 0, \dots, N, \tag{3.18e}$$
$$\hat{y}_t = N_\theta \theta_t \quad t = 0, \dots, N, \tag{3.18f}$$
$$x_0 := x_{\text{init}}, \tag{3.18g}$$
$$(x_N, \theta_N) \in \Omega^w_{t,K} \tag{3.18h}$$

where $x_t \in \mathbb{R}^n$, $u_t \in \mathbb{R}^m$ indicate the $t$-step-ahead state and control predictions, respectively. Furthermore, $\theta_t \in \mathbb{R}^m$ is the vector of parameters used to generate

the artificial steady state, input, and output $\hat{x}_t$, $\hat{u}_t$, and $\hat{y}_t$, respectively. $M_\theta$ and $N_\theta$ are suitable matrices (refer to [66] for details). In addition, (3.18b) represents the prediction model dynamics, while (3.18d) represents the constraints on the predicted state, input, and output ($\bar{C} \in \mathbb{R}^{c\times n}$, and $\bar{D} \in \mathbb{R}^{c\times m}$). For a prediction horizon of length $N$, $l_t$ in (3.18a) is described as follows:

$$l_t(y_{\text{ref}}, x_t, u_t, \theta_t) := \|x_t - \hat{x}_t\|_Q^2 + \|u_t - \hat{u}_t\|_R^2 + \alpha\|\hat{y}_t - y_{\text{ref}}\|^2,$$

where $Q = Q^{\mathrm{T}} \in \mathbb{S}^n_{\geq 0}$, $R = R^{\mathrm{T}} \in \mathbb{S}^m_{>0}$, and $\alpha > 0$. For $t = N$, the state weighting matrix is replaced by $P \in \mathbb{S}^n_{>0}$, which is computed as the solution of the algebraic Riccati equation associated with the unconstrained infinite-horizon linear quadratic regulator (IH-LQR), characterized by the matrices $A$, $B$, $Q$, and $R$. Finally, the proposed problem formulation relies on $\Omega^w_{t,K} \subseteq \mathbb{R}^{n+m}$, which is an invariant set for tracking. In particular, the following assumption holds:

**Assumption 3.4.2** ([66; 19])**.** *Given the gain $K \in \mathbb{R}^{m\times n}$ obtained by the IH-LQR—characterized by the matrices A, B, Q, and R—and $L = [-K\ I]M_\theta \in \mathbb{R}^{m\times m}$, then* $\forall\,(x,\,\theta) \in \Omega^w_{t,K}$, $((A + BK)x + BL\theta,\,\theta) \in \Omega^w_{t,K}$.

*Remark* 3.4.1. Compared to the design proposed in [66; 19], we consider potentially different values of $\theta$ along the length of the prediction horizon to allow more flexibility in the computation of the artificial reference.

In general, the MPC controller solves Problem (3.18) online and returns an optimal sequence of states and control inputs that minimizes the cost (3.18a). Let the optimal sequence be defined as follows:

$$\{\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}\} := \{x_0, \ldots, x^*_N, u^*_0, \ldots, u^*_{N-1}, \theta^*_0, \ldots, \theta^*_N\}. \tag{3.19}$$

Only the first element of $\mathbf{u}$ is implemented in closed-loop, i.e., the control law obtained using the MPC controller is given by:

$$\kappa_{\text{MPC}}(y_{\text{ref}}, x_{\text{init}}) = u^*_0, \tag{3.20}$$

and the closed-loop system is described by

$$x(t+1) = Ax(t) + B\kappa_{\text{MPC}}(y_{\text{ref}}, x_{\text{init}}). \tag{3.21}$$

In the remainder of Section 3.4 and 3.5, we derive our results for a regulation problem, i.e., by assuming $y_{\text{ref}} = 0$ and the associated $\theta_t$, $\hat{x}_t$, $\hat{u}_t$, and $\hat{y}_t$ equal to zero in Problem (3.18), to simplify the discussion and notation, however the numerical example in Section 3.6 will consider a more general tracking example. In particular,

for analysis and illustrative purposes only, we consider the following problem:

$$\mathcal{V}^*(x_{\text{init}}) = \underset{x,u}{\text{minimize}}\ \frac{1}{2}\sum_{t=0}^{N-1}(x_t^T Q x_t + u_t^T R u_t) + x_N^{\mathrm{T}} P_N x_N \tag{3.22a}$$

$$\text{subject to: } x_{t+1} = Ax_t + Bu_t, \qquad t=0,\dots,N-1 \tag{3.22b}$$

$$\bar{C}x_t + \bar{D}u_t + g \leq 0,\ t=0,\dots,N-1 \tag{3.22c}$$

$$x_0 = x_{\text{init}} \tag{3.22d}$$

$$x_N \in \mathcal{X}_N, \tag{3.22e}$$

where $\mathcal{X}_N := \{x \in \mathbb{R}^n \,|\, F_N x \leq f_N,\ F_N \in \mathbb{R}^{c_N \times n}, f_N \in \mathbb{R}^{c_N}\}$ is used as terminal invariant set according to the following assumption (a particular case of Assumption 3.4.2 for Problem (3.22)):

**Assumption 3.4.3.** *Suppose Assumption 3.4.1 holds. Given the gain $K \in \mathbb{R}^{m\times n}$ obtained by the IH-LQR, characterized by the matrices A, B, Q, and R, $\forall\, x \in \mathcal{X}_N$, then $x \in \mathcal{X},\ Ku \in \mathcal{U}$, and $(A+BK)x \in \tau\mathcal{X}_N,\ 0 < \tau \leq 1$.*

In the following, we show how to reformulate Problem (3.22) for Algorithm 3.1.

We aim to solve Problem (3.22) online, i.e., every time new measurements are available from the plant, using Algorithm 3.1. According to Section 3.3, we can exploit a similar approach as the one proposed in [121] and decompose Problem (3.22) along the length of the prediction horizon $N$ into $N+1$ independent subproblems to be solved by $N+1$ parallel workers $\Pi_t$ ($t=0,\dots,N$).

The coupling in Problem (3.22) is caused by the dynamics (3.18b). Hence, the auxiliary $z_t \in \mathbb{R}^n$ in (3.2) are used to store the local predicted state $x_{t+1}$ of each subproblem and exchange this stored information to guarantee consensus between neighboring subproblems, i.e., to ensure that the predicted state of the $(t)$-th subproblem, namely $x_{t+1}^{(t)}$, is equal to the current state of the $(t+1)$-st subproblem, namely $x_{t+1}^{(t+1)}$ (the superscript $(t)$ is used to indicate that $x_t$ belongs to Worker $\Pi_t$). Specifically, the *consensus constraints* (3.2) can be reformulated as $z_{t+1} = x_{t+1}^{(t)} = x_{t+1}^{(t+1)}$ and, by defining $\phi_t := [x_t^{(t)\,\mathrm{T}}\ u_t^{(t)\,\mathrm{T}}]^{\mathrm{T}}$, $H_1 := [I_n\ 0]$, $H_2 := [A\ B]$, $\bar{\mathbf{V}}_t := \text{blkdiag}\{Q, R\}$ (for $t = 0,\dots,N-1$), $\bar{\mathbf{V}}_N := H_1^{\mathrm{T}} P H_1$, $G_t := [\bar{C}\ \bar{D}]$, $g_t := g$ (for $t = 0,\dots,N-1$), $G_N := [F_N\ 0_{c_N\times m}]$, and $g_N := -f_N$ Problem (3.22) is equivalent to Problem (3.3). In addition, Assumption 3.3.1 is satisfied given the definition of $Q$ and $R$, while Assumption 3.3.2 is satisfied if Assumption 3.4.1 holds. Hence, we can use the PDFG to solve Problem (3.22) (or, in the more general case, Problem (3.18)). We have to take into account, however, that the PDFG solver is an *inexact* solver due to the finite number of iterations it performs. Hence, the proposed choice of ER and IT parameters (Sections 3.3.2 and 3.3.3) might not be enough, from the control perspective, to formally ensure recursive feasibility and closed-loop stability using the solution computed using Algorithm 3.1. The next section discusses how to improve the choice of the IT parameters to formally ensure the aforementioned control properties.

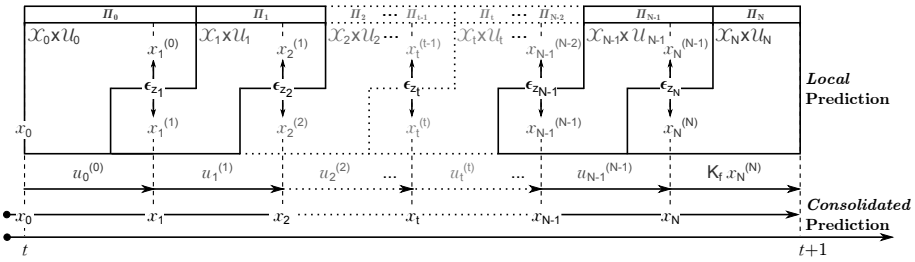


**Figure 3.4:** *Local* and *consolidated* predictions.

## 3.5 Parallel MPC Using Adaptive ER-IT Parameters

Section 3.4 showed how to reformulate the MPC Problem (3.22) (or Problem (3.18), in the general case) for the PDFG and Section 3.3.3 detailed how to choose the tightening parameter $\epsilon_t$ of each IT subproblem to ensure that the $t$-th *local* solution, i.e., the solution computed by the $t$-th IT subproblem (3.15), is primal feasible for the $t$-th ER subproblem. This section provides guidelines to improve the choice of the tightening parameter of each IT subproblem (3.15) in order to guarantee closed-loop stability and recursive primal feasibility of the *consolidated* solution, i.e., the predictions obtained, starting from the initial state $x_0$, using the control sequence

$$\bar{\mathbf{u}}_\epsilon := \left\{ \bar{u}^{(0)}_{0,\epsilon_0}, \ldots, \bar{u}^{(N-1)}_{N-1,\epsilon_{N-1}} \right\}, \tag{3.23}$$

where the elements of $\bar{\mathbf{u}}_\epsilon$ are computed by the independent IT subproblems (3.15). The consolidated solution (3.23) is a suboptimal solution for the original problem (3.22). Figure 3.4 highlights the difference between the local and the consolidated prediction. In particular, the subproblems (3.15) (represented in the top half of Figure 3.4 as boxes characterized by the dedicated worker, local states, and control commands) compute in parallel $\left(x_0, \bar{u}^{(0)}_{0,\epsilon_0}\right), \ldots, \left(\bar{x}^{(N-1)}_{N-1,\epsilon_{N-1}}, \bar{u}^{(N-1)}_{N-1,\epsilon_{N-1}}\right)$, and $x^{(N)}_{N,\epsilon_N}$, respectively (local predictions). According to the results of Section 3.3, the pair $(\bar{x}^{(t)}_{t,\epsilon_t}, \bar{u}^{(t)}_{t,\epsilon_t})$ is primal feasible for the $t$-th subproblem (3.14), thanks to the introduction of the IT subproblems. Nevertheless, due to the relaxation introduced on the equality constraints (3.11a)-(3.11b), there is a bounded mismatch (highlighted in Figure 3.4) between $x^{(t)}_{t+1}$ and $x^{(t+1)}_{t+1}$ $(t = 0, \ldots, N-1)$. Hence, starting from $x_0 := x_{\text{init}}$, when the control sequence $\bar{\mathbf{u}}_\epsilon$ is applied to compute the consolidated state prediction (represented in the bottom half of Figure 3.4)

$$\bar{\mathbf{x}}_\epsilon := \{x_0, \bar{x}_{1,\epsilon_1} \ldots, \bar{x}_{N,\epsilon_N}\}, \tag{3.24}$$

the feasibility of $\bar{\mathbf{x}}_\epsilon$ is no longer guaranteed. Note, however, that $\bar{\mathbf{u}}_\epsilon \in \mathcal{U} := \mathcal{U}_1 \times \ldots \times \mathcal{U}_N$, i.e., $\bar{\mathbf{u}}_\epsilon$ is feasible. Hence, no additional tightening is needed on the input constraints.

In the following, we define an upper bound on the maximal feasibility violation of $\bar{\mathbf{x}}_\epsilon$ in Section 3.5.1. This feasibility violation is a consequence of the local

relaxations of the equality constraints. Then, we introduce sufficient conditions to ensure the primal feasibility of the *consolidated* prediction and provide guidelines for the choice of the tightening parameters for each IT subproblem in Section 3.5.2. Finally, we introduce Algorithm 3.2 to update adaptively the ER-IT parameters and guarantee recursive feasibility and closed-loop stability of the system controlled by the MPC control command (3.20) in Section 3.5.3.

### 3.5.1 Upper Bound on the Maximal Feasibility Violation of $\bar{\mathbf{x}}_\epsilon$

Let $\bar{\mathbf{u}}_\epsilon$ and $\bar{\mathbf{x}}_\epsilon$ be defined by (3.23) and (3.24), respectively. Moreover, from (3.11a) and (3.11b), the following holds:

$$\|\bar{x}^{(t-1)}_{t,\epsilon_{t-1}} - \bar{x}^{(t)}_{t,\epsilon_t}\| \leq 2\epsilon_{z_t}. \tag{3.25}$$

Our goal is to characterize how far the *consolidated* predicted state is from the *local* predicted state.

**Lemma 3.5.1.** *Let the $t$-step-ahead consolidated prediction $\bar{x}_{t,\epsilon_t}$ be defined by* (3.24) *and assume that* (3.25) *holds. Then, there exists $\alpha_t \in \mathbb{R}, \alpha_t \geq 0$, such that the mismatch between $\bar{x}_{t,\epsilon_t}$ and the state of the $t$-th subproblem $\bar{x}^{(t)}_{t,\epsilon_t}$ is bounded, as follows:*

$$\|\bar{x}_{t,\epsilon_t} - \bar{x}^{(t)}_{t,\epsilon_t}\| \leq \alpha_t. \tag{3.26}$$

*Proof.* See Appendix B. □

*Remark* 3.5.1. According to the proof of Lemma 3.5.1 a possible choice for $\alpha_t$ is the following:

$$\alpha_t := 2\sum_{j=0}^{t-1} \|A^j\|\epsilon_{z_{t-j}}. \tag{3.27}$$

### 3.5.2 Selection of the Tightening Parameters

According to Lemma 3.5.1, $\bar{x}_{t,\epsilon_t}$ differs from $\bar{x}^{(t)}_{t,\epsilon_t}$ by a quantity bounded by $\alpha_t$. Thus, $\bar{x}_{t,\epsilon_t}$ might violate the constraints of the $t$-th subproblem (3.3) by as much as $\alpha_t$, in the worst-case scenario. In particular, we must ensure that $\bar{C}_t\bar{x}_{t,\epsilon_t} + \bar{D}_t\bar{u}_{t,\epsilon_t} + g_t \leq 0$. Using the computed upper bound (3.26), the following holds:

$$\begin{gathered}\bar{C}_t\bar{x}^{(t)}_{t,\epsilon_t} + \bar{D}_t\bar{u}^{(t)}_{t,\epsilon_t} + g_t + |\bar{C}_t|\alpha_t\,\mathbf{1}_n + \epsilon_t\,\mathbf{1}_{c_t} \leq 0 \\ \Updownarrow (3.26) \\ \bar{C}_t\bar{x}_{t,\epsilon_t} + \bar{D}_t\bar{u}^{(t)}_{t,\epsilon_t} + g_t + |\bar{C}_t|\alpha_t\,\mathbf{1}_n + \epsilon_t\,\mathbf{1}_{c_t} \leq 0,\end{gathered}$$

where $|\bar{C}_t|$ indicates the absolute value of $\bar{C}_t$.

Recall that these mismatches are caused by the use of inexact solvers and that $\alpha_t$ depends on $\epsilon_{z_t}$. In the following, we provide guidelines to improve the choice of $\epsilon_t$ for each subproblem. Furthermore, we provide a modified upper bound for the optimal Lagrange multiplier associated with the tightened subproblems (3.15), which considers the additional tightening introduced by $\alpha_t$.

**Lemma 3.5.2.** *Consider the following IT subproblems:*

$$\mathbf{V}^*_{\gamma_t} = \min_{\xi_t} \ \mathbf{V}_t(\xi_t) \ \textit{s.t.:} \ G_{\xi_t}\xi_t + g_{\xi_t} + \gamma_t \leq 0, \tag{3.28}$$

*for* $t=0,\ldots,N$, *where* $\gamma_t := [(|\bar{C}_t|\alpha_t \mathbf{1}_n + \epsilon_t \mathbf{1}_{c_t})^T \ \epsilon_t \mathbf{1}_{4n}^T]^T$. *Consider the assumptions of Lemma 3.3.1 and the existence of* $\alpha_t$ *for all* $t=1,\ldots,N$ *according to Lemma 3.5.1. Then, for each subproblem, there exist* $\epsilon_t \geq 0$, $\epsilon_{z_t} > \epsilon_t, \epsilon_{z_{t+1}} > \epsilon_t$ *such that the upper bound for the optimal Lagrange multiplier associated with the IT subproblems* (3.28) *is described by*

$$\|\mu^*_{t,\gamma_t}\| \leq 2\mathcal{R}_t := 2\frac{\mathbf{V}_t(\tilde{\xi}_t) - d_t(\tilde{\mu}_t)}{\min_{j=1,\ldots,c_t+4n}\{[\Gamma_{\alpha_t}]_j\}}, \ t=0,\ldots,N,$$

$\Gamma_{\alpha_t} := \left[[-(G_t\tilde{\phi}_t + g_t)^T - (|\bar{C}_t|\alpha_t \mathbf{1}_n)^T - \epsilon_t \mathbf{1}_{c_t}^T][(\epsilon_{z_t} - \epsilon_t)\mathbf{1}_{2n}^T][(\epsilon_{z_{t+1}} - \epsilon_t)\mathbf{1}_{2n}^T]\right]^T \in \mathbb{R}^{c_t+2n}$.

*Proof.* See Appendix C. □

*Remark* 3.5.2. The choice of $\epsilon_t$ $(t=0,\ldots,N)$ is not unique and depends on the choice of $\epsilon_{z_t}$ $(t=1,\ldots,N)$. For example, given $\alpha_t$ in (3.27), a possible choice of $\epsilon_{z_t}$ $(t=1,\ldots,N)$ is:

$$\epsilon_{z_t} \leq \min\left\{ \frac{\epsilon_{z_N}}{\|A^{N-t}\|}, .., \frac{\epsilon_{z_{t+1}}}{\|A\|}, \frac{\min\limits_{j=1,\ldots,c_t}\{-(G_t\tilde{\phi}_t + g_t)_j\}}{1 + 2t \max\limits_{j=1,\ldots,c_t}\{\sum_{i=1}^n |[\bar{C}_t]_{j,i}|\}} \right\}. \tag{3.29}$$

Consequently, the tightening parameters are given by:

$$\epsilon_t \leq \frac{1}{2}\min\left\{ \epsilon_{z_t}, \epsilon_{z_{t+1}}, \min_{j=1,\ldots,c_t}\{-(G_t\tilde{\phi}_t + g_t)_j\} \right\}, \tag{3.30}$$

for $t=0,\ldots,N$. This choice implies that first we select the relaxation parameters and then we *adapt* the tightening parameters on the original inequality constraints based on the choice of $\epsilon_{z_t}$ for all $t=1,\ldots,N$. An alternative is to fix $\epsilon_t$ for the inequality constraints and consequently compute $\epsilon_{z_t}$. In general, the choice of the parameters strongly depends on the system-state matrix $A$ in (3.17).

*Remark* 3.5.3. In the next section we propose a method (Algorithm 3.2) to adapt the above derived parameters at each problem instance. If we consider a fixed tightening scheme, such as the one proposed by [111], $\epsilon_t$ and $\epsilon_{z_t}$ can be computed offline (for all the initial states in the region of attraction).

*Remark* 3.5.4. We define the ER and the IT parameters to be the same for all the constraints of the subproblem without taking into account the magnitudes of the states/inputs/outputs. This choice is made to simplify the notation in the Lemmas/Theorems. In practice, this choice could be very conservative if there is a large

difference among the magnitudes of the states/inputs/outputs (see Section 3.6, for example). A possible strategy to avoid conservatism is to define a state-transformation to normalize the magnitudes or adopt different ER and IT parameters based on the state/input/output magnitudes.

In the following, we show that by using $\{\bar{\mathbf{x}}_\gamma, \bar{\mathbf{u}}_\gamma\}$—where $\bar{\mathbf{u}}_\gamma$ is the control sequence obtained by solving the IT subproblems (3.28) and $\bar{\mathbf{x}}_\gamma$ is the corresponding consolidated prediction—the inequality constraints of the original MPC problem (3.22) are satisfied. Consequently, the predicted final state is in the terminal set of the original problem.

**Theorem 3.5.1.** *Under the same assumptions of Lemma 3.5.1, if the desired level of suboptimality of Algorithm 3.1 is chosen as:*

$$\eta_t := \frac{1}{2}\epsilon_t \tag{3.31}$$

*then the following holds:*

$$\bar{C}_t \bar{x}_{t,\gamma_t} + \bar{D}_t \bar{u}_{t,\gamma_t} + g_t < 0 \quad t=0,\dots,N,$$

*where $\bar{x}_{t,\gamma_t}$ is the $t$-step-ahead consolidated prediction computed using the solution to the IT subproblem* (3.28) *with tightening parameter $\gamma_t$.*

*Proof.* See Appendix D. □

*Remark* 3.5.5. The IT parameters depend on $\eta_t$ (3.31) that is related to the number of iterations (3.10). If $\bar{k}$ is fixed (e.g., according to the available computational resources) $\eta_t$ can be computed and the IT parameters can be selected accordingly to ensure real-time performance. It could happen, however, that their choice might lead to primal infeasible solutions. How to handle these scenarios is part of our future work. For the proposed example in Section 3.6, $\bar{k}$ is fixed and the ER-IT parameters are tuned offline to ensure that the solution remains indeed feasible.

In summary, this section showed that there exists a choice of the relaxation and tightening parameters that guarantee a feasible consolidated prediction with respect to the original problem (3.22).

### 3.5.3 Suboptimality, Recursive Feasibility, and Closed-Loop Stability

In the following, we derive bounds for $\mathbf{V}_\gamma := \sum_{t=0}^{N} \mathbf{V}_t(\bar{\mathbf{x}}_\gamma, \bar{\mathbf{u}}_\gamma)$, i.e., the cost obtained using $\{\bar{\mathbf{x}}_\gamma, \bar{\mathbf{u}}_\gamma\}$, with respect to the optimal cost $\mathcal{V}^*$ of the original problem.

**Theorem 3.5.2.** *Assuming that there exist $\epsilon_t$ ($t=0,\dots,N$) and $\epsilon_{z_t}$ ($t=1,\dots,N$) selected according to Lemma 3.5.2, then the following holds:*

$$\mathcal{V}^* \le \mathbf{V}_\gamma \le \mathcal{V}^* + 2\sum_{t=0}^{N} \mathcal{R}_t \sqrt{c_t}\bar{\gamma}_t, \tag{3.32}$$

*where $\bar{\gamma}_t := \epsilon_t + \max_{j=1,\dots,c_t} \left\{\sum_{i=1}^{n} |[\bar{C}_t]_{j,i}|\right\} \alpha_t$.*

*Proof.* See Appendix E. □

Theorem 3.5.2 established the level of suboptimality of the consolidated prediction with respect to the original problem. In particular, the sequence $\{\bar{\mathbf{x}}_\gamma, \bar{\mathbf{u}}_\gamma\}$ is suboptimal for the original problem and satisfies the original inequality constraints (including those associated with $\mathcal{X}_N$).

Recall that for the update of $\mathcal{R}_t$, our algorithm requires a strictly feasible vector $\tilde{\phi}_t$ for (3.3d). Hence, every time new measurements are available from the plant, our algorithm must provide a strictly feasible solution (not necessarily optimal) for the first $c_t$ inequality constraints of each ER subproblem. This strictly feasible solution can be computed according to [78]. In particular, let $\bar{\phi}_\gamma$ be defined as $\bar{\phi}_\gamma := [\bar{\phi}_{0,\gamma_0}^{\mathrm{T}} \dots \bar{\phi}_{N,\gamma_N}^{\mathrm{T}}] = [(x_0^{\mathrm{T}}\ \bar{u}_{0,\gamma_0}^{\mathrm{T}}) \dots (\bar{x}_{N-1,\gamma_{N-1}}^{\mathrm{T}}\ \bar{u}_{N-1,\gamma_{N-1}}^{\mathrm{T}})\ (\bar{x}_{N,\gamma_N}^{\mathrm{T}}))]^{\mathrm{T}}$. Then, a feasible $\tilde{\phi}^+$ at the next problem instance, is given by:

$$\tilde{\phi}^+ = [\bar{\phi}_{\gamma_{[2:N+1]}}((A + BK_f)\bar{x}_{N,\gamma_N})^{\mathrm{T}}]^{\mathrm{T}}. \tag{3.33}$$

We want to show that the cost decreases at each problem instance and, consequently, asymptotic stability can be guaranteed. Hence, we introduce the following theorem.

**Theorem 3.5.3.** *Under Assumption 3.4.3, there exist $\epsilon_{z_t}$, $\epsilon_t \geq 0$ $(t = 0, \dots, N)$ such that $\bar{x}_{N,\gamma_N} \in \mathcal{X}_N$ and $\mathbf{V}_0 \geq \sum_{t=0}^{N} \nu\left(\bar{\gamma}_t^+, \mathcal{R}_t^+\right)$, $\nu\left(\bar{\gamma}_t^+, \mathcal{R}_t^+\right) = \bar{\gamma}_t^+(2\mathcal{R}_t^+\sqrt{c_t})$, where $\bar{\gamma}_t^+$ and $\mathcal{R}_t^+$ represent the updated values of these parameters according to $\tilde{\phi}^+$. Consequently, the following holds:*

$$\sum_{t=0}^{N} \mathbf{V}_t(\bar{\phi}_{t,\gamma_t}^+) \leq \sum_{t=0}^{N} \mathbf{V}_t(\bar{\phi}_{t,\gamma_t}) - \mathbf{V}_0(\phi_0) + \sum_{t=0}^{N} \nu(\bar{\gamma}_t^+, \mathcal{R}_t^+), \tag{3.34}$$

*Proof.* See Appendix F. □

The inequality above shows that the total cost decreases at each problem instance if $\mathcal{X}_N$ is defined according to Assumption 3.4.3 and if the $N$-step-ahead *consolidated* prediction lies in the terminal set.

*Remark* 3.5.6. A possible choice to update $\epsilon_{z_t}$ $(t = 1, \dots, N)$ to fulfill (3.34) is the following:

$$\epsilon_{z_t}^+ \leq \min \left\{ \mathbf{V}_0 \left[ 4N\mathcal{R}_t^+ \sqrt{c_t} \left( 1 + 2t \max_{j=1,\dots,c_t} \left\{ \sum_{i=1}^{n} |[\bar{C}_t]_{j,i}| \right\} \right) \right]^{-1}, \right. \\ \left. \epsilon_{z_t} \text{ satisfying (3.29)} \right\}. \tag{3.35}$$

Consequently, $\epsilon_t$ can be selected according to (3.30) to preserve the definition of the upper bound on the optimal Lagrange multipliers given in Lemma 3.5.2.

*Remark* 3.5.7. The iterative fast gradient method leads to a nonmonotonic improvement of the cost function. Nevertheless, thanks to Theorems 1 and 2, which provide

**Algorithm 3.2** MPC design with adaptive parallel tightening scheme.

1: Given $A, B, \bar{C}$, and $\bar{D}$.
2: Compute offline: $N, K, P, F_N, f_N, \bar{H}_1, \bar{H}_2, H_z, Q$, and $R$.
3: Compute offline the constant matrices in Problem (3.3).
4: **for** $t = 0$ **to** $N$ **do**
5: Compute: initial strictly feasible vector $\tilde{\phi}_t$.
6: Compute: initial tightening according to Lemma 3.5.2.
7: **end for**
8: Measure: initial state $x_{\text{init}}$ at time $t = 0$.
9: **repeat**
10: Measure: initial state $x_{\text{init}}$.
11: **if** $x_{\text{init}} \in \mathcal{X}_N$ **then**
12: Compute: $u = Kx_{\text{init}}$.
13: **else**
14: Compute in parallel (Alg. 1): $\bar{\xi}_{0,\gamma_t}, \ldots, \bar{\xi}_{N,\gamma_N}$ exploiting (3.28).
15: Compute: $u = \bar{u}_{\gamma_0}$.
16: Update: $\tilde{\phi} \leftarrow \tilde{\phi}^+$ according to (3.33).
17: **for** $t = 0$ **to** $N - 1$ **do**
18: Update: $\epsilon_{z_{N-t}} \leftarrow \epsilon^+_{z_{N-t}}$ according to Lemma 3.5.2.
19: **end for**
20: **for** $t = 1$ **to** $N$ **do**
21: Update: $\epsilon_t \leftarrow \epsilon_t^+$ according to Lemma 3.5.2.
22: Update: $\gamma_t \leftarrow \gamma_t^+$ according to Lemma 3.5.2.
23: **end for**
24: **end if**
25: Implement $u$.
26: **end repeat**

a monotonic bound for the cost, we can rely on (3.34). Furthermore, when $V_0 \to 0$ for $x \to 0$, the property $\nu(\cdot) \to 0$ due to the definition of $\mathcal{R}_t$ and (3.35) is no longer used, given that the state will enter the terminal set, which contains the origin in its interior.

Algorithm 3.2 summarizes the main steps needed to obtain a stabilizing control law when the original MPC problem is solved in parallel using the PDFG. If the measured state is in $\mathcal{X}_N$, from Assumption 3.4.3, the state and the control constraints are automatically satisfied without solving the MPC problem in parallel. Algorithm 3.2 must be called every time new measurements are available from the plant to ensure performance in closed loop.

*Remark* 3.5.8. Steps 17-23 of Algorithm 3.2 are the only nonparallel ones of the algorithm (Algorithm 3.1 is fully parallelizable). The main reason is in the adaptive nature of the algorithm (see also Remark 3.5.3). Algorithm 3.2 adapts $\epsilon_t$ and

**Algorithm 3.3** MPC design with simplified tightening strategy.

1: Given $A, B, \bar{C}$, and $\bar{D}$.
2: Compute offline: $N, \bar{H}_1, \bar{H}_2, H_z, Q$, and $R$.
3: Compute offline the constant matrices in Problem (3.3).
4: Compute offline: $\bar{k}$ according to Remark 3.5.5.
5: Compute offline: $\epsilon_t$ and $\epsilon_{z_t}$ for $t = 0, \dots, N$.
6: Compute offline: $\gamma_t$ for $t = 0, \dots, N$.
7: **repeat**
8: Compute in parallel (Alg. 1): $\bar{\xi}_{0,\gamma_t}, \dots, \bar{\xi}_{N,\gamma_N}$ exploiting (3.28).
9: Compute: $u = \bar{u}_{\gamma_0}$.
10: Implement $u$.
11: **end repeat**

$\epsilon_{z_t}$ every time new measurements are available from the plant. A fully parallel Algorithm 3.2 is possible using a fixed tightening strategy, in which $\epsilon_t$ and $\epsilon_{z_t}$ are computed offline.

Algorithm 3.2 formally shows how to rigorously select the ER-IT parameters online. The proposed adaptive strategy has the advantages (discussed also in [78]) that the obtained solutions are less conservative compared to an offline tuning of the same parameters. Furthermore, the effectiveness of the proposed approach has been verified in practice on an academic example presented in [20]. The proposed adaptive strategy, however, is less suitable for real-time computation, given that $\bar{k}$ can vary according to the current state measurements. Hence, in practice, when the real-time requirements are critical for the application, we can consider to use Algorithm 3.3, which is a simplified version of Algorithm 3.2. In particular, Algorithm 3.3 replaces the terminal set and cost, which are challenging to compute for some applications, with a long enough prediction horizon $N$ that is tuned offline to ensure the system to enter a control invariant set within the given $N$ steps [83; 72]. Furthermore, Algorithm 3.3 assumes that the number of iterations $\bar{k}$ is fixed, according to the available computational units (Remark 3.5.5). Hence, the ER-IT parameters are computed offline, as any other tuning parameters, to ensure good closed-loop performance of the algorithm.

*Remark* 3.5.9. Algorithms 3.2 and 3.3 require to store some information concerning the system. Nevertheless, the information that needs to be stored is comparable to the information stored when designing a classical MPC controller (such as, the system matrices, constraint description, weighting matrices, etc.). In particular, compared to solving Problem (3.18), the size of the weighting and constraint matrices does not grow with $N$. The algorithm only needs to store the matrices describing the cost and constraints for subproblem 0, $t$, and $N$, given that the description of the subproblems for $t = 1, \dots, N-1$ is the same. For the proposed aerospace application

discussed in Sections 3.2 and 3.6, compared to the baseline controller, the amount of data might be larger, given that the baseline controller requires only to store some precomputed gains that are updated based on given criteria. Nevertheless, as Section 3.2 shows, if we can overcome the computational burden, the MPC controller can largely improve the performance of the aircraft and simplify other tasks, such as the control reconfiguration after a fault [23].

*Remark* 3.5.10. The on-board control unit on an aircraft uses legacy hardware and software that are certified to guarantee a specified set of operational criteria. Advance mathematical operations, such as QP solvers for MPC, are not available. Our solver, compared to second-order solvers, requires only basic algebraic operations (vector sums and multiplications) supported by any on-board control unit. Compared to a classical Nesterov's dual fast gradient method [78], the memory requirements are smaller (given that the size of the matrices does not grow with $N$, as discussed in Remark 3.5.9). Compared to classical ADMM [8], our algorithm provides an upper bound on the level of suboptimality of the obtained solution and primal infeasibility. Compared to the alternating minimization algorithms (AMA) [35], the proposed algorithm does not have the requirement of strong convexity of the cost functions, which could lead to conservative behaviors of the system controlled by the MPC controller.



## 3.6 Longitudinal Control of a Passenger Aircraft

In this example we use an Airbus civilian aircraft simulator [36; 37] and our aim is to show that the proposed strategy can lead to significant benefits to the MPC controller from the solver perspective motivating further investigations of MPC in the aerospace industry.

We control the aircraft by using an (LTI) MPC controller, as depicted in Figure 3.5. In particular, our MPC design exploits the available linearized models of the aircraft at different points (trim conditions) in the flight envelope to build the predictor as described below. At this stage, we assume that there is no model mismatch, that is, we assume the same aircraft and prediction model. Nevertheless, all the important properties that could affect the performance of the solver (such as conditioning of the problem, number of decision variables, number of constraints, etc.) are preserved in this example.

**Aircraft dynamics.** The discretized (at $T_s = 0.04$ sec) aircraft longitudinal dynamics at a given trim condition are described by the following system:

$$x_{\text{A/C}}(k+1) = A_{\text{A/C}}x_{\text{A/C}}(k) + B_{\text{A/C}}\delta(k), \tag{3.36a}$$

$$y_{\text{A/C}}(k) = C_{\text{A/C}}x_{\text{A/C}}(k) + D_{\text{A/C}}\delta(k), \tag{3.36b}$$

where the state vector $x_{\text{A/C}} := \begin{bmatrix} q & p & v_{\text{g}} & \alpha & \theta & h \end{bmatrix}^{\text{T}}$ includes the pitch rate, roll rate, ground speed, angle of attack, pitch angle, and altitude, respectively, the

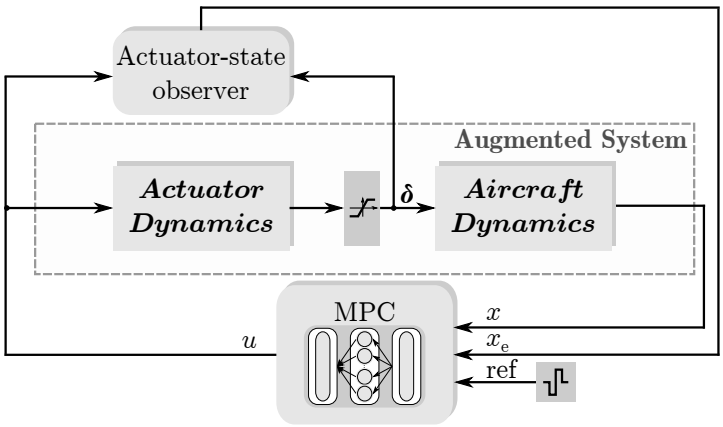


**Figure 3.5:** Proposed Control Architecture.

control input $\delta := \begin{bmatrix}\delta e_{\text{il}} & \delta e_{\text{ir}} & \delta e_{\text{ol}} & \delta e_{\text{or}}\end{bmatrix}^{\text{T}}$ includes the inner-left, inner-right, outer-left, and outer-right elevator outputs, respectively, and the measured outputs $y := \begin{bmatrix}x^{\text{T}} & n_z\end{bmatrix}^{\text{T}}$ are the states $x$ and $n_z$, that is, the vertical load factor (a parameter related to the acceleration on the vertical axis). In particular, the linearized aircraft model is obtained at the following trim condition inside the flight envelope:

$$\begin{aligned} x_{\text{init}}^{\text{T}} &:= \left[\ q,\, p,\, v_{\text{g}},\, \alpha,\, \theta,\, h\ \right]_0 \\ &= \left[\ 0\,\tfrac{\text{deg}}{\text{sec}},\quad 0\,\tfrac{\text{deg}}{\text{sec}},\quad 205\,\text{knots},\quad 2.30\,\text{deg},\quad 2.30\,\text{deg},\quad 12500\,\text{ft}\ \right]. \end{aligned}$$

**Actuator dynamics.** As Figure 3.5 shows, the control input $\delta$ is generated by the output of the actuator system, described by the following equations:

$$x_{\text{act}}(k+1) = A_{\text{act}}x_{\text{act}}(k) + B_{\text{act}}u(k), \tag{3.37a}$$
$$\delta(k) = C_{\text{act}}x_{\text{act}}(k) + D_{\text{act}}u(k), \tag{3.37b}$$

where $x_{\text{act}} \in \mathbb{R}^{12}$ and $u$ is the control command generated by the MPC controller.

**Actuator-state observer.** While dedicated sensors measure the aircraft states $x_{\text{A/C}}$, the actuator states $x_{\text{act}}$ are not available and have to be estimated in order to include them in the MPC prediction model. Given that $u$ and $\delta$ are measured, we design a classical Luenberger state observer characterized by a static gain $L$ to estimate $x_{\text{act}}$, as Figure 3.5 depicts.

**Augmented aircraft dynamics.** The augmented aircraft model used by the MPC controller to build the predictions is given by:

$$x(k+1) = Ax(k) + Bu(k) \tag{3.38a}$$
$$y(k) = Cx(k) + Du(k), \tag{3.38b}$$

where $x(k) := \begin{bmatrix} x_{\text{A/C}}^{\text{T}} & x_{\text{act}}^{\text{T}} \end{bmatrix}^{\text{T}}$ and $y(k) := y_{\text{A/C}}$ are the states and the outputs, respectively.

To control the longitudinal dynamics of the aircraft, the MPC tracks a desired vertical load factor $y_{\text{ref}} := n_{z_{\text{ref}}}$ generated by a pilot-stick command (i.e., the same general tracking scenario as considered in Section 3.2). In particular, the given stick command generates the maximum allowed vertical load factor as reference signal. This reference signal allows to test the behavior of the solver when the output of the system is steered towards its saturation limits. Furthermore, the MPC takes into account the constraints acting on $x(t)$, $u(t)$, and $y(t)$. In particular, the vertical load factor $n_z$ must be constrained to lie between $\overline{n}_z := 2.5$ g and $\underline{n}_z := -1$ g. We use the aforementioned information to formulate Problem (3.18).

Compared to Problem (3.18), the MPC controller in this numerical example cannot rely on a terminal set for tracking, given the size of the considered optimization problem (this is a practical limitation that we have to deal with given that the available algorithms are not able to provide a solution in bounded time) and, for the proposed application, Algorithm 3.3 is used. In this respect, we chose a long enough prediction horizon $N$ to ensure closed-loop stability. In this respect, we fixed $N := 30$ samples (i.e., 1.2 sec). Hence, according to Section 3.3, we can decompose the original problem (3.18) into 31 $(N + 1)$ independent subproblems that can be solved in a parallel, batch, or serialized fashion.

A design issue that we have to take into account concerns the different magnitudes of the state variables that influence the choice of the ER-IT parameters and the conditioning of the system. Compared to the academic example in [20], for the proposed aerospace application, we must take into account that some of the states are angular measurements that can vary of few degrees, while some other states (such as altitude or speed) can vary thousand of units. Hence, if we impose the same amount of relaxation and tightening for all the states the closed-loop performance might be penalized. Hence, we can introduce a suitable state transformation to normalize the system matrices and, consequently, impose the same amount of relaxation/tightening to all the state variables. This strategy can be seen as a preconditioning of the MPC problem and can be beneficial not only for the choice of the ER-IT parameters, but also to improve the convergence of the PDFG algorithm. This strategy led to a significant performance improvement of the solver and motivates further investigation of preconditioning strategies when using first-order solvers as part of our future work.

Finally, given the absence of the terminal set and the small computation time, according to Algorithm 3.3, we tuned the ER-IT parameters offline to preserve the closed-loop performance of the MPC. In particular, we relaxed each equality constraint according to (3.12a)-(3.12b) by a quantity $\epsilon_{z_t} = 10^{-4}$. Furthermore, we tightened the inequality constraints of the obtained ER-IT subproblems according to (3.15) by a quantity $\epsilon_t = 10^{-2}$.

In the context of this work, we chose to solve the problem in a serialized fashion to allow the comparison with other state-of-the-art solvers that cannot benefit

from parallel hardware architectures. In particular, we compared the closed-loop behavior obtained by solving the MPC problem (3.18) using Algorithm 3.3 (using Algorithm 3.1 in a serialized fashion) with the one obtained using the state-of-the-art Nesterov's dual fast gradient method proposed in [78] (we use the same amount of tightening $\epsilon_t$ and weighting matrices for this solver). We run the closed-loop simulations in MATLAB 64-bit on Windows-based OS with an Intel i7 CPU (at 1.90 GHz). Both solvers are implemented as MATLAB `mex` files. The size (in terms of memory) of the `mex` function obtained for Algorithm 3.3 is only 81 KB, while the size of `mex` file obtained to implement the algorithm in [78] is 1553 KB, i.e., almost 20 times larger than the one obtained using our proposed algorithm. This is an encouraging observation for the future implementation of the algorithm on an embedded platform.

*Remark* 3.6.1. The results obtained in this section rely on floating-point units (FPUs). Nevertheless, several studies show that first-order solvers perform well on embedded platforms that do not have FPUs available and tailored strategies can be derived to ensure reliable operations and convergence of these algorithms when FPUs are not available (as discussed, for example in [53; 92]).

In Figures 3.6-3.8, we use the following notation: the solid black line represents the behavior of the system using the PDFG, the short-dashed grey line represents the behavior of the system using the DFG, the long-dashed black line represents the behavior of the system obtained by using the optimal solution of the MPC problem, the dot-dashed black line represents the reference signal, and the dotted black line represents the bound on the quantity of interest.

Figure 3.6 shows the computation time required for online optimization $t_{\text{opt}}$ obtained using the two solvers in real-time compared with the sampling time of the system. The maximum number of iterations of each solver is fixed to satisfy the real-time requirements (50 iterations for the PDFG and 80 for the DFG). Note that the iterations of the PDFG are more expensive from the computational point of view, given that the algorithm runs in serialized mode. Nevertheless, its computational cost can be significantly reduced even further by exploiting parallel hardware architecture. The smaller number of iterations of the PDFG, however, does not impact the closed-loop performance, in fact as Figures 3.7 and 3.8 show. In particular, Figure 3.7 shows the behavior of the vertical load factor, while Figure 3.8 shows the control command allocated to one of the elevators. The PDFG outperforms the DFG in closed loop. In particular, note that after approximately 3 sec from the beginning of the simulation (Figures 3.7 and 3.8), the quality of the solution obtained within the available number of iterations using the DFG decreases causing the closed-loop behavior to diverge from the optimal one (long-dashed black line), while the quality of the solution obtained using the PDFG (solid black line) remains higher (as the comparison with the optimal one highlights), despite the limited number of iterations. The main reason is related to the improved numerical properties of the subproblems. Furthermore, each subproblem has a smaller dimension, that is, less

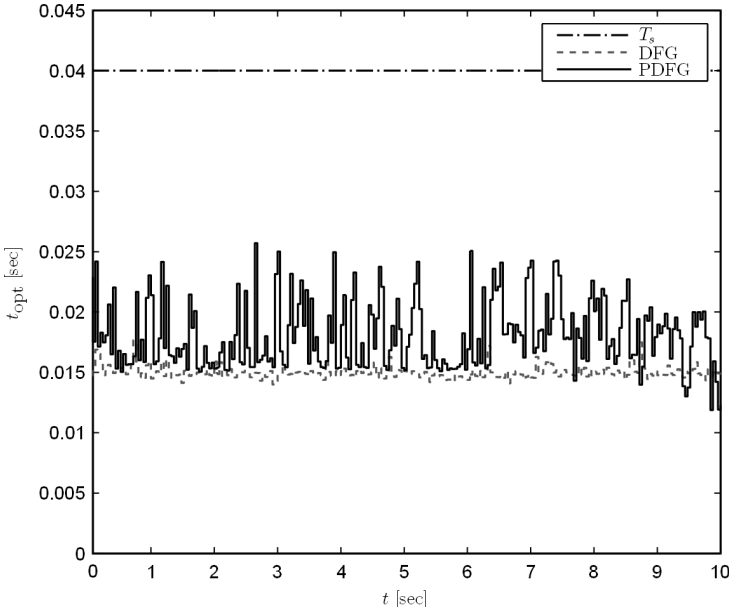


**Figure 3.6:** Computation time.

decision variables, compared to Problem (3.18).

These results confirm the benefits highlighted in [20] for regulation problems and show the potential that the algorithm has in practical applications.

## 3.7 Conclusions

We presented a parallel MPC solver, which is very promising for applications with strong real-time requirements, such as aerospace applications. The proposed algorithm, compared to state-of-the-art first-order solvers shows better conditioning and convergence to a suboptimal solution within a smaller number of iterations. Furthermore, we showed how to use the proposed solver on a practical example, i.e., the longitudinal control of an Airbus passenger aircraft. The main purpose of this example was to show the performance of the proposed solver on a realistic application with hard real-time constraints. Our solver can ensure computation of the model predictive control problem solution within the sampling time of the system (compared to second-order solvers) by using simple algebraic operations (making the controller more suitable for embedded applications).

## A Proof of Lemma 3.3.1 in Section 3.3.3

*Proof.* The following inequality holds:

$$\begin{aligned} d_t(\tilde{\mu}_t, \tilde{\xi}_t) &\leq \mathbf{V}_t(\tilde{\xi}_t) + \mu_{t,\epsilon_t}^{*\mathrm{T}} \nabla_{\mu_t}^{\mathrm{T}} d_{t,\epsilon_t}(\mu_t, \xi_t) \\ &= \mathbf{V}_t(\tilde{\xi}_t) + \lambda_{t,\epsilon_t}^{*\mathrm{T}}(G_t\tilde{\phi}_t + g_t + \epsilon_t \mathbf{1}_{c_t}) + \\ &\quad + w_{t,\epsilon_t}^{*\mathrm{T}}(\bar{H}_1\tilde{\phi}_t - H_z z_t - \epsilon_{z_t}\mathbf{1}_{2n_z} + \epsilon_t \mathbf{1}_{2n_z}) + \\ &\quad + v_{t+1,\epsilon_t}^{*\mathrm{T}}(\bar{H}_2\tilde{\phi}_t - H_z z_{t+1} - \epsilon_{z_{t+1}}\mathbf{1}_{2n_z} + \epsilon_t \mathbf{1}_{2n_z}) \\ &\leq \mathbf{V}_t(\tilde{\xi}_t) + \lambda_{t,\epsilon_t}^{*\mathrm{T}}(G_t\tilde{\phi}_t + g_t + \epsilon_t \mathbf{1}_{c_t}) + w_{t,\epsilon_t}^{*\mathrm{T}}(-\epsilon_{z_t} + \epsilon_t)\mathbf{1}_{2n_z} + \\ &\quad + v_{t+1,\epsilon_t}^{*\mathrm{T}}(-\epsilon_{z_{t+1}} + \epsilon_t)\mathbf{1}_{2n_z}, \end{aligned} \tag{3.39}$$

where the last inequality takes into account that $\tilde{\phi}_t$ satisfies the consensus constraints (3.2) at equality. Consequently, the following holds:

$$\|\mu_{t,\epsilon_t}^*\| \leq \|[\lambda_{t,\epsilon_t}^{*\mathrm{T}}\ w_{t,\epsilon_t}^{*\mathrm{T}}\ v_{t+1,\epsilon_t}^{*\mathrm{T}}]^{\mathrm{T}}\|. \tag{3.40}$$

Furthermore, recalling that $\lambda_{t,\epsilon_t}^* \in \mathbb{R}_+^{c_t}$, $w_{t,\epsilon_t}^* \in \mathbb{R}_+^{n_z}$, and $v_{t+1,\epsilon_t}^* \in \mathbb{R}_+^{n_z}$, the following holds:

$$\|[\lambda_{t,\epsilon_t}^{*\mathrm{T}}\ w_{t,\epsilon_t}^{*\mathrm{T}}\ v_{t+1,\epsilon_t}^{*\mathrm{T}}]^{\mathrm{T}}\| \leq [\lambda_{t,\epsilon_t}^{*\mathrm{T}}\ w_{t,\epsilon_t}^{*\mathrm{T}}\ v_{t+1,\epsilon_t}^{*\mathrm{T}}]^{\mathrm{T}}\mathbf{1}_{p_{\xi_t}}. \tag{3.41}$$

Hence, if we compute an upper bound for the vector $[\lambda_{t,\epsilon_t}^{*\mathrm{T}}\ w_{t,\epsilon_t}^{*\mathrm{T}}\ v_{t+1,\epsilon_t}^{*\mathrm{T}}]^{\mathrm{T}}$, we obtain an upper bound for $\|\mu_{t,\epsilon_t}^*\|$. Thus, from the inequality (3.39), it follows that:

$$\begin{bmatrix} \lambda_{t,\epsilon_t}^* \\ w_{t,\epsilon_t}^* \\ v_{t+1,\epsilon_t}^* \end{bmatrix}^{\mathrm{T}} \underbrace{\begin{bmatrix} -(G_t\tilde{\phi}_t + g_t) - \epsilon_t \mathbf{1}_{c_t} \\ (\epsilon_{z_t} - \epsilon_t)\mathbf{1}_{2n_z} \\ (\epsilon_{z_{t+1}} - \epsilon_t)\mathbf{1}_{2n_z} \end{bmatrix}}_{\Gamma_t} \leq \mathbf{V}_t(\tilde{\xi}_t) - d(\tilde{\mu}_t, \tilde{\xi}_t). \tag{3.42}$$

Notice that choosing $\epsilon_t < \min_{j=1,\ldots,c_t}\{-(G_t\tilde{\phi}_t + g_t)_j\}$, $\epsilon_{z_t} > \epsilon_t, \epsilon_{z_{t+1}} > \epsilon_t$, i.e., according to the assumptions of the lemma, the elements of $\Gamma_t$ are all greater than zero.

Thus, using (3.40) and (3.41) leads to

$$\frac{1}{2}\min_{j=1,\ldots,p_{\xi_t}}\{[\Gamma_t]_j\}\|\mu_{t,\epsilon_t}^*\| \leq \begin{bmatrix}\lambda_{t,\epsilon_t}^* & w_{t,\epsilon_t}^* & v_{t+1,\epsilon_t}^*\end{bmatrix}\Gamma_t. \tag{3.43}$$

Consequently, the upper bound on the optimal Lagrange multiplier is given by:

$$\|\mu_{t,\epsilon_t}^*\| \leq 2\frac{\mathbf{V}_t(\tilde{\xi}_t) - d_t(\tilde{\mu}_t, \tilde{x}i_t)}{\min_{j=1,\ldots,p_{\xi_t}}\{[\Gamma_t]_j\}}.$$

□

## B Proof of Lemma 3.5.1 in Section 3.5.1

*Proof.* In the following, we omit the dependence on $\epsilon_t$ to simplify the notation. The proof is constructive. For $t = 0$, $x_0 \equiv \bar{x}_0^{(0)}$. For $t = 1$, $\bar{x}_1 = Ax_0 + B\bar{u}_0 \equiv \bar{x}_1^{(0)}$, which is the 1-step-ahead state computed by the subproblem associated with worker $\Pi_0$. Hence, the mismatch between $\bar{x}_1$ and $\bar{x}_1^{(1)}$ is simply given by

$$\|\bar{x}_1 - \bar{x}_1^{(1)}\| \leq 2\epsilon_{z_1} = \alpha_1.$$

For $t=2,\ldots,N$, the following holds:

$$\begin{aligned}
\left\|\bar{x}_2 - \bar{x}_2^{(2)}\right\| &= \left\|\bar{x}_2 - \bar{x}_2^{(1)} + \bar{x}_2^{(1)} - \bar{x}_2^{(2)}\right\| \leq \left\|\bar{x}_2 - \bar{x}_2^{(1)}\right\| + \left\|\bar{x}_2^{(1)} - \bar{x}_2^{(2)}\right\| \\
&\leq \left\|A\bar{x}_1^{(0)} + B\bar{u}_1 - A\bar{x}_1^{(1)} - B\bar{u}_1\right\| + 2\epsilon_{z_2} \leq 2(\|A\|\epsilon_{z_1} + \epsilon_{z_2}) = \alpha_2, \\
&\vdots \\
\left\|\bar{x}_N - \bar{x}_N^{(N)}\right\| &\leq 2(\|A^{N-1}\|\epsilon_{z_1} + \|A^{N-2}\|\epsilon_{z_2} + \ldots + \epsilon_{z_N}) = \alpha_N,
\end{aligned}$$

which proves the lemma. □

## C Proof of Lemma 3.5.2 in Section 3.5.2

*Proof.* This lemma follows from Lemma 3.3.1 applied to the subproblems (3.28). From inequality (3.42) formulated for subproblem (3.28), the following must hold

$$\begin{bmatrix} \lambda^*_{t,\epsilon_t} \\ w^*_{t,\epsilon_t} \\ v^*_{t+1,\epsilon_t} \end{bmatrix}^{\mathrm{T}} \underbrace{\begin{bmatrix} -(G_t\tilde{\phi}_t + g_t) - |\bar{C}_t|\alpha_t\,\mathbf{1}_n - \epsilon_t\mathbf{1}_{c_t} \\ (\epsilon_{z_t} - \epsilon_t)\mathbf{1}_{2n} \\ (\epsilon_{z_{t+1}} - \epsilon_t)\mathbf{1}_{2n} \end{bmatrix}}_{\Gamma_{\alpha_t}} \leq \mathbf{V}_t(\tilde{\xi}_t) - d_t(\tilde{\mu}_t).$$

Hence, in order to satisfy the inequality above, we can select the relaxation parameters $\epsilon_{z_t}$ and the tightening parameters $\epsilon_t$ according to the assumption of the lemma for $t=0,\ldots,N$, i.e., the following must hold:

$$\begin{aligned}
&\textit{(i)}\ \frac{1}{2}\min_{j=1,\ldots,c_t}\{-(G_t\tilde{\phi}_t + g_t)_j\} \geq \max_{j=1,\ldots,c_t}\left\{\sum_{i=1}^{n}|[\bar{C}_t]_{j,i}|\right\}\alpha_t + \epsilon_t \\
&\textit{(ii)}\ \max_{j=1,\ldots,c_t}\left\{\sum_{i=1}^{n}|[\bar{C}_t]_{j,i}|\right\}\alpha_t + \epsilon_t > 0 \\
&\textit{(iii)}\ \epsilon_{z_t}, \epsilon_{z_{t+1}} \geq \epsilon_t \geq 0,
\end{aligned}$$

where $\tilde{\phi}_t$ is a strictly feasible solution for the original $t$-th subproblem. Hence, there exists $\mathcal{R}_t$ such that the upper bound on the optimal Lagrange multiplier $\mu^*_{t,\gamma_t}$ is

defined as follows:

$$\|\mu^*_{t,\gamma_t}\| \leq \frac{\mathbf{V}_t(\tilde{\xi}_t) - d_t(\tilde{\mu}_t)}{\min\limits_{j=1,\ldots,c_t+2n}\{[\Gamma_t \alpha_t]\}}.$$

□

## D Proof of Theorem 3.5.1 in Section 3.5.2

*Proof.* If the desired level of suboptimality of Algorithm 3.1 is chosen according to (3.31) then, according to Theorem 3.3.1, there exists $\bar{\xi}_{t,\gamma_t} := [\bar{\phi}^{\mathrm{T}}_{t,\gamma_t}\ \bar{z}^{\mathrm{T}}_{t,\gamma_t}\ \bar{z}^{\mathrm{T}}_{t+1,\gamma_t}]^{\mathrm{T}}$ such that $\|[\nabla^{\mathrm{T}}_{\mu_t} d_t(\bar{\xi}_{t,\gamma_t})]_+\| \leq \eta_t < \epsilon_t$. Using similar arguments as in [78], the following holds for $t=0,\ldots,N$:

$$\left[G_{\xi_t}\bar{\xi}_{t,\gamma_t} + g_{\xi_t} + \begin{bmatrix}|\bar{C}_t|\alpha_t\,\mathbf{1}_n + \epsilon_t\,\mathbf{1}_{c_t}\\ \epsilon_t\,\mathbf{1}_{4n}\end{bmatrix}\right]_+ < \epsilon_t\,\mathbf{1}_{c_t+4n}\,.$$

Hence, for all $j = 1,\ldots,c_t$, the following holds

$$\left[[\bar{C}_t\bar{x}^{(t)}_{t,\gamma_t} + \bar{D}_t\bar{u}^{(t)}_{t,\gamma_t} + g_t + |\bar{C}_t|\alpha_t\,\mathbf{1}_n + \epsilon_t\,\mathbf{1}_{c_t}]_j\right]_+ \leq \epsilon_t.$$

Consequently, exploiting the upper bound (3.26), for all $j = 1,\ldots,c_t$, we have:

$$[\bar{C}_t\bar{x}_{t,\gamma_t} + \bar{D}_t\bar{u}_{t,\gamma_t} + g_t + |\bar{C}_t|\alpha_t\,\mathbf{1}_n + \epsilon_t\,\mathbf{1}_{c_t}]_j \leq \epsilon_t$$

which leads to $\bar{C}_t\bar{x}_{t,\gamma_t} + \bar{D}_t\bar{u}_{t,\gamma_t} + g_t < 0\, t=0,\ldots,N$. □

## E Proof of Theorem 3.5.2 in Section 3.5.3

*Proof.* The feasible region of the tightened subproblems is contained in that of the original ones (as shown in Section 3.5.2). This implies that due to the tightening of the original inequality constraints $\mathbf{V}_\gamma \geq \mathcal{V}^*$.

Recall that the consolidated prediction satisfies the equality constraints (3.22b) by construction. Hence, the following holds:

$$\begin{aligned}\mathbf{V}_\gamma \leq& \sum_{t=0}^{N}\left[\max_{\lambda\geq 0}(\min_{x_t,u_t}\mathcal{V}_t(x_t,u_t) + \langle\lambda, C_t x_t + D_t u_t + g_t\rangle) + \left\langle\lambda^*_{\gamma_t}, [I_{c_t}\ 0]\gamma_t\right\rangle\right]\\ \leq& \mathcal{V}^* + 2\sum_{t=0}^{N}\mathcal{R}_t\sqrt{c_t}\left(\epsilon_t + \max_{j=1,\ldots,c_t}\left\{\sum_{i=1}^{n}|[\bar{C}_t]_{j,i}|\right\}\alpha_t\right).\end{aligned}$$

where $[I_{c_t}\ 0]\,\gamma_t$ selects the first $c_t$ components of the vector $\gamma_t$. □

# F Proof of Theorem 3.5.3 in Section 3.5.3

*Proof.* Using a similar argument as in [103], under Assumption 3.4.3 on $\mathcal{X}_N$ and ensuring that $\bar{x}_{N,\gamma_N} \in \mathcal{X}_N$ (thanks to a proper choice of the tightening parameters, as Section 3.5.2 showed), we can show that:

$$\sum_{t=0}^{N} \mathbf{V}_t(\tilde{\phi}_t^+) \leq \sum_{t=0}^{N} \mathbf{V}_t(\bar{\phi}_{t,\gamma_t}) - \mathbf{V}_0(\phi_0) \; \forall \phi_0 \in \Phi_{\text{attr}}, \tag{3.44}$$

where $\Phi_{\text{attr}}$ is the region of attraction. Hence, from (3.32) and (3.44), the following holds:

$$\sum_{t=0}^{N} \mathbf{V}_t(\bar{\phi}_{t,\gamma_t}^+) \overset{(3.32)}{\leq} \mathcal{V}^*(x^+) + \sum_{t=0}^{N} \nu(\bar{\gamma}_t^+, \mathcal{R}_t^+) \tag{3.45a}$$

$$\leq \sum_{t=0}^{N} \mathbf{V}_t(\tilde{\phi}_t^+) + \sum_{t=0}^{N} \nu(\bar{\gamma}_t^+, \mathcal{R}_t^+) \tag{3.45b}$$

$$\overset{(3.44)}{\leq} \sum_{t=0}^{N} \mathbf{V}_t(\bar{\phi}_{t,\gamma_t}) - \mathbf{V}_0(\phi_0) + \sum_{t=0}^{N} \nu(\bar{\gamma}_t^+, \mathcal{R}_t^+). \tag{3.45c}$$

Asymptotic stability of our controller follows from $\mathbf{V}_0(\phi_0) \geq \sum_{t=0}^{N} \nu(\bar{\gamma}_t^+, \mathcal{R}_t^+)$, which can be satisfied by a proper choice of $\epsilon_t$ and $\epsilon_{z_t}$. □

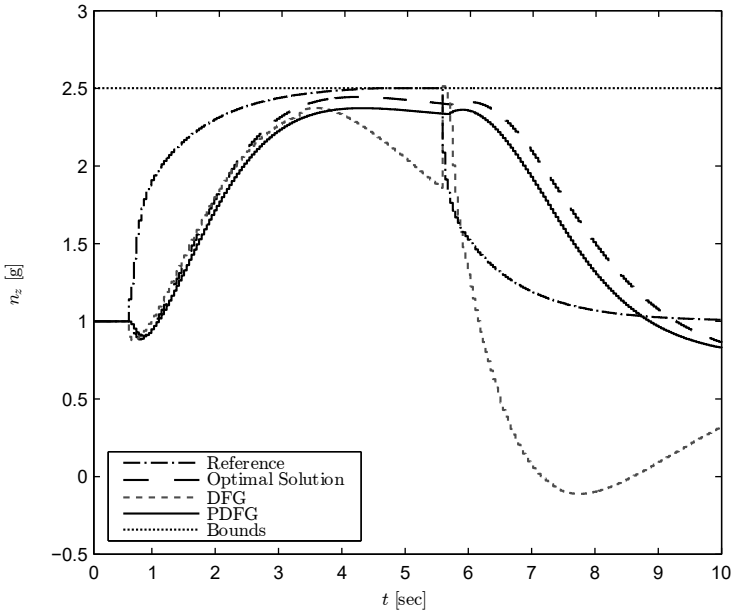


**Figure 3.7:** Vertical load factor.

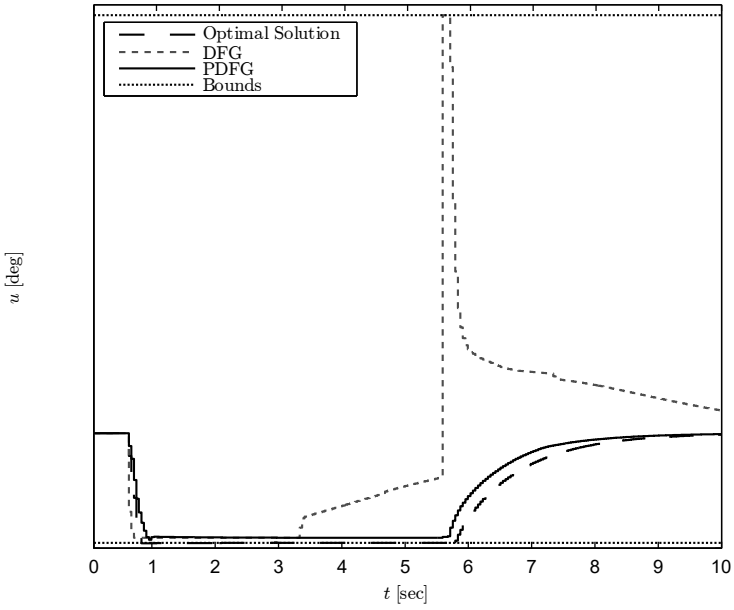


**Figure 3.8:** Allocated elevator command.

Chapter 3

# 4

# SVR-AMA: an Asynchronous Alternating Minimization Algorithm with Variance Reduction for Model Predictive Control Applications




## Abstract

This chapter focuses on the design of an asynchronous dual solver suitable for model predictive control (MPC) applications. The proposed solver relies on a state-of-the-art variance reduction (VR) scheme, previously used in the context of stochastic proximal gradient methods (Prox-SVRG), and on the alternating minimization algorithm (AMA). The resultant algorithm, a stochastic AMA with VR (SVR-AMA), shows geometric convergence (in the expectation) to a suboptimal solution of the MPC problem and, compared to other state-of-the-art dual asynchronous algorithms, allows one to tune the probability of the asynchronous updates to improve the quality of the estimates. Two novel accelerated versions of the Prox-SVRG (and by duality of SVR-AMA) are also provided by relying on the similarity between the stochastic proximal gradient with variance reduction and an inexact proximal-gradient scheme. We apply the proposed algorithm to a specific class of splitting methods, that is, the decomposition along the length of the prediction horizon. Numerical results on the longitudinal control problem of an Airbus passenger aircraft show the benefits that we can gain in terms of computation time when using our proposed solver with an adaptive probability distribution.

## 4.1 Introduction

Model Predictive Control (MPC) applications to systems with fast dynamics are still relatively limited [57; 50]. Applications in fields such as automotive and aerospace have to deal often with embedded legacy systems. These systems usually run on certified (for safety purposes) hardware architectures with limited availability of, for example, parallel computation units and only support a small set of (certified) mathematical functions. In particular, the availability of optimization toolboxes suitable for MPC purposes on these platforms are limited (or nonexistent).

Growing attention has been recently dedicated to the design of *simple* first-order solvers for MPC [106; 90; 121; 20; 96]. These solvers are relatively easy to certify (in terms of level of *suboptimality* of the solution), use only simple algebraic operations, and require little memory. In [121] and [96], operator-splitting methods, such as the alternating direction method of multipliers (ADMM) [8] and the fast alternating minimization algorithm (FAMA) [35], have been used to exploit the MPC problem structure and speed-up the computation of the solution. These algorithms most of the time require frequent exchanges of information at given synchronization points. To reduce the bottleneck af the synchronization points, a solver that can offer more *flexibility* in how the solutions are computed (for example, by allowing asynchronous updates) would be attractive. In this work, we are interested in extending the use of splitting methods, such as AMA, to this asynchronous framework.

*Contribution.* The contribution of the chapter is threefold. First, we propose a novel algorithm, a stochastic alternating minimization algorithm with variance reduction (SVR-AMA), suitable for MPC applications with state and input constraints. The proposed algorithm operates in the dual space and combines the advantages of the variance reduction scheme proposed in [55; 136] for the proximal stochastic gradient method with the alternating minimization algorithm [128]. The result is that the solution of the MPC problem can be computed in an asynchronous fashion (i.e., at each iteration, the algorithm updates a randomly selected subset of the dual variables instead of the whole set of dual variables) and the resultant algorithm has geometric convergence (in the expectation) to the optimal solution. Furthermore, the proposed algorithm allows the use of a generic probability distribution for the asynchronous updates. In addition, the probability distribution can be updated online to improve the quality of the estimates, as our numerical results show. Finally, the algorithm relies on simple algebraic operations, an appealing quality for embedded MPC applications.

Second, we show how Prox-SVRG can be accelerated, by relying on its similarities with the inexact proximal-gradient method in [116] and the existing convergence results in [136]. In particular, first we show how the variance reduction scheme in [136] can be viewed as an error in the calculation of the gradient that converges (according to the analysis of [136]) geometrically in the expectation. By exploiting this observation, we can rely on the analysis of [116] to accelerate the variance-reduction loop (or inner loop) of Prox-SVRG. Second, we derive similar

conclusions for the outer loop of Prox-SVRG by combining the convergence analysis derived in [136] for the inner loop with classical stability arguments based on dynamical systems theory. Finally, we apply the proposed acceleration strategies for Prox-SVRG in the dual framework to derive an inner-accelerated SVR-AMA (IA-SVR-AMA) and an outer-accelerated SVR-AMA (OA-SVR-AMA) that can be used to solve problems that come from MPC applications.

Third, we show how we can use SVR-AMA (the same analysis can be extended to its accelerated versions) for a specific splitting technique, that is, the decomposition along the length of the prediction horizon (or *time splitting* [121]), and present simulation results on a practical aerospace application, that is, the longitudinal control of an Airbus passenger aircraft [37]. The results show that the proposed algorithms (i.e., SVR-AMA and its accelerated versions) are more robust when solving ill-conditioned problems, outperforming synchronous methods in terms of computation time (measured in terms of number of iterations) and suboptimality level of the solution.

*Related Work.* SVR-AMA derives from the application to the dual problem of the proximal stochastic gradient method with variance reduction (Prox-SVRG) proposed in [136] and has been proposed by the same authors of this manuscript in [24]. Compared to [24], we extend the analysis of the proposed algorithm with two different acceleration techniques, which is not trivial, provide additional proofs for the convergence of SVR-AMA, and improve the numerical results. An accelerated version of the inner-loop of Prox-SVRG has been proposed in [84]. Compared to [84], we approach the analysis of the acceleration of the inner-loop from the perspective of an inexact proximal-gradient algorithm, which significantly simplifies the analysis, and we provide guidelines to select the number of inner-loop iterations to exploit both the benefits of the acceleration and variance reduction. Furthermore, an accelerated version of the outer loop of an algorithm similar to Prox-SVRG (i.e., the stochastic dual coordinate ascent method or SDCA [119]) has been proposed in [120]. The algorithm in [120] focused on regularized loss minimization. Compared to our proposed outer-loop acceleration of Prox-SVRG, [120] requires the minimization of a regularized version of the original cost function in the outer loop. Furthermore, our analysis derives from the results of [136] leading to a simplified proof and an algorithm that can be used to minimize the sum of two functions in which one of the two terms does not have to be strongly convex, which is of importance in order to handle control problems.

The investigation of asynchronous dual algorithms for MPC is gaining more attention recently. In [85], for example, an asynchronous dual algorithm is proposed. Compared to [85], SVR-AMA allows the use of a generic (i.e., not necessarily uniform) probability distribution and, consequently, more flexibility in the tuning phase of the algorithm.

Finally, the idea of the time splitting has been previously proposed in [121]. Their work relies on a synchronous ADMM algorithm. In this context, we reformulate the approach for AMA to exploit SVR-AMA.

*Outline.* The chapter is structured as follows. Section 4.2 introduces the MPC problem formulation. Section 4.3 summarizes AMA and Prox-SVRG. Section 4.4 details the two acceleration techniques (Section 4.4.1 describes the acceleration of the inner loop and Section 4.4.2 describes the acceleration of the outer loop). Section 4.5 introduces SVR-AMA and its accelerated versions. Then, Section 4.6 shows how to reformulate the proposed MPC problem for SVR-AMA using the time splitting. Section 4.7 presents numerical results using an aerospace example. Section 4.8 concludes the chapter. Finally, Appendices A-C provide all the proofs contained in the manuscript.

*Notation.* For $u \in \mathbb{R}^n$, $\|u\| = \sqrt{\langle u, u \rangle}$ is the Euclidean norm. Let $\mathbb{C}$ be a convex set. Then, $\mathbf{Pr}_{\mathbb{C}}(u)$ is the projection of $u$ onto $\mathbb{C}$. Let $f : \mathcal{D} \to \mathcal{C}$ be a function. Then, $f^\star(y) = \sup_x(y^{\mathrm{T}}x - f(x))$ and $\nabla f(x)$ are the conjugate function and the gradient of $f(x)$, respectively. Furthermore, $\mathcal{I}_{\mathbb{C}}(\sigma)$ is the indicator function on the convex set $\mathbb{C}$, which is zero if $\sigma \in \mathbb{C}$ and infinity otherwise. Let $A \in \mathbb{R}^{n\times m}$. Then, $\mathrm{eig}_{\max}(A)$ and $\mathrm{eig}_{\min}(A)$ are the largest and the smallest (modulus) eigenvalues of $A^{\mathrm{T}}A$. $P \in \mathbb{S}_+^{n\times n}$ denotes that $P \in \mathbb{R}^{n\times n}$ is positive definite. In addition, let $x \in \mathbb{R}^n$ be a random variable, $\mathbb{E}[x]$ is its expected value. Finally, details on the notions of strong convexity and Lipschitz continuity used in the chapter can be found in [7].

## 4.2 Problem Formulation

Consider the discrete linear time-invariant (LTI) system described by the following equation:

$$x(t+1) = Ax(t) + Bu(t), \quad t = 0, 1, 2, \dots \tag{4.1}$$

The state $x(t) \in \mathbb{R}^n$ and the control input $u(t) \in \mathbb{R}^m$ are subject to the following polyhedral constraints:

$$Cx(t) + Du(t) \leq d, \tag{4.2}$$

where $C \in \mathbb{R}^{p\times n}$ and $D \in \mathbb{R}^{p\times m}$. Note that the definition of the constraints (4.2) can include constraints on $x(t)$ only or on $u(t)$ only. We aim to regulate the state $x(t)$ to the origin using the control input $u(t)$ while respecting the constraints (4.2). This goal can be translated into the following model predictive control (MPC) problem:

$$\min_{x,u} \ \frac{1}{2}\sum_{t=0}^{N} \left(x_t^{\mathrm{T}} Q x_t + u_t^{\mathrm{T}} R u_t\right) \tag{4.3a}$$

$$\text{s.t.: } x_{t+1} = Ax_t + Bu_t \quad t = 0, \dots, N-1 \tag{4.3b}$$

$$Cx_t + Du_t \leq d \qquad t = 0, \dots, N \tag{4.3c}$$

$$x_0 = x_{\text{init}}, \tag{4.3d}$$

where $x_t$ and $u_t$ represent the $t$-step-ahead state and control predictions, respectively, $N$ indicates the length of the prediction horizon, $Q \in \mathbb{S}_+^{n\times n}$, $R \in \mathbb{S}_+^{m\times m}$, and $x_{\text{init}}$ is the initial (measured) state vector. The MPC law implemented in closed loop

**Algorithm 4.1** AMA [128].

Given $\mu_0$, $T$, and $\tau < \sigma_f / \operatorname{eig}_{\max}(H_y)$.
**while** $k = 1, \ldots, T$ **do**
1a. $y_k = \operatorname{argmin}_y \; f(y) + \langle \mu_{k-1}, -H_y y \rangle$.
1b. $z_k = \operatorname{argmin}_z \; g(z) + \langle \mu_{k-1}, -H_z z \rangle + \frac{\tau}{2}\|d - H_y y_k - H_z z\|^2$.
2. $\mu_k = \mu_{k-1} + \tau(d - H_y y_k - H_z z_k)$
**end while**

is given by the first element of the optimal control sequence obtained by solving Problem (4.3), that is, $u_{\text{MPC}} = u_0^*$.

Our goal is to solve Problem (4.3) in an embedded environment. In particular, we assume that explicit MPC [4] cannot be used due to the problem size and that the computational resources are limited, that is, parallel architectures are not available, memory resources are limited, and only simple algebraic operations are supported. With this framework in mind, in the following, we focus on the design of a simple solver for Problem (4.3) that relies on operator-splitting methods (which, for example, usually rely on parallel hardware architectures) and asynchronicity (which allows one to perform updates of a randomly selected subset of variables to reduce the computational effort). The next section introduces the techniques we rely on to solve Problem (4.3), i.e., AMA proposed in [128] and the proximal stochastic gradient descent method with variance reduction (Prox-SVRG) proposed in [136].



## 4.3 Preliminaries

### 4.3.1 Alternating Minimization Algorithm

Consider the following problem:

$$\text{minimize } f(y) + g(z) \tag{4.4a}$$
$$\text{subject to } H_y y + H_z z = d, \tag{4.4b}$$

where $f(y) := \sum_{t=0}^{N} f^{(t)}(y)$ under the following assumptions:

**Assumption 4.3.1.** *$f^{(t)}$ is a strongly convex function and $\sigma_{f^{(t)}}$ denotes its convexity parameter ($t = 0, \ldots, N$).*

**Assumption 4.3.2.** *$f^{(t)}$ has a Lipschitz continuous gradient with modulus $L_{f^{(t)}}$ ($t = 0, \ldots, N$).*

**Assumption 4.3.3.** *$g$ is a convex function not necessarily smooth.*

**Assumption 4.3.4.** *$H_y := \left[ H_y^{(0)^T} \ldots H_y^{(N)^T} \right]^T$. Each $H_y^{(t)}$ indicates the constraint matrix associated with $f^{(t)}(y)$.*

Furthermore, recall the following properties of the conjugate function $f^\star$:

**Lemma 4.3.1** (Thm. 4.2.1 [48])**.** *If $f$ is strongly convex with convexity parameter $\sigma_f$, then $f^\star$ has a Lipschitz continuous gradient with constant $L(\nabla f^\star) = \sigma_f^{-1}$.*

**Lemma 4.3.2** (Thm. 4.2.2 [48])**.** *If $f$ is convex and has a Lipschitz continuous gradient with modulus $L_f$, then $f^\star$ is strongly convex with convexity parameter $L_f^{-1}$.*

A state-of-the-art algorithm to solve Problem (4.4) is AMA [128]. AMA operates as a proximal gradient algorithm (such as ISTA [3]) on the dual of Problem (4.4). Specifically, given the dual of Problem (4.4) (under the assumptions above), described as follows:

$$\underset{\mu\in\mathbb{R}^{n_\mu}}{\text{maximize}}\; D(\mu)\left\{:=-F(\mu)-G(\mu)\right\}, \tag{4.5}$$

where $F(\mu) := \sum_{t=0}^{N} F_t(\mu)$, $F_t(\mu) := f^{(t)^\star}\left(H_y^{(t)^{\mathrm{T}}}\mu\right)$ for $t = 0, \ldots, N$, and $G(\mu) := g^\star(H_z^{\mathrm{T}}\mu) - d^{\mathrm{T}}\mu$, the following holds:

**Lemma 4.3.3.** *If Assumptions 4.3.1-4.3.3 are satisfied, $F(\mu)$ is strongly convex with Lipschitz continuous gradient characterized by Lipschitz constant $L^\star = L(\nabla F) := \mathrm{eig}_{\max}(H_y)\sigma_f^{-1}$. Furthermore $G(\mu)$ is convex with convexity parameter $\sigma_G$.*

*Proof.* We can use Lemmas 4.3.1 and 4.3.2 to derive the properties of $F(\mu)$. Convexity of $G(\mu)$ follows from the properties of the conjugate of a convex function and from the fact that $d^{\mathrm{T}}\mu$ is a linear function. □

*Remark* 4.3.1. If $g(z)$ is the indicator function on the closed convex set $\mathbb{C}$, that is, $g(z) = \mathcal{I}_{\mathbb{C}}(z)$, then $G(\mu)$ is a support function, that is, $G(\mu)$ is a convex set. Furthermore, note that $\sigma_G$ is only required for theoretical purposes and its value is not needed to tune the parameters of Algorithm 4.5.

AMA updates the dual variables $\mu \in \mathbb{R}^{p_\mu}$ as described in Algorithm 4.1. In general, AMA uses only simple algebraic operations (if $y$ and $z$ are unconstrained steps (1a) and (1b) can be performed efficiently) and does not require advanced hardware architectures. Nevertheless, the algorithm requires frequent exchange of information at given synchronization points (e.g., step (1b) requires $y$ computed at step (1a), which can lead to bottlenecks in the computation of the problem solution). Hence, it would be better to have some flexibility in the update strategy. Motivated by this observation, the following section introduces Prox-SVRG used to derive our proposed asynchronous AMA, as described in Section 4.5.

### 4.3.2 Prox-SVRG: Stochastic Proximal Gradient Method with Variance Reduction

Consider the following primal problem:

$$\underset{y\in\mathbb{R}^{n_y}}{\text{minimize}}\; P(y)\left\{:=F(y)+G(y)\right\}, \tag{4.6}$$

**Algorithm 4.2** Prox-SVRG [136].

Given $\tilde{y}_0$, $N$, $\bar{s}$, $\mathcal{I}_N := \{0, \ldots, N\}$, $\eta$, and $T$.
**while** $s \leq \bar{s}$ **do**
  0a. Set $\tilde{y} = \tilde{y}_{s-1}$.
  0b. Set $\tilde{\beta} = \nabla F(\tilde{y})$.
  0c. Set $y_0 = \tilde{y}$.
  0d. Set $\Pi := \{\pi_0, \ldots, \pi_N\}$.
  **for** $k = 1, \ldots, T$ **do**
    1. Pick $i \in \mathcal{I}_N$ randomly according to $\Pi$.
    2. $\beta_k = \tilde{\beta} + \frac{\nabla F_i(y_{k-1}) - \nabla F_i(\tilde{y})}{\pi_i}$
    3. $y_k = \text{prox}_{\eta G}(y_{k-1} - \eta\beta_k)$
  **end for**
  4. $\tilde{y}_s = (1/T)\sum_{k=1}^{T} y_k$.
**end while**

where $F(y) := \sum_{t=0}^{N} F_t(y)$ and $G(y)$ satisfy the following assumptions:

**Assumption 4.3.5.** *$F(y)$ is a strongly convex function with convexity parameter $\sigma_F$ and Lipschitz continuous gradient characterized by a Lipschitz constant $L \leq \sum_{t=0}^{N} L_t$, where $L_t$ are the Lipschitz constants of each $F_t(y)$.*

**Assumption 4.3.6.** *$G(y)$ is a convex function.*

Furthermore, define the *proximal operator* as follows:

$$\text{prox}_{\eta G}(y) := \underset{y \in \mathbb{R}^{n_y}}{\text{argmin}} \left\{ \frac{1}{2} \|y - x\|^2 + \eta G(y) \right\}. \tag{4.7}$$

The main idea behind Prox-SVRG [136] is to eliminate the dependency of the number of iterations (typical of stochastic gradient methods) in the definition of the step size and to reduce the burden in the computation of $\nabla F(y)$ (typical of classical gradient methods). As pointed out in [136], proximal stochastic gradient methods (such as [138; 51]) suffer from sublinear convergence (to a suboptimal solution of Problem (4.6)) given that the step size decreases at each iteration of the algorithm, but behave well when $N$ is large. On the other hand, classical proximal gradient methods require at each iteration of the algorithm to compute the full gradient of $F(y)$, which can be an involved operation if $N$ is large, but the step size is fixed and independent of the number of iterations (leading to better theoretical convergence properties). Hence, Prox-SVRG aims to exploit the benefits of the two techniques as explained below and described in Algorithm 4.2.

Prox-SVRG uses a multistage strategy to gradually reduce the variance in the estimation of the full gradient $\nabla F(y)$ (without computing the actual full gradient at each iteration). In particular, the full gradient of $F(y)$ is updated only every $T$ iterations to reduce the computational effort compared to the classical gradient methods, and the proximal step (step (3)) uses a modified direction $\beta_k$ (step (2)) that

leads to a smaller variance $\mathbb{E}\|\beta_k - \nabla F(y_{k-1})\|^2$ compared to the one obtained using classical stochastic gradient methods $\mathbb{E}\|\nabla F_i(y_{k-1}) - \nabla F(y_{k-1})\|^2$ ($i \in \mathcal{I}_N$), where $\nabla F_i(y_{k-1})$ is used as update direction (refer to [136] for more details). Furthermore, the random sampling (step (1)) is performed on a probability distribution $\Pi := \{\pi_0, \ldots, \pi_N\}$ that does not necessarily have to be uniform, i.e., the algorithm allows more flexibility in the tuning phase by supporting other distributions as well, such as Poisson distributions, normal distributions, etc. Algorithm 4.2 achieves geometric convergence in the expectation, as stated in the following theorem:

**Theorem 4.3.1** (Thm. 3.1 in [136])**.** *Suppose Assumptions 4.3.5 and 4.3.6 hold. Let* $y_* = \operatorname{argmin}_y P(y)$ *and* $L_\Pi := \max_t L_t/\pi_t$*. Assume that* $0 < \eta < 1/(4L_\Pi)$ *and* $T$ *is sufficiently large so that:*

$$\rho := \frac{1}{\eta\sigma_F T(1 - 4\eta L_\Pi)} + \frac{4\eta L_\Pi(T+1)}{T(1 - 4\eta L_\Pi)} < 1. \tag{4.8}$$

*Then, for* $\bar{s} > 1$*, Algorithm 4.2 has geometric convergence in expectation:*

$$\mathbb{E}P(\tilde{y}_{\bar{s}}) - P(y_*) \leq \rho^{\bar{s}} \left[P(\tilde{y}_0) - P(y_*)\right]. \tag{4.9}$$

*Remark* 4.3.2. The dependency on the probability $\pi_t$ in the choice of the step size $\eta$ can be problematic when $\pi_t \to 0$ or when $N \to \infty$ (e.g., the constrained infinite horizon LQR). Nevertheless, this dependency can be removed in the special case in which $F(y) = \sum_{t=1}^{N} F_t(y_t)$, i.e., when the cost is separable in $y_t$. In this scenario, we can select $0 < \eta < 1/(4 \max_t L_t)$. From the MPC perspective, this is very often the case when the dual formulation is used. Hence, this observation, when the algorithm is used in the dual framework, can be very beneficial to improve the choice of the step size and the quality of the MPC solution.

## 4.4 Accelerated Prox-SVRG

In the following we propose two different acceleration strategies for Algorithm 4.2. In particular, Section 4.4.1 describes a strategy to accelerate the inner loop of Prox-SVRG, while Section 4.4.2 describes a strategy to accelerate the outer loop of Prox-SVRG. We rely on the following observation in order to show that we can accelerate Prox-SVRG.

### 4.4.1 Analysis of Algorithm 4.3

Algorithm 4.2 can be interpreted as a proximal gradient method (such as ISTA [3]) in which the gradient of $F$ is computed inexactly, i.e., by using a modified direction $\beta_k$ obtained by using the variance reduction strategy. In particular, Prox-SVRG reduces to the inexact proximal-gradient proposed by [116], in which the full gradient of $F$

**Algorithm 4.3** Inner-loop acceleration of Prox-SVRG.

Given $\tilde{y}_0$, $N$, $\bar{s}$, $\mathcal{I}_N := \{0, \ldots, N\}$ $\eta$, $\gamma := \frac{\sigma_F}{L}$, and $T$.
**while** $s \leq \bar{s}$ **do**
  0a. Set $\tilde{y} = \tilde{y}_{s-1}$.
  0b. Set $\tilde{\beta} = \nabla F(\tilde{y})$.
  0c. Set $y_0 = \tilde{y}$.
  0d. Set $\Pi := \{\pi_0, \ldots, \pi_N\}$.
  **for** $k = 1, \ldots, T+1$ **do**
    1. Pick $i \in \mathcal{I}_N$ randomly according to $\Pi$.
    2. $\beta_k = \tilde{\beta} + \frac{\nabla F_i(\hat{y}_{k-1}) - \nabla F_i(\tilde{y})}{\pi_i}$
    3. $y_k = \text{prox}_{\eta G}(\hat{y}_{k-1} - \eta\beta_k)$
    4. $\hat{y}_k = y_k + \frac{1-\sqrt{\gamma}}{1+\sqrt{\gamma}}(y_k - y_{k-1})$.
  **end for**
  5. $\tilde{y}_s = (1/T)\sum_{k=1}^{T} y_k$.
**end while**

is computed inexactly (step (3) of Algorithm 4.2):

$$y_k = \text{prox}_{\eta G}\left(y_{k-1} - \eta\beta_k\right) \tag{4.10a}$$
$$= \text{prox}_{\eta G}\left(y_{k-1} - \eta\left(\nabla F(y_{k-1}) + e_k\right)\right). \tag{4.10b}$$

The error in the gradient calculation is defined as follows:

$$e_k = \beta_k - \nabla F\left(y_{k-1}\right). \tag{4.11}$$

Hence, Algorithm 4.2 is a particular case of the inexact proximal gradient method (I-PGM) proposed in [116]. As a consequence, we can derive a framework similar to the one proposed in [116] to show the acceleration of Prox-SVRG. In this respect, we proceed as follows:

1. The first step is to use the convergence results in [116, Proposition 4] to derive conclusions on the convergence of the inner loop of Algorithm 4.3.

2. Second, we check that the expectation on the gradient calculation error, $\mathbb{E}\|e_k\|^2$ is bounded and converges linearly to zero, in order to satisfy the assumptions in [116].

3. Third, we show how the acceleration of the inner loop impacts the outer loop of Algorithm 4.3.

**Analysis of the inner loop of Algorithm 4.3** In the following, we reformulate Proposition 4 in [116] for the problem we take into account, that is, we do not consider the error in the calculation of the proximal operator and we consider that we deal with stochastic variables in the inner loop of Algorithm 4.3.

**Proposition 4.4.1.** *Under the same assumptions of Theorem 4.3.1, for $k \geq 0$ and $y_k$ computed according to Algorithm 4.3, the following holds:*

$$\begin{aligned} &\mathbb{E}P(y_k) - P(y_*) \leq \\ &(1-\gamma)^k \left[ \sqrt{2(P(\tilde{y}) - P(y_*))} + \Gamma_k(\tilde{y}) \sqrt{\frac{2}{\sigma_F}} \right]^2, \end{aligned} \tag{4.12}$$

*with*

$$\Gamma_k(\tilde{y}) := \sum_{i=1}^{k} \mathbb{E}\|e_{i-1}(\tilde{y})\|(1-\sqrt{\gamma})^{-i/2}. \tag{4.13}$$

*Proof.* The proof can be found in Appendix A.1. □

**Bound on the variance in Algorithm 4.3** The error in the gradient calculations that we take into account by using Prox-SVRG is a stochastic error and it is bounded (in the expectation) according to Corollary 3.5 in [136]. We can compute the bound on the error when $\beta_k$ is computed according to step (2) in Algorithm 4.3.

**Corollary 4.4.1.** *Consider $\beta_k$ defined as follows ($k = 0, \ldots, T$):*

$$\beta_k = \nabla F(\tilde{y}) + \frac{\nabla F_{i_k}(\hat{y}_{k-1}) - \nabla F_{i_k}(\tilde{y})}{\pi_{i_k}}. \tag{4.14}$$

*In addition, let $L_\Pi = \max_{i=1,\ldots,N} L_i/(\pi_i)$.[1] Conditioned on $y_{k-1}$, we have $\mathbb{E}\beta_k = \nabla F(\hat{y}_{k-1})$ and*

$$E\|e_{k-1}(\tilde{y}_{s-1})\|^2 \leq 4L_\Pi[P(\hat{y}_{k-1}) + P(\tilde{y}_{s-1}) - 2P(y_*)]. \tag{4.15}$$

*Proof.* The proof follows from the one of Corollary 3.5 in [136] by using the update rule for $\beta_k$ in step (2) of Algorithm 4.3. □

*Remark* 4.4.1. Accelerating the inner loop does not affect the upper bound on the error, as can be easily shown by looking at the proof of Corollary 3.5 in [136]. The upper bound above suggests that when $\tilde{y}_s \to y_*$ and $\hat{y}_{k-1} := (1-\alpha)y_{k-1} + \alpha y_{k-2} \to y_*$ ($\alpha = (1-\sqrt{\gamma})/(1+\sqrt{\gamma}) < 1$) the expected error in the gradient calculation is zero, that is, the error goes to zero at the same rate of the estimates (i.e., $\hat{y}_k$ and $\tilde{y}_s$) of the optimal solution of Problem (4.6). As shown in Theorem 4.3.1, in which the convergence of Prox-SVRG is discussed, $\tilde{y}_s \to y_*$ and consequently $\mathbb{E}\|e_k\|$ is guaranteed by design to converge to zero. For $\mathbb{E}\|e_k\|^2$ to decrease to zero linearly (in order to exploit the upper bound provided in Proposition 4.4.1), the following condition must be verified (we omit the dependency on $\tilde{y}_{s-1}$ when it is clear from the context):

$$\mathbb{E}\|e_k\|^2 \leq \frac{1}{k}\mathbb{E}\|e_{k-1}\|^2. \tag{4.16}$$

[1] $i$ in this case is not the iteration counter, but indicates that $L_i$ is associated with the function $F_i$, $i = 1, \ldots, N$.

The condition above can be checked on the error upper bound (4.15). If we formulate this condition, we notice immediately that we can derive guidelines to select the batch size $T$, that is, the number of inner loop iterations. In particular, the following holds:

$$P(y_{k-1}) \leq \frac{1-\alpha-k\alpha}{k(1-\alpha)}P(y_{k-2}) + \frac{\alpha}{k(1-\alpha)}P(y_{k-3}) + \frac{2}{(1-\alpha)}P(y_*) + \frac{1}{k(1-\alpha)}P(\tilde{y}).$$

The right hand side of the equation above is the sum of positive terms if and only if the coefficient of $P(y_{k-2})$ is greater than or equal to zero. Hence, we have to limit the number of inner loop iterations. In particular, we can derive the following upper bound

$$T \leq \left\lceil \frac{2\sqrt{\gamma}}{1-\sqrt{\gamma}} \right\rceil, \tag{4.17}$$

which means that if the problem is ill-conditioned the acceleration is not recommended.

**Analysis of the outer loop of Algorithm 4.3** By exploiting Proposition 4.4.1 we are able to immediately derive an upper bound for the cost computed using Algorithm 4.3 that depends on $\Gamma_k$, as described in the theorem below.

**Theorem 4.4.1.** *Under the same assumptions as Theorem 4.3.1, let $y_*$ be the optimal solution of Problem* (4.6)*, $\gamma := (\sigma_F/L_\Pi)$, and $0 < \eta \leq (1/L_\Pi)$. For $\left\lfloor \frac{2}{\gamma}(1-\gamma) \right\rfloor \leq T \leq \left\lceil \frac{2\sqrt{\gamma}}{1-\sqrt{\gamma}} \right\rceil$ and $\bar{s} > 0$ the following holds:*

$$\mathbb{E}P(\tilde{y}_{\bar{s}}) - P(y_*) \leq \rho_{\textit{IA,I-PGM}}^{\bar{s}}\left[P(\tilde{y}_0) - P(y_*)\right] + \frac{2}{T\sigma_F}\sum_{s=0}^{\bar{s}-1}\rho_{\textit{IA,I-PGM}}^{s}\Gamma(\tilde{y}_s), \tag{4.18}$$

*where $\rho_{\textit{IA,I-PGM}}$ and $\Gamma(\tilde{y}_s)$ are defined as follows:*

$$\rho_{\textit{IA,I-PGM}} := \frac{2}{T\gamma}\left[(1-\gamma) - (1-\gamma)^{T+1}\right] < 1 \tag{4.19a}$$

$$\Gamma(\tilde{y}_s) := \sum_{k=1}^{T}\frac{\mathbb{E}\|e_{k-1}(\tilde{y}_{s-1})\|^2}{(1-\sqrt{\gamma})^k}. \tag{4.19b}$$

*Proof.* The proof can be found in Appendix A.2. □

*Remark* 4.4.2. Note that $T > \frac{2}{\gamma}(1-\gamma)$ is required in order to have $\rho_{\text{IA,I-PGM}} < 1$. If we analyze the upper and lower bound on $T$ in the statement of Theorem 4.4.1, Algorithm 4.3 can be used only when the conditioning of Problem (4.6) is such that the following holds:

$$1 < \gamma + \sqrt{\gamma}, \tag{4.20}$$

which only holds if $1 > \gamma > 0.4$, that is, if the problem is well conditioned.

**Algorithm 4.4** Outer-loop acceleration of Prox-SVRG.

Given $\tilde{y}_0$, $N$, $\bar{s}$, $\mathcal{I}_N := \{0, \dots, N\}$ $\eta$, and $T$.
**while** $s \leq \bar{s}$ **do**
  0a. Set $\tilde{y} = \hat{y}_{s-1}$.
  0b. Set $\tilde{\beta} = \nabla F(\tilde{y})$.
  0c. Set $y_0 = \tilde{y}$.
  0d. Set $\Pi := \{\pi_0, \dots, \pi_N\}$.
  **for** $k = 1, \dots, T$ **do**
    1. Pick $i \in \mathcal{I}_N$ randomly according to $\Pi$.
    2. $\beta_k = \tilde{\beta} + \frac{\nabla F_i(y_{k-1}) - \nabla F_i(\tilde{y})}{\pi_i}$
    3. $y_k = \text{prox}_{\eta G}(y_{k-1} - \eta \beta_k)$
  **end for**
  4. $\tilde{y}_s = (1/T) \sum_{k=1}^{T} y_k$.
  5. $\hat{y}_s = \tilde{y}_s + \frac{1-\sqrt{\gamma}}{1+\sqrt{\gamma}}(\tilde{y}_s - \tilde{y}_{s-1})$.
**end while**

### 4.4.2 Analysis of Algorithm 4.4

In the following, we propose an alternative acceleration strategy of Prox-SVRG. Compared to the results provided in the previous subsection, this section focuses on the acceleration of the outer loop of Prox-SVRG. In this respect, we proceed by analyzing Algorithm 4.4 as follows:

1. First, we check that the acceleration of the outer loop does not affect the bound on the error in the gradient calculations of the inner loop.

2. Second, we provide a proof of convergence of Algorithm 4.4. This proof mainly relies on the convergence analysis in [136].

**Bound on the variance in Algorithm 4.4** In the following, we show that the acceleration of the outer loop does not affect the variance reduction strategy in the inner loop and that the results of Corollary 3.5 in [136] hold to prove the convergence of Algorithm 4.4. In particular, when $\beta_k$ is computed according to step (2) in Algorithm 4.4 the following holds.

**Corollary 4.4.2.** *Consider $\beta_k$ defined as follows:*

$$\beta_k = \nabla F(\tilde{y}) + \frac{\nabla F_{i_k}(y_{k-1}) - \nabla F_{i_k}(\tilde{y})}{\pi_{i_k}}. \tag{4.21}$$

*In addition, let $L_\Pi := \max_i L_i/(N\pi_i)$, $\beta_s := \sum_{k=1}^{T} \beta_k$, and $y_{s-1,avg} := \frac{1}{T}\sum_{k=1}^{T} y_{k-1}$. Conditioned on $y_{k-1}$, the followings hold:*

$$\mathbb{E}\beta_s := \nabla F(y_{avg})$$

*and*

$$\mathbb{E}\|e_s\|^2 \leq \frac{2L_\Pi}{T}\left[P\left(y_{s-1,avg}\right) + P(\tilde{y}) - 2P(y_*)\right]. \tag{4.22}$$

*Proof.* The proof can be found in Appendix B.1. □

Accelerating the outer loop does not affect the upper bound on the error expectation provided by [136]. Furthermore, notice that (4.22) provides guidelines to select the number of inner loop iterations. In particular, if the following holds the error in the gradient calculations decreases linearly with respect to $\tilde{y} \to y_*$ and $y_{s-1,\text{avg}} \to y_*$:

$$T > \lceil 2L_\Pi \rceil. \tag{4.23}$$

**Convergence of Algorithm 4.4** We can now show the convergence of Algorithm 4.4 using the following theorem.

**Theorem 4.4.2.** *Under the same assumptions of Theorem 4.3.1, for $s \geq 0$, the following holds for $\tilde{y}_s$ computed according to Algorithm 4.4:*

$$\mathbb{E}P(\tilde{y}_s) - P(y_*) \leq (1 - \sqrt{\gamma})^s (P(\tilde{y}_0) - P(y_*)). \tag{4.24}$$

*Proof.* The proof can be found in Appendix B.2. □

*Remark* 4.4.3. The outer loop acceleration converges geometrically at a rate that depends on $\gamma := \mu/L$. If $(1 - \sqrt{\gamma}) < \rho$, we can expect to converge to the optimal solution at a faster rate, compared to Algorithm 4.2. If that does not hold, due to the problem conditioning, the acceleration of the outer loop is not beneficial. In general, note that $\rho$ depends on the number of inner-loop iterations $T$ and $\rho << 1$ only if $T$ is large. Hence, in general, we expect that the condition for the acceleration $(1 - \sqrt{\gamma}) < \rho$ holds in many applications to keep the overall computation time of the algorithm bounded (i.e., we expect to keep $T$ small to reduce the computation time of the algorithm).

## 4.5 Stochastic AMA with Variance Reduction and Its Accelerated Versions

Our goal is to solve Problem (4.4) in an asynchronous fashion, that is, by allowing updates of a randomly selected subset of the dual variables at each iteration of the solver. Hence, given that Algorithm 4.2 cannot be directly applied to Problem (4.4), we proceed as explained in Section 4.3.1, that is, we apply Algorithm 4.2 to the dual of Problem (4.4). The resultanting algorithm (SVR-AMA) is described by Algorithm 4.5. In order to derive convergence results for Algorithm 4.5, we consider the dual formulation of [136, Lemma 3.6].

**Lemma 4.5.1.** *Let $D(\mu) := -F(\mu) - G(\mu)$ defined in (4.5), where $\nabla F(\mu)$ is Lipschitz continuous with parameter $L^\star := \sigma_f^{-1}$ (according to Lemma 4.3.1), and $F(\mu)$ and $G(\mu)$ have convexity parameters $\sigma_F := L_f^{-1}$ and $\sigma_G$, respectively. For any $\mu \in \mathrm{dom}(G)$ and $\beta \in \mathbb{R}^{n_\mu}$, define:*

$$\mu^+ = \mathrm{prox}_{\eta G(\mu)}\left(\mu - \eta\beta\right) \tag{4.25a}$$

$$h = \frac{1}{\eta}(\mu - \mu^+) \tag{4.25b}$$

$$\Delta = \beta - \nabla F(\mu), \tag{4.25c}$$

*where $\eta \leq \sigma_f$. Then, for any $\tilde{\mu} \in \mathbb{R}^{n_\mu}$, we have*

$$\begin{aligned} D(\mu^+) - D(\tilde{\mu}) \geq & h^T(\tilde{\mu} - \mu) + \frac{\eta}{2}\|h\|^2 + \\ & \frac{\sigma_F}{2}\|\tilde{\mu} - \mu\|^2 + \frac{\sigma_G}{2}\|\tilde{\mu} - \mu^+\|^2 + \\ & \Delta^T(\mu^+ - \tilde{\mu}). \end{aligned} \tag{4.26}$$

*Proof.* The definition of $\mu^+$ in (4.25a) follows from [35, Theorem 4] (in particular using [35, Theorem 4] it is shown that steps 2 and 4 in Algorithm 4.5 are equivalent to $\mathrm{prox}_{\eta G}\left(\mu - \eta\beta\right)$, which is equivalent to the proximal step 3 of Algorithm 4.2). Then, (4.26) follows from the proof of Lemma 3.6 in [136] by taking into account the definition of $D(\mu)$. □

We can now establish the convergence of Algorithm 4.5.

**Theorem 4.5.1.** *Suppose Assumptions 4.3.1-4.3.4 hold. Let $\mu^* = \mathrm{argmax}_\mu\, D(\mu)$, where $D(\mu)$ is the dual cost defined in (4.5). Let $L_\Pi^\star := \max_{t=0,\ldots,N} \mathrm{eig}_{\max}(H_y)$ $(\pi_t\sigma_f)^{-1} = \max_{t=0,\ldots,N} \pi_t^{-1} L^\star$, $\pi_t \in \Pi$. Assume that $0 < \eta < 1/(4L_\Pi^\star)$ and $T \geq 1$ such that:*

$$\rho^\star := \frac{L_f}{\eta T(1 - 4\eta L_\Pi^\star)} + \frac{4\eta L_\Pi^\star (T+1)}{T(1 - 4\eta L_\Pi^\star)} < 1. \tag{4.27}$$

*Then, for $\bar{s} > 0$, Algorithm 4.5 has geometric convergence in expectation:*

$$D(\mu_*) - \mathbb{E}D(\tilde{\mu}_{\bar{s}}) \leq \rho^{\star \bar{s}} \left[D(\mu_*) - D(\tilde{\mu}_0)\right]. \tag{4.28}$$

*Proof.* The proof can be found in Appendix C.1. □

**Corollary 4.5.1** (Corollary 3.5 in [136] on the dual)**.** *Consider $\beta_k$ defined in step 3 of Algorithm 4.5. Conditioned on $\mu_{k-1}$, the following holds:*

$$\mathbb{E}\beta_k = \nabla F(\mu_{k-1}), \tag{4.29}$$

*and*

$$\mathbb{E}\|\beta_k - \nabla F(\mu_{k-1})\|^2 \leq 4L_\Pi^\star [P(y_k) + P(\tilde{y}) - 2P(y_*)]. \tag{4.30}$$

**Algorithm 4.5** SVR-AMA.

Given $\tilde{\mu}_0$, $N$, $\bar{s}$, $\mathcal{I}_N := \{0, \dots, N\}$, $\eta$, and $T$.
**while** $s \leq \bar{s}$ **do**
  0a. Set $\tilde{\mu} = \tilde{\mu}_{s-1}$, $\tilde{y} = \tilde{y}_{s-1}$.
  0b. Set $\tilde{\beta} = \nabla F(\tilde{\mu})$.
  0c. Set $\mu_0 = \tilde{\mu}$.
  0d. Set $\Pi := \{\pi^{(0)}, \dots, \pi^{(N)}\}$.
  **for** $k = 1, \dots, T$ **do**
    1. Pick $i \in \mathcal{I}_N$ randomly according to $\Pi$.
    2a. $y_k = \operatorname{argmin}_y \; f^{(i)}(y) + \langle \mu_{k-1}, -H_y^{(i)} y \rangle$.
    2b. $z_k = \operatorname{argmin}_z \; g(z) + \langle \mu_{k-1}, -H_z z \rangle + \frac{\eta}{2}\|d - H_y y_k - H_z z\|^2$.
    3. $\beta_k = \tilde{\beta} + \frac{\nabla F_i(\mu_{k-1}) - \nabla F_i(\tilde{\mu})}{\pi_i}$
    4. $\mu_k = \mu_{k-1} - \eta(\beta_k + H_z z_k - d)$
  **end for**
  5. $\tilde{\mu}_s = \frac{1}{T}\sum_{k=1}^{T} \mu_k$, $\tilde{y}_s = \frac{1}{T}\sum_{k=1}^{T} y_k$.
**end while**

*Proof.* The proof can be found in Appendix C.2. □

*Remark* 4.5.1. According to Corollary 4.5.1, by exploiting the definition of $\nabla F(\mu)$, we can relate the upper bound on the variance in the gradient calculations in the dual framework with the primal cost. This observation is useful to preserve (in the dual framework) the results obtained in the primal framework in Sections 4.4.1 and 4.4.2 to select the inner- and outer-loop iterations for $\mathbb{E}\|\beta_k - \nabla F(\mu_{k-1})\|^2 \to 0$.

Sections 4.4.1 and 4.4.2 showed how to accelerate Prox-SVRG, while Section 4.5 showed how Algorithm 4.5 (i.e., SVR-AMA) is equivalent to Algorithm 4.2 (i.e., Prox-SVRG [136]) applied to the dual of Problem (4.6). This observation allows us to formulate the accelerated versions of SVR-AMA, knowing that their convergence can be derived from Theorem 4.4.1 and Theorem 4.4.2 applied to the dual. In this respect, Algorithm 4.6 describes the inner accelerated version of SVR-AMA (or IA-SVR-AMA), while Algorithm 4.7 describes the outer accelerated version of SVR-AMA (or OA-SVR-AMA).

## 4.6 MPC formulation for SVR-AMA

Our aim is to solve the MPC Problem (4.3) presented in Section 4.2 using SVR-AMA and its accelerated versions. Hence, we must show that the MPC Problem (4.3) is a particular case of Problem (4.4).

First, we decompose Problem (4.3) along the length of the prediction horizon $N$ into $N + 1$ smaller subproblems, according to the *time-splitting* strategy proposed in [121]. This results is achieved thanks to the introduction of $N$ consensus variables $z_t \in \mathbb{R}^n$ ($t = 1, \dots, N$) used to break up the dynamic coupling (4.3b). This

**Algorithm 4.6** Inner-loop accelerated SVR-AMA.

Given $\tilde{\mu}_0$, $N$, $\bar{s}$, $\mathcal{I}_N := \{0, \dots, N\}$ $\eta$, $\gamma$, and $T$.
**while** $s \leq \bar{s}$ **do**
0a. Set $\tilde{\mu} = \tilde{\mu}_{s-1}$, $\tilde{y} = \tilde{y}_{s-1}$.
0b. Set $\tilde{\beta} = \nabla F(\tilde{\mu})$.
0c. Set $\hat{\mu}_0 = \mu_0 = \tilde{\mu}$.
0d. Set $\Pi := \{\pi_0, \dots, \pi_N\}$.
**for** $k = 1, \dots, T+1$ **do**
1. Pick $i \in \mathcal{I}_N$ randomly according to $\Pi$.
2a. $y_k = \operatorname{argmin}_y \; f^{(i)}(y) + \langle \hat{\mu}_{k-1}, -H_y^{(i)} y \rangle$.
2b. $z_k = \operatorname{argmin}_z \; g(z) + \langle \hat{\mu}_{k-1}, -H_z z \rangle + \frac{\eta}{2}\|d - H_y y_k - H_z z\|^2$.
3. $\beta_k = \tilde{\beta} + \frac{\nabla F_i(\hat{\mu}_{k-1}) - \nabla F_i(\tilde{\mu})}{\pi_i}$
4. $\mu_k = \hat{\mu}_{k-1} - \eta(\beta_k + H_z z_k - d)$
5. $\hat{\mu}_k = \mu_k + \frac{1-\sqrt{\gamma}}{1+\sqrt{\gamma}}(\mu_k - \mu_{k-1})$.
**end for**
6. $\tilde{\mu}_s = (1/T)\sum_{k=1}^{T} \mu_k, \tilde{y}_s = (1/T)\sum_{k=1}^{T} y_k$.
**end while**

decomposition allows to reformulate Problem (4.3) as follows:

$$\min_{x,u} \; \frac{1}{2}\sum_{t=0}^{N} \left( x_t^{(t)^{\mathrm{T}}} Q x_t^{(t)} + u_t^{(t)^{\mathrm{T}}} R u_t^{(t)} \right) \tag{4.31a}$$

$$\text{s.t.: } z_{t+1} = A x_t^{(t)} + B u_t^{(t)} \quad t = 0, \dots, N-1 \tag{4.31b}$$

$$z_{t+1} = x_{t+1}^{(t+1)} \qquad t = 0, \dots, N-1 \tag{4.31c}$$

$$C x_t^{(t)} + D u_t^{(t)} \leq d \qquad t = 0, \dots, N \tag{4.31d}$$

$$x_0^{(0)} = x_{\text{init}}, \tag{4.31e}$$

where the original dynamic coupling (4.3b) has been replaced by the consensus constraints (4.31b) and (4.31c). Note that we introduced the superscript $t$ to emphasize that $x_t$ and $u_t$ are local variables of the subproblems obtained after the time splitting [121]. Finally, if we introduce $N+1$ additional slack variables $\sigma_t \in \mathbb{R}^p$ to remove the inequality constraints (4.31d) and define $\mathbb{C} := \{\sigma_t \in \mathbb{R}^p \,|\, \sigma_t \geq 0\}$, Problem (4.31) can be written as the sum of the following subproblems:

$$\min_{y_t} \; f_t(y_t) + \sum_{i=1}^{p} \mathcal{I}_{\mathbb{C}}(\sigma_{t_i}) \tag{4.32a}$$

$$\text{s.t.:} \quad w_t : \; z_t = H_1 y_t, \tag{4.32b}$$

$$v_{t+1} : \; z_{t+1} = H_2 y_t, \tag{4.32c}$$

$$\lambda_t : \; \sigma_t = d - G y_t, \tag{4.32d}$$

where we define $y_t^{\mathrm{T}} := [x_t^{(t)^{\mathrm{T}}} \; u_t^{(t)^{\mathrm{T}}}]$, $f_t(y_t) := y_t^{\mathrm{T}} \mathcal{Q} y_t$, $\mathcal{Q} := \operatorname{diag}\{Q, R\}$, $G :=$

**Algorithm 4.7** Outer-loop accelerated SVR-AMA.

Given $\hat{\mu}_0$, $\tilde{y}_0$, $N$, $\bar{s}$, $\mathcal{I}_N := \{0, \dots, N\}$ $\eta$, $\gamma$ and $T$.
**while** $s \leq \bar{s}$ **do**
  0a. Set $\tilde{\mu} = \hat{\mu}_{s-1}$, $\tilde{y} = \tilde{y}_{s-1}$.
  0b. Set $\tilde{\beta} = \nabla F(\tilde{\mu})$.
  0c. Set $\mu_0 = \tilde{\mu}$.
  0d. Set $\Pi := \{\pi_0, \dots, \pi_N\}$.
  **for** $k = 1, \dots, T$ **do**
    1. Pick $i \in \mathcal{I}_N$ randomly according to $\Pi$.
    2a. $y_k = \operatorname{argmin}_y \; f^{(i)}(y) + \langle \mu_{k-1}, -H_y^{(i)} y \rangle$.
    2b. $z_k = \operatorname{argmin}_z \; g(z) + \langle \mu_{k-1}, -H_z z \rangle + \frac{\eta}{2}\|d - H_y y_k - H_z z\|^2$.
    3. $\beta_k = \tilde{\beta} + \frac{\nabla F_i(\mu_{k-1}) - \nabla F_i(\tilde{\mu})}{\pi_i}$
    4. $\mu_k = \mu^{k-1} - \eta(\beta_k + H_z z_k - d)$
  **end for**
  5. $\tilde{\mu}_s = (1/T)\sum_{k=1}^{T} \mu_k$, $\tilde{y}_s = (1/T)\sum_{k=1}^{T} y_k$.
  6. $\hat{\mu}_s = \tilde{\mu}_s + \frac{1-\sqrt{\gamma}}{1+\sqrt{\gamma}}(\tilde{\mu}_s - \tilde{\mu}_{s-1})$.
**end while**

$[C\ D]$, $H_1 := [I_n\ 0_{n\times m}]$, $H_2 := [A\ B]$. Furthermore, for each equality constraint in Problem (4.32), the corresponding Lagrange multipliers have been highlighted.

If we define the following:

$$\mathbf{y}^{\mathrm{T}} = [y_0^{\mathrm{T}} \dots y_N^{\mathrm{T}}],$$

$$f(\mathbf{y}) \;=\; \mathbf{y}^{\mathrm{T}}\mathbf{Q}\mathbf{y} \;=\; \sum_{t=0}^{N} f_t(y_t),$$

$$\mathbf{z}^{\mathrm{T}} = [z_1^{\mathrm{T}} \dots z_N^{\mathrm{T}}\ \sigma_0^{\mathrm{T}} \dots \sigma_N^{\mathrm{T}}],$$

$$\mathbf{Q} \;=\; \mathrm{diag}\{\mathcal{Q} \dots \mathcal{Q}\},$$

$$g(\mathbf{z}) \;=\; \sum_{t=0}^{N} \mathcal{I}_{\mathbb{C}}(\sigma_t),$$

$$h_{y_0}^{\mathrm{T}} = [H_2^{\mathrm{T}} \mid -G^{\mathrm{T}}],$$

$$h_{y}^{\mathrm{T}} = [H_1^{\mathrm{T}}\ H_2^{\mathrm{T}} \mid -G^{\mathrm{T}}],$$

$$h_{y_N}^{\mathrm{T}} = [H_1^{\mathrm{T}} \mid -G^{\mathrm{T}}],$$

$$h_{d_0}^{\mathrm{T}} = h_{d_N}^{\mathrm{T}} = [0_n \mid -d^{\mathrm{T}}],$$

$$h_{d}^{\mathrm{T}} = [0_n\ 0_n \mid -d^{\mathrm{T}}],$$

$$H_{\mathbf{y}} := \begin{bmatrix} h_{y_0} & 0 & \dots & \dots & 0 \\ 0 & h_y & & & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \dots & h_y & 0 \\ 0 & 0 & \dots & 0 & h_{y_N} \end{bmatrix}, \; \mathbf{d} := \begin{bmatrix} h_{d_0} \\ h_d \\ \vdots \\ h_d \\ h_{d_N} \end{bmatrix},$$

$$H_{\mathbf{z}} := \left[\begin{array}{cccc|cccc} I_n & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & I_p & 0 & \dots & 0 \\ \hline I_n & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ 0 & I_n & \dots & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & I_p & \dots & 0 \\ \hline \vdots & & \ddots & & \vdots & & \ddots & 0 \\ \hline 0 & 0 & \dots & I_n & 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & I_p \end{array}\right],$$

Problem (4.3) can be rewritten as follows:

$$\text{minimize } f(\mathbf{y}) + g(\mathbf{z}) \tag{4.33a}$$

$$H_{\mathbf{y}}\mathbf{y} + H_{\mathbf{z}}\mathbf{z} = \mathbf{d}, \tag{4.33b}$$

According to the definition of $\mathcal{Q}$, $f(\mathbf{y})$ is strongly convex, has a convexity parameter $\sigma_f := \text{eig}_{\min}(\mathcal{Q}) = \text{eig}_{\min}(\text{blockdiag}\{Q, R\}) = \sigma_{f_t}$, and a Lipschitz constant $L_f := \text{eig}_{\max}(\mathcal{Q})$. In addition, $g(\mathbf{z})$ is a convex function.

For the proposed splitting, Assumptions 4.3.2 and 4.3.3 are satisfied. Concerning Assumption 4.3.1, note that $L(\nabla F) := \text{eig}_{\max}(H_{\mathbf{y}})\sigma_f^{-1} = \max_t(\text{eig}_{\max}(H_{y_t})\sigma_{f_t}^{-1}) = \max_t(L_t(\nabla F_t)) = L_t(\nabla F_t) \leq \sum_{t=0}^{N} L_t(\nabla F_t)$, where the last equality follows from the fact that we deal with LTI systems ($L_0 = L_1 = \ldots = L_N$). Hence, on the dual, Assumption 4.3.5 still holds and, consequently, we can use SVR-AMA to solve Problem (4.3).

The associated SVR-AMA algorithm to solve Problem (4.33) is detailed in Algorithm 4.8[2]. In particular, defining $\boldsymbol{\mu}^{\mathrm{T}} := [v_1^{\mathrm{T}}\, \lambda_0^{\mathrm{T}} \mid w_1^{\mathrm{T}} \,\ldots\, w_{N-1}^{\mathrm{T}}\, v_N^{\mathrm{T}}\, \lambda_{N-1}^{\mathrm{T}} \mid w_N^{\mathrm{T}}\, \lambda_N^{\mathrm{T}}]$, according to the partitioning of $H_{\mathbf{y}}$ and $H_{\mathbf{z}}$, $F(\boldsymbol{\mu}) = f^\star(H_{\mathbf{y}}^{\mathrm{T}}\boldsymbol{\mu})$. Furthermore, $\nabla F(\mathbf{w})$ is the gradient of $F$ at $\mathbf{w}$, $\nabla F(\mathbf{v})$ is the gradient of $F$ at $\mathbf{v}$, and $\nabla F(\boldsymbol{\lambda})$ is the gradient of $F$ at $\boldsymbol{\lambda}$. Note that the calculation of the gradient step for this particular splitting is very simple and requires the evaluation of the product $H_{\mathbf{y}}\mathbf{y}$, which can be performed efficiently by exploiting the structure of the matrix $H_{\mathbf{y}}$. Finally, note that, given the structure of $F(\boldsymbol{\mu})$ the probability $\pi_t$ does not affect the choice of the step size $\eta$, according to Remark 4.3.2.

The following complexity upper bound on the primal sequence can be defined:

**Theorem 4.6.1.** *Consider Problem* (4.33)*. Let* $\{\mathbf{y}^k\}$ *and* $\{\boldsymbol{\mu}^k\}$ *be the sequence of primal and dual variables, respectively, generated by Algorithm 4.8. If Assump-*

[2]The accelerated versions are omitted here due to space limitations, but can be easily derived from Algorithms 4.6 and 4.7, respectively.

**Algorithm 4.8** SVR-AMA for Problem (4.33).

Given $\tilde{\mu}^0$, $N$, $\bar{s}$, $\mathcal{I}_N := \{0, \ldots, N\}$, $L^\star := (\sigma_f)^{-1} \operatorname{eig}_{\max}(H_y)$, $\eta$, and $T$.
**while** $s \leq \bar{s}$ **do**
0a. Set $\tilde{\mathbf{w}} = \tilde{\mathbf{w}}^{s-1}$, $\tilde{\mathbf{v}} = \tilde{\mathbf{v}}^{s-1}$,
$\tilde{\boldsymbol{\lambda}} = \tilde{\boldsymbol{\lambda}}^{s-1}$, and $\tilde{\boldsymbol{y}} = \tilde{\boldsymbol{y}}^{s-1}$.
0b. Set $\tilde{\beta}_{\mathbf{w}} = \nabla F(\tilde{\mathbf{w}})$, $\tilde{\beta}_{\mathbf{v}} = \nabla F(\tilde{\mathbf{v}})$, and
$\tilde{\beta}_{\boldsymbol{\lambda}} = \nabla F(\tilde{\boldsymbol{\lambda}})$.
0c. Set $\mathbf{w}^0 = \tilde{\mathbf{w}}$, $\mathbf{v}^0 = \tilde{\mathbf{v}}$, and $\boldsymbol{\lambda}^0 = \tilde{\boldsymbol{\lambda}}$.
0d. Set $\Pi := \{\pi_0, \ldots, \pi_N\}$ on $\mathcal{I}_N$
**for** $k = 1, \ldots, T$ **do**
1. Pick $i \in \mathcal{I}_N$ randomly according to $\Pi$.
2a. $y_i^k = \operatorname{argmin}_y \; f_i(y_i) + \langle w_i, H_1 y_i \rangle +$
$\langle v_{i+1}, H_2 y_i \rangle + \langle \lambda_i, -G y_i \rangle$.
2.b $\sigma_i^k = \mathbf{Pr}_{\mathbb{C}}(G y_i^k - d - \eta \lambda_i)$.
2c. $z_i^k = \frac{1}{2}\left[H_1 y_i^k + H_2 y_{i-1} - \eta(w_i + v_i)\right]$.
3a. $\beta_{w_i}^k = \tilde{\beta}_{w_i} + \frac{(y_i^k - \tilde{y}_i)^{\mathrm{T}} H_1^{\mathrm{T}}}{\pi_i}$.
3b. $\beta_{v_i}^k = \tilde{\beta}_{v_i} + \frac{(y_{i-1} - \tilde{y}_{i-1})^{\mathrm{T}} H_2^{\mathrm{T}}}{\pi_i}$.
3c. $\beta_{\lambda_i}^k = \tilde{\beta}_{\lambda_i} - \frac{(y_i^k - \tilde{y}_i)^{\mathrm{T}} G^{\mathrm{T}}}{\pi_i}$.
4a. $w_i^k = w_i + \eta\left(z_i^k - \beta_{w_i}^k\right)$.
4b. $v_i^k = v_i + \eta\left(z_i^k - \beta_{v_i}^k\right)$.
4c. $\lambda_i^k = \lambda_i + \eta\left(\beta_{\lambda_i}^k + d - \sigma_i^k\right)$.
**end for**
5. $\tilde{\mathbf{w}}^s = \frac{1}{T}\sum_{k=1}^T \mathbf{w}^k$, $\tilde{\mathbf{v}}^s = \frac{1}{T}\sum_{k=1}^T \mathbf{v}^k$,
$\tilde{\boldsymbol{\lambda}}^s = \frac{1}{T}\sum_{k=1}^T \boldsymbol{\lambda}^k$, and $\tilde{\boldsymbol{y}}^s = \frac{1}{T}\sum_{k=1}^T \boldsymbol{y}^k$ .
**end while**

*tions 4.3.1-4.3.3 are satisfied, given $\tilde{\boldsymbol{\mu}}^0 \in \operatorname{dom}(G)$, where $G := g^\star(H_{\boldsymbol{z}}^T \boldsymbol{\mu}) - \boldsymbol{d}^T \boldsymbol{\mu}$, then, the following holds:*

$$\mathbb{E}\|\tilde{\boldsymbol{y}}^s - \boldsymbol{y}^*\|^2 \leq \frac{2}{\sigma_f}(D(\boldsymbol{\mu}^*) - \mathbb{E}D(\tilde{\boldsymbol{\mu}}^0)). \tag{4.34}$$

*Proof.* The inequality can be derived by the results of Theorem 5.3 in [96] by noticing that the primal updates in the inner loop are the same as AMA. Then, we have to take into account for Algorithm 4.8 that the primal variables are stochastic variables and that we must consider their expected values. These observations combined with the results of Theorem 4.5.1 lead to (4.34). □

*Remark* 4.6.1. The initial value of the dual variables $\tilde{\boldsymbol{\mu}}^0$ should be a feasible starting point in order to use the results of Theorem 5.3. This can be accomplished by noticing the following. Concerning the $\tilde{\lambda}_t^0$ components of $\tilde{\boldsymbol{\mu}}^0$, they must be in $\mathbb{C}_t$.

Concerning the $\tilde{w}_t^0$ and $\tilde{v}_t^0$ components of $\tilde{\boldsymbol{\mu}}^0$, by providing an initial primal solution satisfying the consensus constraints (e.g., by using the evolution of the state starting from $x_{\text{init}}$ under the associated unconstrained LQR control law $u_t = K_{\text{LQR}} x_t$), they can be set equal to zero.

The *decomposition along the length of the prediction horizon* offers several advantages. First, the size of the subproblems (4.32) is fixed and independent from the length of the prediction horizon. Second, the resulting subproblems have improved numerical properties (in terms of condition number, for example), compared to solving Problem (4.3). Third, this decomposition allows one to fully parallelize the solution of Problem (4.3), thanks to the introduction of the consensus variables. In theory, if $N + 1$ independent workers are available, the dual update of each subproblem can be assigned to its dedicated worker that exchanges information with its neighbors only at dedicated synchronization points to update the consensus variables, as detailed in [121]. If the prediction horizon, however, is larger than the number of available workers the computation of the solution has to be partially (or fully, if only one worker is available) serialized. This scenario can be quite common for embedded legacy control systems, where the *serial* hardware architecture is formally verified and the costs to upgrade to the *parallel* one are too high. In this scenario, Algorithm 4.5 plays a fundamental role to compute a suboptimal solution of Problem (4.3).

Algorithm 4.5 applied to Problem (4.31) translates into the possibility of *asynchronous updates* of the independent subproblems. Compared to solving the subproblems in a serialized fashion (i.e., one after the other) in a synchronous framework, the asynchronous updates lead to less costly (in terms of computation time) iterations of the algorithm. In particular, assuming that only one worker is available, at each inner-loop iteration (steps 1-4 of the algorithm), only one subproblem is randomly selected for the update. In a synchronous framework, the update of all the subproblems would have been required, which can be costly if the length of the horizon is large.

Compared to other asynchronous dual algorithms (e.g., [85]), Algorithm 4.5 allows one to *tune and adapt (online) the probability distribution* $\Pi$. This is particularly useful, for example, to give priority in the update to those subproblems whose associated dual variables vary the most between two iterations of the algorithm, as shown in the next section.

## 4.7 Numerical Example

This section considers the linearized model (at a given trim condition) of an Airbus passenger aircraft [37] to test the proposed design. Aerospace applications offer several challenges for MPC from the computational perspective. First, these applications usually have strict real-time requirements. Second, the states have different magnitudes (ranging from few degrees to thousands of feet) affecting the condi-

tioning of the MPC problem. Third, the open-loop system has complex conjugate eigenvalues close to the imaginary axis. Finally, the problem size is relatively large and a long prediction horizon is required.

We focus on the longitudinal control of the aircraft. In this respect, the model we consider has $n = 6$ states (to describe the longitudinal dynamics) and $m = 4$ control actuators. In particular, the states associated with the longitudinal dynamics are pitch rate [deg/sec], roll rate [deg/sec], ground speed [knots], angle of attack [deg], pitch angle [deg], and altitude [ft]. Finally, the control surfaces for the longitudinal dynamics are the four elevators on the tail of the aircraft.

The sampling time of the system is $T_s = 0.04$ sec and we consider an horizon length $N = 60$. The total number of decision variables is 600. Furthermore, we have 3000 inequality constraints and 600 equality constraints. The goal of the MPC controller is to regulate the state of the system to the origin starting from a nonzero initial condition close to the saturation limits of the system.

We compared the behavior of Algorithms 4.5 (SVR-AMA), 4.6 (SVR-AMA with inner-loop acceleration), and 4.7 (SVR-AMA with outer-loop acceleration) with Algorithm 4.1 (AMA). The baseline for the comparison is the trajectory obtained using the MPC functions of MPT3 [47].

First, we are interested in showing that the possibility of tuning the probability distribution that the algorithm offers can lead to improvements in terms of performance of the MPC controller, especially when the solver runs for a limited number of iterations to reach a medium accuracy. In this respect, we consider three different probability distributions (depicted in Figures 4.1-4.2): (i) uniform, (ii) generalized Pareto, and (iii) adaptive. The adaptive distribution is computed online by the algorithm according to the following guidelines. We initialize $\Pi$ to be the Pareto distribution. Then, every $T$ inner-loop iterations, we check, for each $t = 0, \dots, N$ whether the following condition is verified $\|\tilde{\mu}^T - \tilde{\mu}^{T-1}\|^2 < 0.01$. If the condition is verified, $\pi_t \leftarrow 0.5\pi_t$ and the probabilities of its neighbors become $\pi_{t+1} \leftarrow \pi_{t+1} + 0.25\pi_t$ and $\pi_{t-1} \leftarrow \pi_{t-1} + 0.25\pi_t$. Figure 4.2 shows how the distribution varies when running Algorithm 4.5.

Second, we are interested in showing the benefits that the acceleration of the inner and outer loops can bring in terms of number of iterations needed to reach a suboptimal solution of the MPC problem (4.4). In this respect, we run the proposed algorithms (i.e., Algorithms 4.5, 4.6, and 4.7) and AMA for one problem instance that causes active inequality constraints at the optimum until the value of the dual variables between two iterates, in norm, does not vary more than $10^{-4}$ (i.e., $\|\tilde{\mu}_s - \tilde{\mu}_{s-1}\| \leq 10^{-4}$).

Figures 4.3-4.10[3] present the results obtained when testing the proposed algorithms. For the comparison, we fixed the total number of iterations for all the algorithms ($T\bar{s} = 15 \cdot 10^5$) and we use the same step size $\eta$. Concerning the tuning of the number of inner iterations, on one hand, we select $T = 500$ when using Algo-

[3]The units on the vertical axes of the presented plots have been removed upon request of our industrial partners.

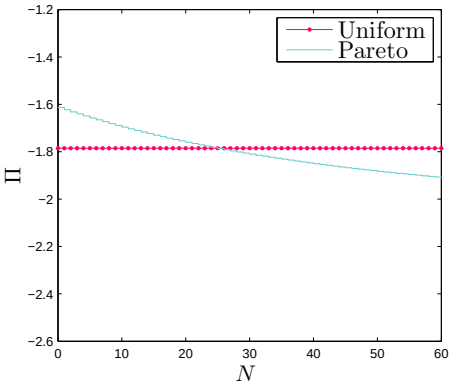


**Figure 4.1:** Probability distributions used to test the performance of Algorithms 4.5, 4.6, and 4.7 plotted in $\log_{10}$ scale.

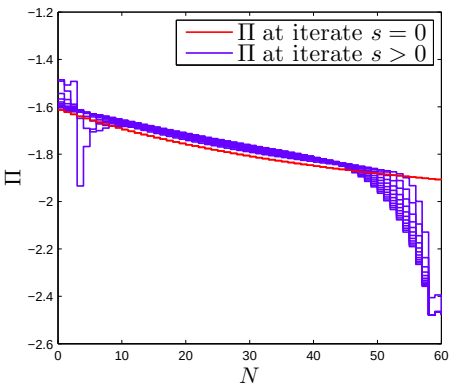


**Figure 4.2:** Adaptive probability generated when running Algorithm 4.5 (SVR-AMA) plotted in $\log_{10}$ scale.

rithm 4.6 (according to the guidelines provided to select the number of inner-loop iterations for IA-SVR-AMA), leading to a larger number of outer-loop iterations. On the other hand, we select $T = 3000$ when using Algorithm 4.7 (according to the guidelines provided to select the number of inner-loop iterations for OA-SVR-AMA), leading to a smaller number of outer-loop iterations (that requires the update of the full gradient). Concerning the tuning of $\eta$, we selected the value according to the guidelines that derive from the theory. Nevertheless, we notice that the proposed algorithms allow one to use a larger step size compared to AMA, potentially leading to better performance when we need to further reduce the number of iterations (and the computation time). Figures 4.3 and 4.4 show the behavior of one of the actuators and one of the states (the pitch rate, which is also interesting for the tracking of the vertical load factor to regulate the trajectory of the aircraft) when using Algorithm 4.5. The behavior of SVR-AMA is compared with AMA and the optimal trajectory. Hence, we notice that, within the limited number of iterations, the possibility to tune $\Pi$ leads to improvements in terms of quality of the solution. In particular, notice that the uniform distribution leads to a behavior comparable to AMA, while using the Pareto and the adaptive distribution lead to improved trajectories.

Figures 4.5 and 4.6 show the behavior of one of the actuators and one of the states (the pitch rate) when using Algorithm 4.6 to accelerate the inner-loop iterates. The behavior of the inner-accelerated algorithm is compared with SVR-AMA (with adaptive distribution) and AMA. Hence, we notice that the algorithm leads to some improvements in the calculation of the optimal solution of the MPC problem. The main issue is that the algorithm requires more updates of the full gradient of $F$, which can be problematic when $N$ is large. It is, however, interesting to notice that

the combined use of variance reduction (which is also an acceleration strategy from the stochastic point of view) and inner-loop acceleration (which is an acceleration strategy in a more classical sense) still lead to some benefits in the computation of the optimal solution.

Figures 4.7 and 4.8 show the behavior of one of the actuators and one of the states (the pitch rate) when using Algorithm 4.7 to accelerate the outer-loop iterates. As in the previous case, the behavior of the outer-accelerated algorithm is compared with SVR-AMA (with adaptive distribution) and AMA. We notice that the solution returned by OA-SVR-AMA is closer to the optimal one. Furthermore, we can also observe the benefits of tuning $\Pi$ online. In particular, note that the Pareto and the adaptive distributions clearly outperform the uniform distribution.

Finally, Figures 4.9 and 4.10 directly compare the *best* results obtained using SVR-AMA, IA-SVR-AMA, and OA-SVR-AMA. In particular, the plots highlight the significant improvements obtained using the outer-loop acceleration. Furthermore, note that Algorithm 4.7 requires less full gradient updates and fully exploits the benefits of the variance reduction scheme in the inner loop. Hence, it can be more efficient when used for applications with large $N$, compared to Algorithm 4.6.



## 4.8 Conclusions

We presented an asynchronous alternating minimization algorithm with variance reduction (SVR-AMA) scheme and its accelerated versions suitable for model predictive control (MPC) applications. As our numerical example showed, the proposed algorithms, compared to a state-of-the art solver (i.e., the alternating minimization algorithm), provide higher-accuracy solutions within the same number of overall iterations. Furthermore, compared to other state-of-the-art asynchronous dual solvers that only perform random updates according to a uniform distribution, the proposed algorithms allow one to prioritize the update of the variables at the beginning of the prediction horizon, leading to improved behavior in closed loop, as our numerical example showed.

We analyzed the possibility of tuning the probability distribution to improve the performance of the algorithm in terms of number of iterations. As part of our future work, we plan to further investigate the benefits that the proposed algorithm (SVR-AMA and its accelerated versions) can have in a distributed framework. In particular, we plan to investigate how to use the probability distribution as a tuning parameter to plan the communications among the agents in the network.

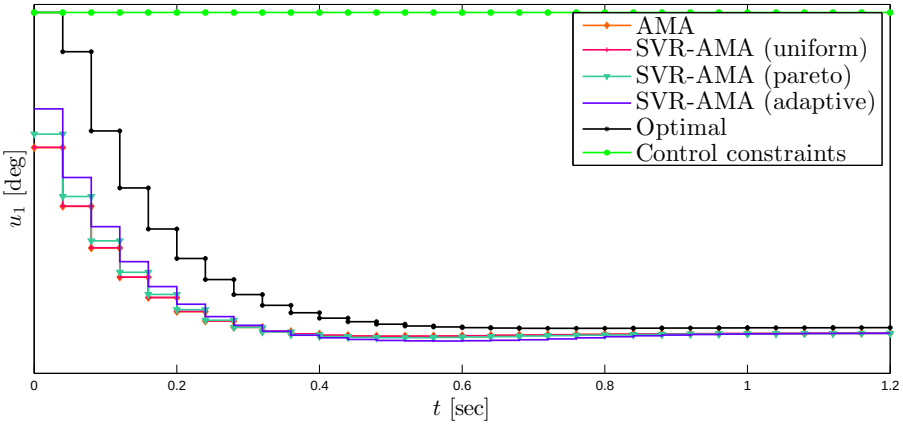


**Figure 4.3:** Control trajectories obtained using different probability distributions $\Pi$ in Algorithm 4.5 in open loop.

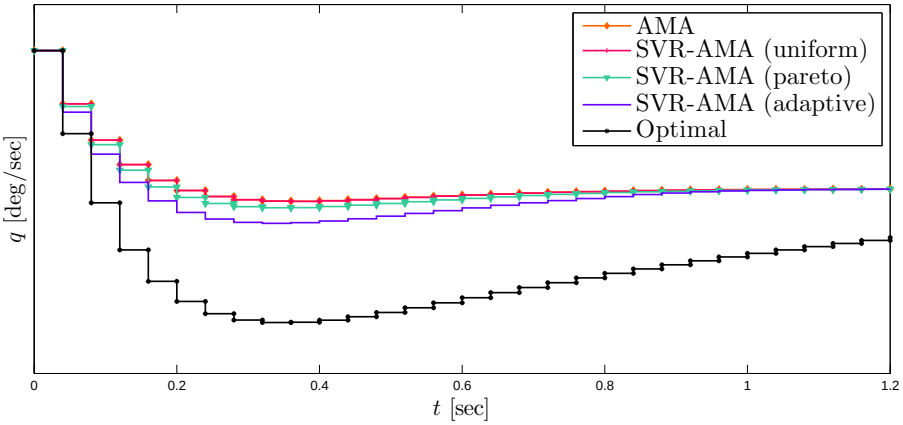


**Figure 4.4:** Pitch-rate trajectories obtained using different probability distributions $\Pi$ in Algorithm 4.5 in open loop.

# A Proofs of Section 4.4.1

## A.1 Proof of Proposition 4.4.1

*Proof.* We exploit the equivalence between Prox-SVRG and I-PGM. In particular, the following holds:

$$y_k = \mathrm{prox}_{\eta G}\left(y_{k-1} - \eta\beta_k\right) \tag{4.35a}$$

$$= \mathrm{prox}_{\eta G}\left(y_{k-1} - \eta\left(\nabla F(y_{k-1}) + e_k\right)\right), \tag{4.35b}$$

where $e_k = \beta_k - \nabla F(y_{k-1})$. Then, the proof follows directly from Proposition 4 in [116] by taking into account that the proximal step is computed exactly and the

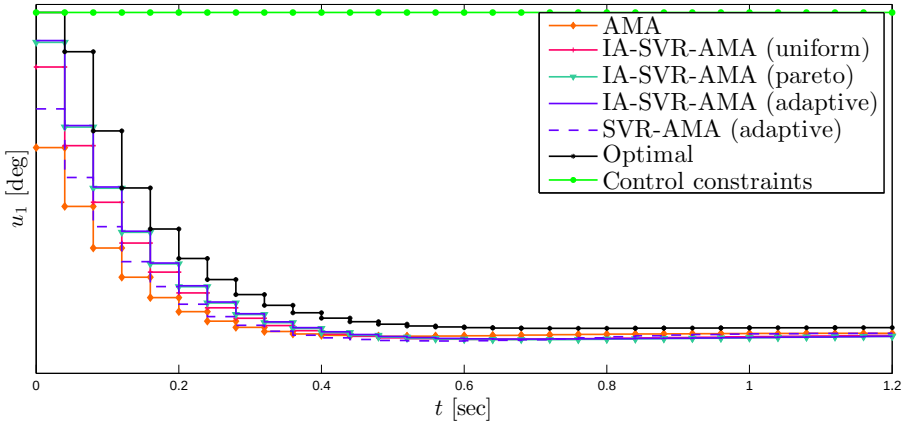


**Figure 4.5:** Control trajectories obtained using different probability distributions $\Pi$ in Algorithm 4.6 in open loop.

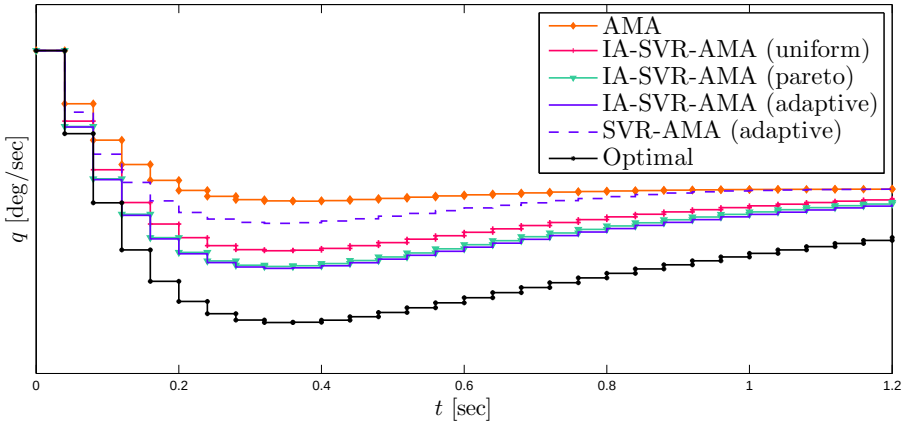


**Figure 4.6:** Pitch-rate trajectories obtained using different probability distributions $\Pi$ in Algorithm 4.6 in open loop.



only error we take into account is the one in the gradient calculations, which is a stochastic error (hence the expectation in the definition of $\Gamma_k$). Furthermore, we take into account that each $y_k$ is a stochastic variable and, consequently, we can consider the expectation in the value of $P(y_k)$. □

## A.2 Proof of Theorem 4.4.1

*Proof.* According to Proposition 4.4.1, the following holds:

$$\begin{aligned} &\mathbb{E}P(y_k) - P(y_*) \leq \\ &2(1-\gamma)^k(P(\tilde{y}_{s-1}) - P(y_*)) + \frac{2}{\sigma_F}(1-\gamma)^k\Gamma_k^2(\tilde{y}_{s-1}), \end{aligned} \tag{4.36}$$

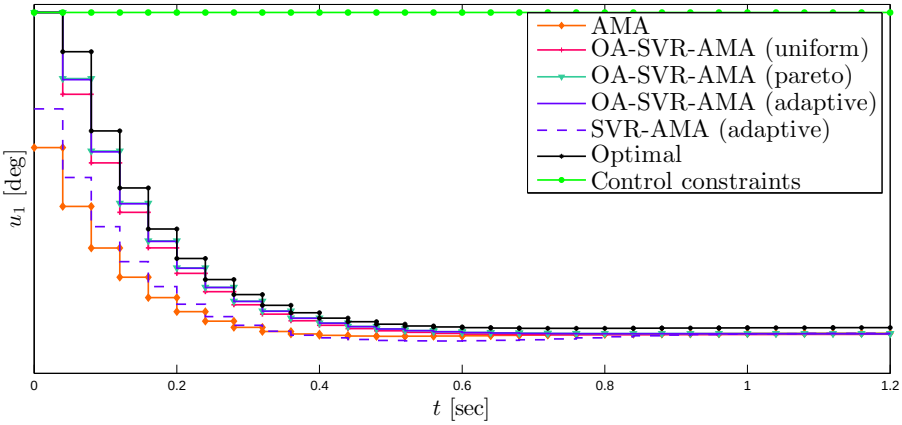


**Figure 4.7:** Control trajectories obtained using different probability distributions $\Pi$ in Algorithm 4.7 in open loop.

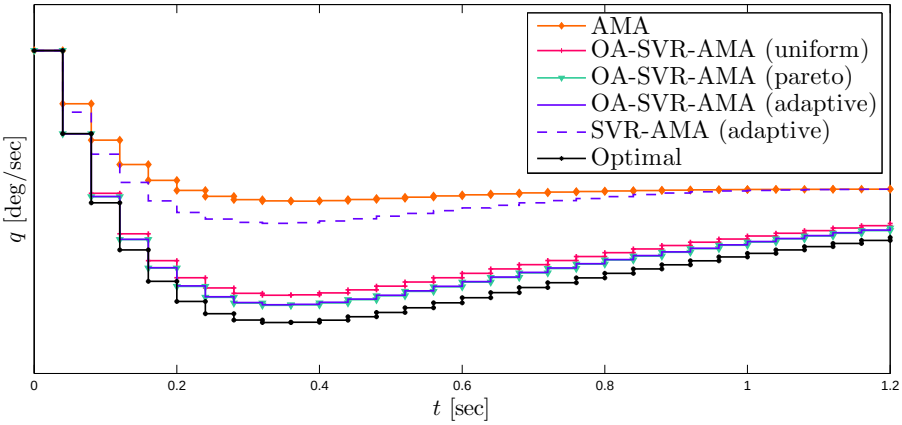


**Figure 4.8:** Pitch-rate trajectories obtained using different probability distributions $\Pi$ in Algorithm 4.7 in open loop.

We are interested in the convergence of the outer loop of the algorithm. Hence, we sum for $k = 1, \ldots, T$ the inequality above:

$$\sum_{k=1}^{T} \left[\mathbb{E}P(y_k) - P(y_*)\right] \leq \sum_{k=1}^{T} \left[2(1-\gamma)^k \left(P(\tilde{y}_{s-1}) - P(y_*)\right)\right] + \frac{2}{\sigma_F} \sum_{k=1}^{T} (1-\gamma)^k \Gamma_k^2(\tilde{y}_{s-1}).$$

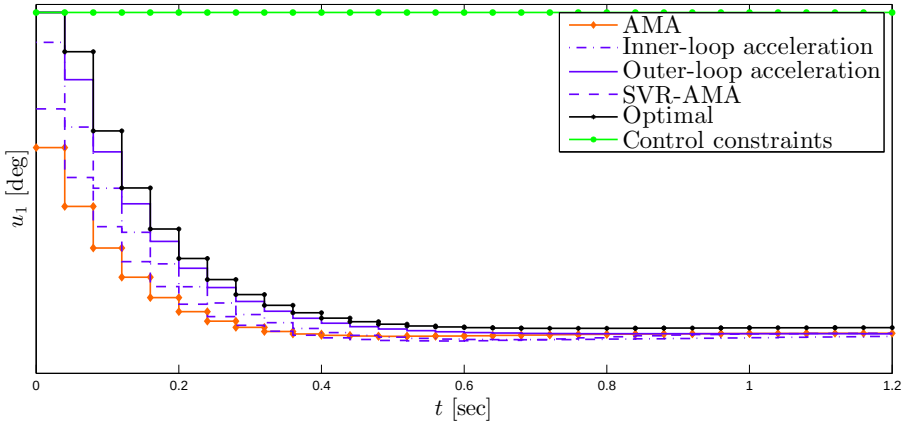


**Figure 4.9:** Comparison of the *best* control trajectory computed by SVR-AMA, IA-SVR-AMA, and OA-SVR-AMA in open loop.

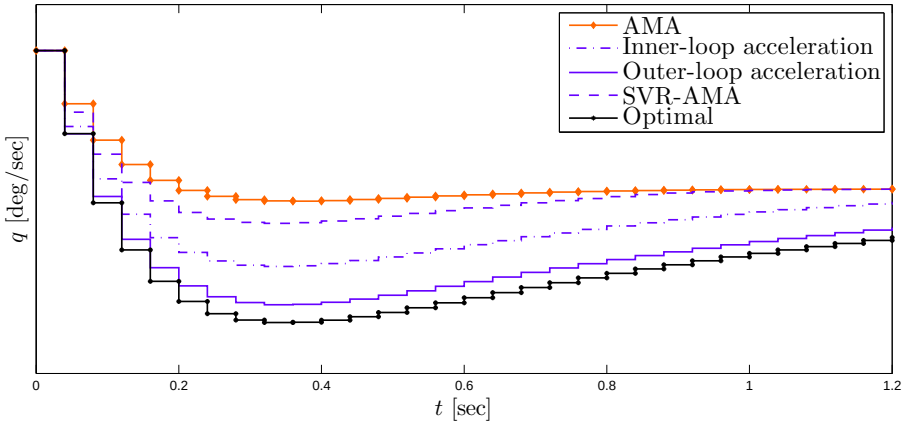


**Figure 4.10:** Comparison of the *best* pitch-rate trajectory computed by SVR-AMA, IA-SVR-AMA, and OA-SVR-AMA in open loop.



Then, the following holds:

$$\sum_{k=1}^{T} \mathbb{E}P(y_k) - TP(y_*) \leq \sum_{k=1}^{T} \left[2(1-\gamma)^k\right] \left[P(\tilde{y}_{s-1}) - P(y_*)\right] + \frac{2}{\sigma_F} \sum_{k=1}^{T} (1-\gamma)^k \Gamma_k^2(\tilde{y}_{s-1}).$$

Using $\sum \mathbb{E}[\cdot] = \mathbb{E}\sum[\cdot]$ and dividing by $T > 0$, the following holds:

$$\mathbb{E}\left(\frac{1}{T}\sum_{k=1}^{T} P(y_k)\right) - P(y_*) \leq \frac{2}{T}\sum_{k=1}^{T}\left[(1-\gamma)^k\right]\left[P(\tilde{y}_{s-1}) - P(y_*)\right] + \frac{2}{T\sigma_F}\sum_{k=1}^{T}(1-\gamma)^k\Gamma_k^2(\tilde{y}_{s-1})$$

Exploiting the convexity of $P$ and the fact that $1-\gamma < 1$, we have:

$$\begin{aligned}&\mathbb{E}P(\tilde{y}_s) - P(y_*)\\ \leq&\frac{2}{T\gamma}\left((1-\gamma) - (1-\gamma)^{T+1}\right)\left[P(\tilde{y}_{s-1}) - P(y_*)\right] +\\ &\frac{2}{T\sigma_F}\sum_{k=1}^{T}(1-\gamma)^k\Gamma_k^2(\tilde{y}_{s-1})\end{aligned}$$

The following holds:

$$\mathbb{E}P(\tilde{y}_s) - P(y_*) \leq \rho_{\text{IA,I-PGM}}\left[P(\tilde{y}_{s-1}) - P(y_*)\right] + \frac{2}{T\sigma_F}\Gamma(\tilde{y}_{s-1})$$

Applying recursively for $s = 1, 2, \ldots, \bar{s}$

$$\mathbb{E}P(\tilde{y}_{\bar{s}}) - P(y_*) \leq \rho_{\text{IA,I-PGM}}^{\bar{s}}\left[P(\tilde{y}_0) - P(y_*)\right] + \frac{2}{T\sigma_F}\sum_{s=0}^{\bar{s}-1}\rho_{\text{IA,I-PGM}}^{s}\Gamma(\tilde{y}_s),$$

which proves the theorem. □

# B Proofs of Section 4.4.2

## B.1 Proof of Corollary 4.4.2

*Proof.* The proof follows the same steps of the one of Corollary 3.5 in [136] but takes into account that we are considering $\beta_s$ instead of $\beta_k$. In particular, the following

holds:

$$\begin{aligned}\mathbb{E}\|e_s\|^2 &= \mathbb{E}\left\|\frac{1}{T}\sum_{k=1}^{T}\left(\beta_k - \nabla F(y_{k-1})\right)\right\|^2 \\ &\leq \frac{1}{T^2}\mathbb{E}\sum_{k=1}^{T}\|\beta_k - \nabla F(y_{k-1})\|^2 \\ &\leq \frac{1}{T^2}\mathbb{E}\sum_{k=1}^{T}\|v_k\|^2,\end{aligned}$$

where $v_k := \frac{\nabla F_{i_k}(y_{k-1}) - \nabla F_{i_k}(\tilde{y})}{\pi_{i_k}} + \nabla F(\tilde{y}) - \nabla F(y_{k-1})$. Using the fact that $\sum_i \mathbb{E}[x_i] = \mathbb{E}\sum_i x_i$ and $\mathbb{E}\|x - \mathbb{E}x\|^2 = \mathbb{E}\|x\|^2 - \|\mathbb{E}x\|^2$ according to [136] the following holds:

$$\begin{aligned}&\mathbb{E}\|e_s\|^2 \\ &\leq \frac{1}{T^2}\mathbb{E}\sum_{k=1}^{T}\|v_k\|^2 \\ &= \frac{1}{T^2}\sum_{k=1}^{T}\mathbb{E}\left\|\frac{\nabla F_{i_k}(y_{k-1}) - \nabla F_{i_k}(\tilde{y})}{\pi_{i_k}}\right\|^2 - \\ &\quad \frac{1}{T^2}\sum_{k=1}^{T}\|\nabla F(y_{k-1}) - \nabla F(\tilde{y})\|^2 \\ &\leq \frac{1}{T^2}\sum_{k=1}^{T}\sum_{i=1}^{N}\frac{1}{N\pi_i}\|\nabla F_i(y_{k-1}) - \nabla F_i(\tilde{y})\|^2 \\ &\leq \frac{1}{T^2}\sum_{k=1}^{T}\sum_{i=1}^{N}\frac{2L_i}{N\pi_i}\left(P_i(y_{k-1}) - P_i(y_*) + P_i(\tilde{y}) - P_i(y_*)\right) \\ &\leq \frac{2L_\Pi}{T^2}\sum_{k=1}^{T}\left(P(y_{k-1}) - P(y_*) + P(\tilde{y}) - P(y_*)\right) \\ &\leq \frac{2L_\Pi}{T}\left(P(y_{s-1,\text{avg}}) - P(y_*) + P(\tilde{y}) - P(y_*)\right).\end{aligned}$$

□

## B.2 Proof of Theorem 4.4.2

*Proof.* In the following, we provide a proof for the convergence of Algorithm 4.7.

Given that the inner loop of Algorithm 4.4 is not affected by the acceleration the following result from Theorem 4.3.1 in [136] holds:

$$\mathbb{E}P(\tilde{y}_s) - P(y_*) \leq \rho\left(P(\hat{y}_{s-1}) - P(y_*)\right). \tag{4.37}$$

Consider for simplicity $\alpha := \frac{1-\sqrt{\gamma}}{1+\sqrt{\gamma}}$. Then, define the following quantity (according to [116, Section 6.4]):

$$v_s := \left(1 - \frac{1}{\sqrt{\gamma}}\right)\tilde{y}_{s-1} + \frac{1}{\sqrt{\gamma}}\tilde{y}_s, \tag{4.38}$$

and

$$\theta = \frac{\sqrt{\gamma} - \gamma}{1 - \gamma} < 1. \tag{4.39}$$

Hence, the following holds:

$$\hat{y}_s = (1-\theta)\tilde{y}_s + \theta v_s. \tag{4.40}$$

Given that $\sqrt{\gamma} < 1$, by using the convexity of $P$ and the definition of $v_s$ in (4.38), the following holds:

$$\begin{aligned}\mathbb{E}P(\tilde{y}_s) - P(y_*) \leq &(1 - \sqrt{\gamma})(P(\tilde{y}_{s-1}) - P(y_*)) + \\ &\sqrt{\gamma}(\mathbb{E}P(v_s) - P(y_*)).\end{aligned} \tag{4.41}$$

At the same time, by using (4.37) and the definition of $\hat{y}_s$ in(4.40), the following holds:

$$\begin{aligned}\mathbb{E}P(\tilde{y}_{s+1}) - P(y_*) \leq &\rho(1-\theta)(P(\tilde{y}_s) - P(y_*)) \\ &+ \rho\theta(P(v_s) - P(y_*)).\end{aligned} \tag{4.42}$$

From (4.42), the following holds:

$$\mathbb{E}P(\tilde{y}_s) - P(y_*) \geq \frac{\mathbb{E}P(\tilde{y}_{s+1}) - P(y_*) - \rho\theta(\mathbb{E}P(v_s) - P(y_*))}{\rho(1-\theta)} \tag{4.43}$$

At the same time, using (4.41), the following holds

$$\begin{aligned}&(\rho(1-\theta))(1 - \sqrt{\gamma})(P(\tilde{y}_{s-1}) - P(y_*)) + \\ &(\rho(1-\theta))\sqrt{\gamma}(\mathbb{E}P(v_s) - P(y_*)) \\ &\geq \rho(1-\theta)(\mathbb{E}P(\tilde{y}_s) - P(y_*)) \\ &\geq (\mathbb{E}P(\tilde{y}_{s+1}) - P(y_*)) - \rho\theta(\mathbb{E}P(v_s) - P(y_*)).\end{aligned}$$

Then, the following holds:

$$\begin{aligned}&(\rho(1-\theta))(1 - \sqrt{\gamma})(P(\tilde{y}_{s-1}) - P(y_*)) + \\ &((\rho(1-\theta))\sqrt{\gamma} + \rho\theta)(\mathbb{E}P(v_s) - P(y_*)) \\ &\geq \rho(1-\theta)(\mathbb{E}P(\tilde{y}_s) - P(y_*)) + \rho\theta(\mathbb{E}P(v_s) - P(y_*)) \\ &\geq (\mathbb{E}P(\tilde{y}_{s+1}) - P(y_*)).\end{aligned} \tag{4.44}$$

Given the fact that $\rho\theta(\mathbb{E}P(v_s) - P(y_*)) \leq ((\rho(1-\theta))\sqrt{\gamma} + \rho\theta)(\mathbb{E}P(v_s) - P(y_*))$ and, at the same time, (4.44) holds, the following must hold for all $s > 0$

$$(\mathbb{E}P(\tilde{y}_s) - P(y_*)) \leq (1 - \sqrt{\gamma})(P(\tilde{y}_{s-1}) - P(y_*)). \tag{4.45}$$

Hence, if we apply the inequality above recursively, the following holds:

$$\mathbb{E}P(\tilde{y}_s) - P(y_*) \leq (1 - \sqrt{\gamma})^s(P(\tilde{y}_0) - P(y_*)). \tag{4.46}$$

□

# C Proofs of Section 4.5

## C.1 Proof of Theorem 4.5.1

*Proof.* If the assumptions of the theorem are satisfied, exploiting a similar argument to the one in [35, Theorem 4] for AMA, we can show that Algorithm 4.5 is equivalent to apply Algorithm 4.2 to Problem (4.5), that is, the dual of Problem (4.4). In particular, according to the proof of Theorem 4, using the optimality conditions that derive from steps (2a) and (2b), we can show that steps (2) and (4) are equivalent to computing $\operatorname{prox}_{\eta G}(\mu_k - \eta\beta_k)$, which is equivalent to the proximal step (3) of Algorithm 4.2. The definition of $\beta_k$ in step (3) of Algorithm 4.5 follows the same logic of Prox-SVRG on the dual. In particular, by defining $\tilde{\beta} = \nabla F(\tilde{\mu})$, we replace the calculation of the full gradient of $F(\mu)$ with the following:

$$\beta_k = \tilde{\beta} + \frac{\nabla F^{(i)}(\mu_{k-1}) - \nabla F^{(i)}(\tilde{\mu})}{\pi^{(i)}} \tag{4.47a}$$

$$= \tilde{\beta} + \frac{(y_k - \tilde{y})^{\mathrm{T}} H_y^{(i)^{\mathrm{T}}}}{\pi^{(i)}} \tag{4.47b}$$

Hence, we can conclude that Algorithm 4.5 is equivalent to Algorithm 4.2 applied to the dual of Problem (4.4). Then, we can use Lemmas 4.3.1 and 4.3.2 to derive the upper bound on $\eta$ and to satisfy the assumptions of Theorem 4.3.1. Finally, by using Lemma 4.5.1 we can derive the results of the theorem by following the proof of Theorem 4.3.1 (applied to the dual). □

## C.2 Proof of Corollary 4.5.1

*Proof.* We first prove (4.29) by using the same logic in the proof of Corollary 3.5 in [136]. In particular, the following holds:

$$\begin{aligned}
\mathbb{E}\beta_k &= \mathbb{E}\nabla F(\tilde{\mu}) + \mathbb{E}\frac{\nabla F^{(i)}(\mu_{k-1})}{\pi^{(i)}} - \mathbb{E}\frac{\nabla F^{(i)}(\tilde{\mu})}{\pi^{(i)}} \\
&= \nabla F(\tilde{\mu}) + \sum_{i=0}^{N} \frac{\pi^{(i)}\nabla F^{(i)}(\mu_{k-1})}{\pi^{(i)}} - \sum_{i=0}^{N} \frac{\pi^{(i)}\nabla F^{(i)}(\tilde{\mu})}{\pi^{(i)}} \\
&= \nabla F(\tilde{\mu}) + \sum_{i=0}^{N} \nabla F^{(i)}(\mu_{k-1}) - \sum_{i=0}^{N} \nabla F^{(i)}(\tilde{\mu}) \\
&= \nabla F(\tilde{\mu}) + \nabla F(\mu_{k-1}) - \nabla F(\tilde{\mu}) \\
&= \nabla F(\mu_{k-1}).
\end{aligned}$$

Then, the proof of (4.30) follows exactly the proof in [136] with the only difference that in the last step we use the definition of $\nabla F(\mu)$. In particular, the following

holds:

$$
\begin{aligned}
&\mathbb{E}(\beta_k - \nabla F(\mu_{k-1})) \\
&\leq 2\sum_{i=0}^{N} \frac{1}{\pi^{(i)}} \|\nabla F^{(i)}(\mu_{k-1}) - \nabla F^{(i)}(\mu_*)\|^2 + \\
&\quad 2\sum_{i=0}^{N} \frac{1}{\pi^{(i)}} \|\nabla F^{(i)}(\tilde{\mu}) - \nabla F^{(i)}(\mu_*)\|^2 \\
&\Downarrow \text{ using the definition of } \nabla F(\mu) \\
&= 2\sum_{i=0}^{N} \frac{1}{\pi^{(i)}} \|H_y^{(i)}(y_k - y_*)\|^2 + 2\sum_{i=0}^{N} \frac{1}{\pi^{(i)}} \|H_y^{(i)}(\tilde{y} - y_*)\|^2 \\
&\Downarrow \text{ using the Cauchy-Schwarz inequality} \\
&\leq 2\sum_{i=0}^{N} \frac{\|H_y^{(i)}\|^2}{\pi^{(i)}} (\|y_k - y_*\|^2 + \|\tilde{y} - y_*\|^2) \\
&\Downarrow \text{ using the fact that } \|y - y_*\|^2 \leq \frac{2}{\sigma_F}(P(y) - P(y_*)) \\
&\leq 4\sum_{i=0}^{N} \frac{\|H_y^{(i)}\|^2}{\sigma_F \pi^{(i)}} [P(y_k) - P(y_*) + P(\tilde{y}) - P(y_*)] \\
&\leq 4L_\pi^\star [P(y_k) - P(y_*) + P(\tilde{y}) - P(y_*)],
\end{aligned}
$$

where $L_\Pi^\star := \max_i(\mathrm{eig}_{\max}(H_y^{(i)^{\mathrm{T}}} H_y^{(i)})/(\sigma_F \pi^{(i)}))$. □

# 5

# Fault-Tolerant Reference Generation for Model Predictive Control with Active Diagnosis of Elevator Jamming Faults


## Abstract

This chapter focuses on the longitudinal control of an Airbus passenger aircraft in the presence of elevator jamming faults. In particular, in this chapter, we address permanent and temporary actuator jamming faults using a novel reconfigurable fault-tolerant predictive control design. Due to their different consequences on the available control authority and fault duration, the above two actuator jamming faults need to be distinguished so that appropriate control reconfigurations can be adopted accordingly. Their similarity in symptoms, however, prevents effective discrimination of the root cause of the jamming when using only a passive fault-diagnosis approach. Hence, we propose the use of model predictive control (MPC) as fault-tolerant controller to actively help the fault-detection (FD) unit discriminate between a permanent and a temporary jamming fault, while ensuring the performance of the aircraft. The MPC controller and FD unit closely interact during the detection and diagnosis phases. In particular, every time a fault is detected, the FD module commands the MPC controller to perform a predefined sequence of reconfigurations to diagnose the root cause of the fault. An artificial reference signal that accounts for changes in the actuator operative ranges is used to guide the system through this sequence of reconfigurations. Our strategy is demonstrated on an Airbus passenger aircraft simulator.

## 5.1 Introduction

The ability to automatically handle faults and component malfunctions while preserving overall performance is the main characteristic of a fault-tolerant control (FTC) system [144]. Fault-tolerant control systems have been largely investigated in the context of flight control taking into account the occurrence of faults on sensors and actuators [11; 15; 44; 110; 87; 93; 140; 142].

In this work, we focus on faults that can occur on the aircraft actuators (i.e., actuator jamming faults). Actuator jamming faults have long been investigated in the field of fault-tolerant flight control (e.g., [70; 15; 10; 140]). Among other techniques, we focus on the use of model predictive control (MPC) as fault-tolerant control. MPC provides a well-recognized framework for fault tolerance [1; 56; 70; 65]. On one hand, by relying on actuator redundancy, MPC (even) without reconfiguration has some inherent *self-reconfiguration* properties that allows one to reallocate the control effort in the presence of actuator faults [68]. On the other hand, reconfigurable MPC further improves fault tolerance capabilities by exploiting extra fault information in a structured manner, especially when it comes to dealing with constraints [68].

In practical applications, the control design has to take into account that the information concerning the fault is provided by a fault-detection (FD) module. Hence, in these scenarios, the design of a reconfigurable MPC controller must be integrated with an FD module. Robustness and guaranteed fault tolerance of this integrated fault-tolerant MPC (FTMPC) scheme was analyzed with set theoretic methods in [126; 139].

In most literature, actuator jamming is attributed to a permanent jamming (or *stuck fault*), during which the actuator is locked at a certain position. The study of temporary jamming due to dynamic manoeuvres (combined with the presence of heavy aerodynamic forces), however, has been only investigated by few researchers (e.g., the authors of [10] propose a sliding mode fault-tolerant control scheme to detect and compensate the effects of the temporary and permanent jamming faults). This temporary jamming—known as *stall load* or blow-down [36; 10] for aerospace applications—leads to more stringent control limits for a bounded period of time. The original limitations of the actuators can be recovered once either the control command is consequently adjusted or the aerodynamic forces become smaller [36]. Although both stuck fault and stall load lead to a jammed actuator, their consequences on the control limits and jamming duration are significantly different. Therefore, we must be able to identify the root cause of the actuator jamming (i.e., identify whether the actuator is temporarily or permanently jammed). Furthermore, in case of stall load, we must be able to determine its end to apply suitable reconfiguration strategies from the control design perspective.

Conventional FD cannot achieve this goal because the fault phenomena of a permanent or temporary jamming have a high similarity. We propose to integrate reconfigurable MPC with active FD to address the challenge above. Instead of passively

monitoring actuator behaviors, we exploit a sequence of reconfiguration strategies using the MPC controller to assist the FD module, not only to distinguish the root cause of the actuator jamming, but also to actively detect the end of a stall load (in case of a temporary jamming). Then, once the root cause of the jamming is detected, the MPC controller adopts suitable successive reconfigurations, aimed to improve the overall control performance. All these improvements from both FD and control perspective cannot be achieved without using active reconfigurations to assist FD.

The use of active FD in the context of FTMPC has been rather limited so far and focused only on permanent faults [98; 100; 137]. In contrast, our contribution lies in discriminating between a permanent and temporary jamming (i.e., stuck fault and stall load, respectively) that share highly similar fault symptoms. Compared to the approach we proposed in [23], we rely on *(i)* an improved FD strategy, *(ii)* a different MPC formulation for tracking, and *(iii)* a modified disturbance observer to incorporate plant-model mismatches. From the detection perspective, in [23] the FD unit relies only on the information from a single control surface, without exploiting the actuator redundancy. In this work, we combine the detection strategy in [23] with an additional check that compares the behavior of the single elevator with the others. This has the additional benefit that if only one (or two) control surfaces are subject to faults, the fault can be detected quickly by monitoring the deviation of the residual signal from the normal behavior of the others. This strategy is useful especially for permanent jamming faults that are more likely to involve only one control surface. Temporary faults that are more likely to affect all the control surfaces can still be detected by monitoring whether the residual signal of each actuator exceeds a predetermined threshold. From the control perspective, in [23] we made the assumption that the desired reference during a manoeuvre could not lead to infeasible solutions and all the control reconfigurations were performed on the actuator constraints directly, without affecting the desired reference signal. In contrast to [23], in this work we exploit a strategy similar to the artificial reference tracking proposed by [66; 19]. In [66; 19], the concept of artificial reference is used to enlarge the region of attraction of the proposed controller while ensuring closed-loop stability guarantees. We reinterpret this idea for fault-tolerant control purposes. In particular, this approach can be used to compute artificial reference signals for the state and the actuator commands in order to compensate for the occurrence of faults that can suddenly affect the feasible region of the MPC controller. In particular, the sequence of reconfigurations used to detect and diagnose the root cause of the jamming is not performed directly on the actuators' constraints, but on the constraints associated with the artificial reference signal. By doing so, when a fault is detected the reference followed by the states and the actuators is adapted to the faulty feasible region. Consequently, if the desired reference signal becomes unfeasible in the presence of a fault, the artificial reference acts as a *fault-tolerant* reference signal to avoid infeasibility (and possible instability) issues. Finally, compared to [23], we incorporate the effects of plant-model mismatches directly in the definition of the artificial reference constraints using the information provided by an improved disturbance estimator

module. We demonstrate the effectiveness of our approach using an Airbus civil aircraft simulator [37].

In the following, Section 5.2 presents the Airbus simulator used to evaluate our design. Section 5.3 describes our fault-tolerant control architecture. Section 5.4 introduces the proposed detection and diagnosis strategy and highlights the interactions between the FD module and the MPC controller. Section 5.5 compares the behavior of the MPC controller with and without the proposed active reconfigurations when multiple faults occur on the elevators. Finally, Section 5.6 concludes this chapter.

## 5.2 Benchmark Model and Scenario Definition

This section describes the RECONFIGURE benchmark model, that is, an Airbus civil aircraft simulator [37] (Section 5.2.1), and details the actuator fault scenarios we focus on in this work (Section 5.2.2).

### 5.2.1 The Aircraft Longitudinal Model

This work focuses on the longitudinal control of an Airbus passenger aircraft in the presence of actuator jamming faults. Our proposed FTC architecture relies on MPC, which is a model-based technique. Hence, a mathematical description of the longitudinal dynamics of the aircraft (i.e., the model) is necessary to ensure performance of our FTC scheme. In this respect, in the control design phase, we can rely on linearized aircraft models at given operating points (or *trim conditions*) to build the prediction model of the MPC controller. In the following, we describe the augmented aircraft model (i.e., the cascade actuator-aircraft dynamics depicted in Figure 5.3) and introduce the notation used to design our MPC control (Section 5.3).

The linearized and discretized longitudinal dynamics of the aircraft can be described as follows:

$$x_{\text{A/C}}(t+1) = A_{\text{A/C}}x(t) + B_{\text{A/C}}u_{\text{A/C}}(t) \tag{5.1a}$$

$$y_{\text{A/C}}(t) = C_{\text{A/C}}x(t) + D_{\text{A/C}}u_{\text{A/C}}(t), \tag{5.1b}$$

where $x_{\text{A/C}} := [q\, p\, v\, \alpha\, \vartheta\, h]^{\text{T}} \in \mathcal{X}_{\text{A/C}} \subseteq \mathbb{R}^{n_{\text{A/C}}}$ is the state vector, which includes the pitch rate, roll rate, ground speed, angle of attack, pitch angle, and altitude, respectively, $u_{\text{A/C}} := [\delta_{\text{e}_{\text{li}}}\, \delta_{\text{e}_{\text{ri}}}\, \delta_{\text{e}_{\text{lo}}}\, \delta_{\text{e}_{\text{ro}}}] \in \mathcal{U} \subseteq R^{n_{\text{u}}}$ is the control input with $\delta_{\text{e}_{\text{li}}}$, $\delta_{\text{e}_{\text{ri}}}$, $\delta_{\text{e}_{\text{lo}}}$, and $\delta_{\text{e}_{\text{ro}}}$ representing the left inner, right inner, left outer and right outer elevator deflections, respectively, and $y_{\text{A/C}} := [n_{\text{z}}\, x^{\text{T}}]^{\text{T}} \in \mathcal{Y}_{\text{A/C}} \subseteq \mathbb{R}^{n_{y_{\text{A/C}}}}$ is the output vector with $n_{\text{z}}$ representing the vertical load factor, which is a quantity related to the acceleration on the vertical axis. All the states describing the longitudinal dynamics are measurable using dedicated sensors. These measurements are, however, affected by delays that must be compensated in the control design (Section 5.3).

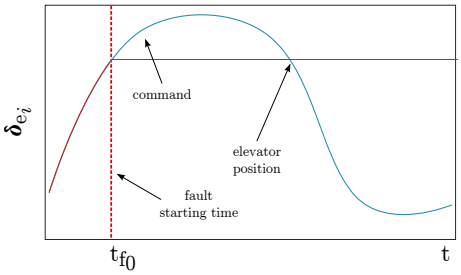


**Figure 5.1:** Stuck fault.

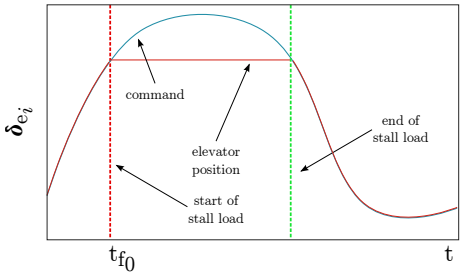


**Figure 5.2:** Stall load.

The elevator dynamics in the RECONFIGURE benchmark model can be modeled as third-order linear time-invariant (LTI) systems. The following model describes the elevator dynamics:

$$x_{\text{el}}(t+1) = A_{\text{el}}x_{\text{el}}(t) + B_{\text{el}}u(t) \tag{5.2a}$$
$$y_{\text{el}}(t) = C_{\text{el}}x_{\text{el}}(t) + D_{\text{el}}u(t) \tag{5.2b}$$

where $x_{\text{el}} \in \mathcal{X}_{\text{el}} \in \mathbb{R}^{n_{\text{el}}}$ (the components of $x_{\text{el}}$ are the elevator position, velocity, and acceleration), $u \in \mathcal{U}_{\text{MPC}} \subseteq \mathbb{R}^{n_u}$, and $y_{\text{el}} \equiv u_{\text{A/C}}$ (i.e., the elevator position).

Finally, we assume that $\mathcal{X}$, $\mathcal{U}, \mathcal{Y}, \mathcal{X}_{\text{el}}$, and $\mathcal{U}_{\text{MPC}}$ are polyhedral sets that contain the origin in their interior. Furthermore, in the remainder of the chapter, we use $\overline{\delta}_{\text{e}_i}$ and $\underline{\delta}_{\text{e}_i}$ to indicate the upper and the lower bounds of the $i$-th elevator output $\delta_{\text{e}_i}$ ($i \in \mathcal{I} := \{\text{li},\ \text{ri},\ \text{lo},\ \text{ro}\}$).

### 5.2.2 Fault Description

This work focuses on elevator jamming scenarios. In these scenarios, one or more elevators remain fixed at an unpredictable value $\delta^{\text{f}}_{\text{e}_i}$ ($i \in \mathcal{I}$), which might differ from their normal saturation limits. The elevator jamming can be attributed to two different root causes exemplified in Figures 5.1 and 5.2:

- *Stuck Fault*. The elevator is permanently jammed at a certain position $\delta^{\text{f}}_{\text{e}_i}$ and cannot be recovered (Figure 5.1). This effect can be modelled as a permanent change at time $t_{\text{f}}$ in the elevator's upper and lower operating bounds that become both equal to the jammed position $\delta^{\text{f}}_{\text{e}_i}$ $\forall t \geq t_{\text{f}}$.

- *Stall Load* [36]. The elevator is temporarily jammed during a dynamic manoeuvre, due to heavy aerodynamic forces preventing the elevator to achieve its commanded control surface deflection (Figure 5.2). In this situation, the elevator can still move within its reduced control limits $[-\underline{\delta}_{\text{e}_i}, \delta^{\text{f}}_{\text{e}_i}]$ or $[-\delta^{\text{f}}_{\text{e}_i}, \overline{\delta}_{\text{e}_i}]$, determined by the jammed position $\delta^{\text{f}}_{\text{e}_i}$. The stall load ends if either the manoeuvre becomes less dynamic or the aerodynamic forces acting on the control surface become smaller.

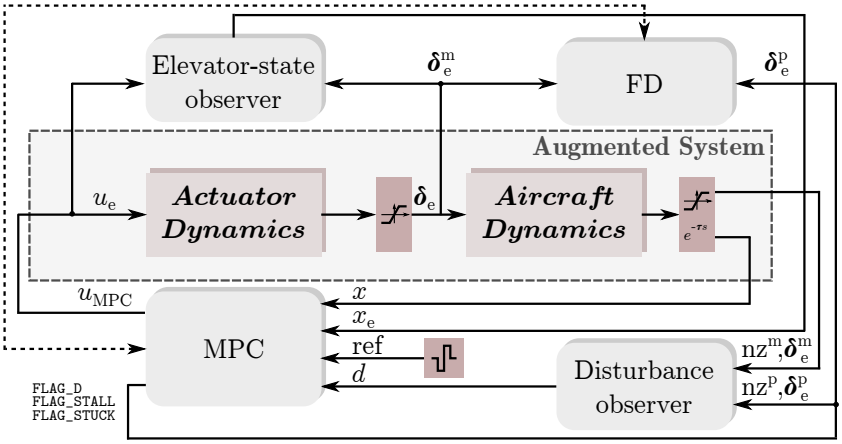


**Figure 5.3:** Proposed control architecture.

Considering their different consequences on the control limits and jamming duration, a stuck elevator and stall load need to be distinguished and require adopting different reconfiguration strategies in FTC. Nevertheless, because of the high similarity in the jamming phenomena, it is difficult to distinguish these two root causes. Hence, our proposed integrated FTC approach actively modifies the control strategies to help the FD module discriminate between the two root causes of the jamming, as detailed in Section 5.4.

*Remark* 5.2.1. This work focuses on jamming faults for which it is nontrivial to distinguish the root cause of the jamming. Although in some practical situations the stall load limits might change overtime leading to control challenges, from the diagnosis point of view we can still distinguish the root cause of the jamming easily in this case (when the fault is detected it is evident that the actuator is not permanently stuck at a given position). Hence, given that our goal is to design the interactions between the FD unit and the MPC controller to diagnose the root cause of a jamming fault, we do not focus on stall load scenarios with time-varying limits.

## 5.3 FTC Architecture

This section focuses on our proposed FTC architecture. In this respect, Figure 5.3 provides an overview of our proposed FTC design and show the interactions among the different components of our control system and the controlled plant. In particular, Figure 5.3 highlights *(i)* in dark grey the main components of the plant (i.e., the augmented aircraft model described in Section 5.2.1, the constraints depicted as saturation blocks, and the sensor delays) and *(ii)* in light grey the main components of our fault-tolerant controller. A detailed description of these components is provided in the remainder of the section.

### 5.3.1 Elevator-State Observer

The elevator states are needed by the MPC controller to build the predictions. By using the elevator model (5.2), four Luenberger observers [67], characterized by a constant gain $L$, are constructed. The gain $L$ is the same for all the operating points, given that the elevators are LTI systems (according to their description in the RECONFIGURE model). Each observer independently monitors one elevator. On one hand, the elevator-state estimates are needed to exploit the elevator dynamics in the MPC problem formulation. On the other hand, these elevator-state estimates are used to compute predicted elevator outputs $\delta_{\mathrm{e}}^{\mathrm{p}}$ for the disturbance observer and the FD module.

The realization we adopt for the elevators is such that, for each elevator, the state associated with the elevator position corresponds to the output of the elevator. Hence, when a saturation is detected on the $i$-th elevator position, the other two states (associated with the velocity and acceleration of the $i$-th elevator) are set to zero and the estimated position value is set to the measured elevator output. This allows us to estimate the elevator states without requiring a more advanced state estimator to handle saturation.

Note that if the model of the elevators is nonlinear or depends on the flight condition the gain $L$ should also vary accordingly. As previously stated, in this work we adopt the elevator description provided in the RECONFIGURE benchmark model, which assumes the elevators to be LTI systems.



### 5.3.2 Disturbance Observer

The disturbance observer is used to compensate constant measurement errors, reduce the effects of plant-model mismatches, and provide useful information to help the FD module detect jamming faults. The proposed observer strongly relies on the information provided by the MPC controller and on the plant measurements.

The observer is composed by two modules used to compensate *(i)* measurement errors and *(ii)* plant-model mismatches, respectively. In particular, the first module estimates a constant disturbance signal (that is then used by the MPC controller) as follows. First, we take into account that the MPC controller does not model the sensor and filter dynamics in the predictor to reduce the number of decision variables (and, consequently, the computation time). Hence, the proposed observer monitors $e_{n_z} := n_{\mathrm{z}}^{\mathrm{m}} - n_{\mathrm{z}}^{\mathrm{p}}$, that is, the mismatch between the measured and the predicted load factor. Second, the observer monitors $e_{\delta_{\mathrm{e}_i}} := \delta_{\mathrm{e}_i}^{\mathrm{m}} - \delta_{\mathrm{e}_i}^{\mathrm{p}}$, that is, the mismatch between the measured and the predicted elevator outputs, for elevator-jamming detection purposes. Hence, the first module of the disturbance observer estimates $d := [d_{n_z}\ d_{\mathrm{e}}^{\mathrm{T}}]^{\mathrm{T}}$ as follows:

$$d(t+1) = d(t) + \begin{bmatrix} e_{n_z} \\ e_{\delta_{\mathrm{e}_i}} \end{bmatrix}. \tag{5.3}$$

This estimated disturbance $d \in \mathbb{R}^{n_d}$ ($n_d = 5$) affects the predicted elevator outputs, the aircraft states, and the aircraft outputs. Hence, we must consider this disturbance as an additional state in the MPC prediction model as explained below.

The second module of the disturbance observer takes into account plant-model mismatches and, eventually, nonlinearities in the plant that are not modelled in the MPC controller, given that only linearized plant models are used to build the predictions. In this respect, we define an upper bound on these plant-model mismatches as $\varepsilon_{\text{nl}} := \|\hat{x}_t - x_{t|t-1}\|_2$, where $\hat{x}_t$ is the measured state of the aircraft (we omitted the subscript A/C to simplify the discussion) at time $t$ and $x_{t|t-1}$ is the value of the state at time $t$ predicted (by the MPC controller) according to the value of the measured state at time $t-1$. This upper bound monitors the distance between the predicted behavior of the plant and the real behavior and can be used (as explained below) to design a robust reference signal to avoid constraint violations in the MPC problem formulation.

*Remark* 5.3.1. The strategy described in (5.3) can only be used to estimate disturbances that can be modeled as constant values. Hence, given that the plant-model mismatches and the nonlinearities in the plant cannot be modelled as constant disturbances, we decided to include their effects in the definition of the MPC constraints as explained below.

### 5.3.3 Fault-Detection Module

The Fault-Detection (FD) module relies on the elevator-output prediction error $e_{\delta_{e_i}}$ to compute the residual signal used for the detection of jamming faults. The generated residual for each elevator is evaluated by its root mean square (RMS) value

$$J_i(t) := \sqrt{\frac{1}{N_{\text{eval}}} \sum_{k=t-N_{\text{eval}}+1}^{t} e^2_{\delta_{e_i}}(k)}\,, \qquad i \in \mathcal{I} \tag{5.4}$$

over a sliding window $[t - N_{\text{eval}} + 1, t]$. $N_{\text{eval}}$ is selected according to the slowest mode of the actuators. This is an empirical choice to give sufficient time to the physical system to register the jamming fault. The choice of $N_{\text{eval}}$ is a trade-off between reducing the risks of miss detection/false alarms and detection delay.

The fault detection decision is made by comparing each residual evaluation value $J_i(t)$ with the related threshold $J_i^{\text{th}}$, that is,

$$\text{FD Logic} : \begin{cases} J_i(t) \leq J_i^{\text{th}} & \Rightarrow \quad \text{fault-free in elevator } i \\ J_i(t) > J_i^{\text{th}} & \Rightarrow \quad \text{jamming in elevator } i. \end{cases} \tag{5.5}$$

After fixing the length of the sliding evaluation window, the thresholds $\{J_i(t)\}$ are determined by the plant-model mismatch of the elevator model (5.2). In practice, each threshold $J_i^{\text{th}}$ can be selected as the peak value of $J_i(t)$ in a large set of fault-free scenarios. In this work, we determine the thresholds by using dynamic fault-free manoeuvres (i.e., when stall loads might be more likely to occur). Its choice is

a trade-off between reducing the miss detections/false alarms and, at the same time, reducing detection delays.

*Remark* 5.3.2. Note that in this work we rely on a simple fault-detection logic with fixed $J_i^{\text{th}}$ to present our integrated approach. Nevertheless, the proposed approach can be extended with the use of more sophisticated detection techniques to select the threshold $J_i^{\text{th}}$ (for example, when an explicit description of multiplicative model uncertainties is taken into account).

Furthermore, we add an additional check to improve the detection of isolated faults for which we can exploit redundancy, that is, the presence of redundant control surfaces. In fault-free conditions, the residual signals of each elevator are sufficiently small and close to each other (in terms of magnitude). Suppose that one of the residual signals starts deviating from the others. This abnormal behavior is an indicator that the elevator associated with that residual signal might be jammed. This strategy is useful when we have to deal with isolated faults on one or two actuators. For example, this strategy is useful to anticipate the detection of a stuck fault, because a permanent jamming is more likely to occur on a single elevator.

*Remark* 5.3.3. The detection logic described above is insufficient to identify the root cause of jamming by itself given that it only informs the controller that the actuator is jammed. At this stage the controller does not know whether the jamming is permanent or temporary. In Section 5.4, we combine the detection logic ((5.5)) with different active reconfigurations to capture more detailed fault information.



### 5.3.4 Model Predictive Controller

MPC controllers rely on *(i)* the plant description to build predictions of the plant behavior over a predefined time window (called prediction horizon), *(ii)* the information on state, input, and output constraints, and *(iii)* current measurements from the plant, such as state measurements and desired reference signals. These controllers offer an intuitive and structured framework to compute the optimal control law to simultaneously satisfy the control objectives and constraints on the plant. This control law is computed by solving (either offline or online [4; 28; 101; 90; 141], depending on the number of decision variables) an optimization problem (usually a quadratic programming problem). For more details on MPC refer to [72; 69; 6] and the references within.

*Remark* 5.3.4. In this work, we solve the MPC optimization problem online. This requires solving a quadratic programming (QP) problem of size proportional to the number of decisions variables and length of the prediction horizon. The solution of this optimization problem in an embedded environment can be challenging, due to small sampling times and limited hardware and software resources (the availability of a QP solver is usually not guaranteed). First-order solvers, such as proximal-gradient and splitting methods (refers to [88; 123] and the references within for an overview) are valid solutions for this problem. In this respect, in the context of

aerospace applications, in [22], we show on the RECONFIGURE benchmark model how we can efficiently compute the MPC problem by relying on these first-order solvers (in particular, by combining the use of Nesterov's dual fast gradient and the alternating direction method of multipliers).

With this framework in mind, we define the model used to compute the predictions in the MPC controller. In particular, given (5.1)-(5.2), this model is computed as follows:

$$x(t+1) = Ax(t) + Bu(t) \quad t \geq 0, \tag{5.6a}$$
$$y(t) = Cx(t) + Du(t) \quad t \geq 0, \tag{5.6b}$$

where $x := [\bar{x}_{\text{A/C}}^{\text{T}}\, x_{\text{el}}^{\text{T}}\, \hat{d}^{\text{T}}]^{\text{T}} \in \mathcal{X}_{\text{MPC}} \subseteq \mathbb{R}^n$ (where $\bar{x}_{\text{A/C}}^{\text{T}} := [q\, v\, \alpha\, h]$ takes into account a subset of the longitudinal states to maintain the size of the prediction model small and $n := n_{\text{A/C}} - 2 + n_{\text{el}} + n_d$), and $y := [y_{\text{A/C}}^{\text{T}}\, y_{\text{el}}^{\text{T}}]^{\text{T}} \in \mathcal{Y}_{\text{MPC}} := \mathcal{Y} \times \mathcal{U} \subseteq \mathbb{R}^{n_{y_{\text{A/C}}} + n_u}$. The structure of $A$, $B$, $C$, and $D$ follows from the choice of the state, input, and output for the cascade actuator-aircraft dynamics depicted in Figure 5.3 (namely, the augmented system) and by describing the disturbance dynamics as constant, that is, $\hat{d}(t+1) = \hat{d}(t)$, where $\hat{d}(t) = d(t)$ (5.3).

*Remark* 5.3.5. Note that we use linearized aircraft models in the MPC problem formulation (as described in Section 5.2.1 as well) to explain our algorithm. Nevertheless, the approach can potentially be extended to linear-parameter varying (LPV) or linear time-varying (LTV) models [71; 115; 16; 94].

In the remainder of the chapter, we consider the following assumption:

**Assumption 5.3.1.** *The augmented system is stabilizable.*

Our goal is to control the longitudinal dynamics of the aircraft. In particular, our goal is to steer the output of system (5.6) to a desired reference value denoted by $\nu$, which is generated by a pilot stick command. The reference value is measured at each sampling time and we assume that is constant along the length of the prediction horizon in the MPC problem formulation. Furthermore, we have to take into account the constraints acting on state, input, and output, that are, $\mathcal{X}_{\text{MPC}}$, $\mathcal{U}$, and $\mathcal{Y}_{\text{MPC}}$, respectively. Hence, compared to [21], we rely on a modified version of the MPC for tracking formulation proposed in [66; 19]. In particular, we can formulate our MPC

problem as follows:

$$\mathcal{V}^*(\nu, x_{\text{init}}) := \underset{x,u,\theta}{\text{minimize}} \sum_{t=0}^{N} l_t(\nu, x_t, u_t, \theta_t) \tag{5.7a}$$

$$\text{subject to: } Ax_t + Bu_t = x_{t+1}, \quad t = 0, \dots, N, \tag{5.7b}$$

$$\begin{bmatrix} \hat{x}_t \\ \hat{u}_t \end{bmatrix} = M_\theta \theta_t, \quad t = 0, \dots, N, \tag{5.7c}$$

$$G_x x_t + G_u u_t + g \leq 0 \quad t = 0, \dots, N, \tag{5.7d}$$

$$G_x \hat{x}_t + G_u \hat{u}_t + g_\theta + E\epsilon_{\text{nl}} \leq 0 \quad t = 0, \dots, N, \tag{5.7e}$$

$$\hat{y}_t = N_\theta \theta_t \quad t = 0, \dots, N, \tag{5.7f}$$

$$x_0 := x_{\text{init}}, \tag{5.7g}$$

where $x_t \in \mathbb{R}^n$, $u_t \in \mathbb{R}^{n_u}$ indicate the $t$-step-ahead state and control predictions, respectively. In addition, (5.7d) represents the constraints on the predicted state, input, and output ($G_x \in \mathbb{R}^{c \times n}$, $G_u \in \mathbb{R}^{c \times n_u}$, and $g_\theta = g$ in fault-free operating conditions) that follow from the definition of $\mathcal{X}_{\text{MPC}}$, $\mathcal{U}$, and $\mathcal{Y}_{\text{MPC}}$. Furthermore, $\theta_t \in \mathbb{R}^{n_u}$ is the vector of parameters used to generate the *artificial* steady state, input, and output $\hat{x}_t$, $\hat{u}_t$, and $\hat{y}_t$, respectively. $M_\theta$ and $N_\theta$ are suitable matrices (refer to [66] for details). For a prediction horizon of length $N$, the cost $l_t$ in (5.7a) is described as follows:

$$l_t(\nu, x_t, u_t, \theta_t) := \|x_t - \hat{x}_t\|_Q^2 + \|u_t - \hat{u}_t\|_R^2 + \rho_1 \|\hat{y}_t - \nu\|_2^2, \tag{5.8}$$

where $Q = Q^{\text{T}} \in \mathbb{S}_{\geq 0}^n$, $R = R^{\text{T}} \in \mathbb{S}_{>0}^m$, and $\rho_1 > 0$.

The main idea of the artificial reference associated with the parameters $\theta_t$ in Problem (5.7) is to generate a reference for the states and the control inputs that achieves the control objectives (i.e., the tracking of the reference $\nu$) while satisfying the constraints on the system. This strategy allows one to compromise between tracking performance and feasibility of the solution when the commanded reference $\nu$ does not lead to feasible state and control trajectories. In this respect, note that in the cost the distance between the desired reference and $\hat{y}_t$ is penalized by a factor $\rho_1 > 0$ (which is a tuning parameter of our design) in order to generate an output trajectory close to the desired one. At the same time, the constraints (5.7e) prevent that the generated trajectory along the prediction horizon becomes infeasible. This strategy has the following advantage compared to the one proposed in [23]. For every problem instance, if a jamming fault is detected on the actuators, with a simple reconfiguration of the constraints on $\theta_t$ (i.e., by changing the definition of $g_\theta$ according to the severity of the fault, but without changing the initial feasible region of the states and control commands) we can generate a feasible reference signal for the state, input, and output that steers the system towards the new (post fault) feasible region. This reference signal is clearly suboptimal (note that we are using the 2-norm in (5.8) to penalize the distance from $\nu$, which is not an exact penalty), but ensures a safer transition to the after-fault feasible region of the controller.

*Remark* 5.3.6. One concern when using this approach is related to the stability of the system controlled by the MPC controller. In [19] a terminal set for tracking is introduced in the MPC problem formulation to guarantee stability. When a jamming fault occurs, this impacts the definition of the terminal set that shrinks according to the severity of the fault. While a rigorous stability proof is out of the scope of this manuscript (our main focus is to provide a strategy for active diagnosis of jamming faults using control reconfiguration and, consequently, in the remainder of the chapter, we consider maneuvers that do not impact the stability of the system), we provide different possible strategies/guidelines to design a robust MPC controller in the presence of faults:

1. The jamming faults can be considered as (possibly persistent) disturbances bounded in a given set $\mathcal{W}$ computed based on some heuristics (for example, by considering different fault combinations). The robust terminal set for tracking computed based on the worst combination of faults can then be used in the MPC formulation (leading to a tube-based MPC design [64] for tracking).

2. If in the current setup we include a terminal set for tracking (according to [19]), when a fault occurs, the only reconfigurations in the MPC problem formulation affect the parameters $\theta$ used to generate the artificial reference signal. The optimizer computes the best artificial reference trajectory to compromise between tracking performance and constraint satisfaction. Hence, if we tighten (according to the severity of the fault) the constraints associated with the parameters $\theta$ this should directly prevent violation of the original terminal set for tracking (which remains unmodified for the states and control commands).

3. Alternatively, if we include a terminal set for tracking in the current MPC formulation (as in the previous point), a solution could be to tighten the terminal set by an amount proportional to the fault and uncertainties in the model. The terminal set associated with the augmented aircraft model takes into account also the dynamics of the actuators. Consequently changes in the actuator bounds will impact the dynamics and the choice of the associated tightening parameters.

An interesting alternative to be investigated (as part of our future research and out of the scope of this manuscript) is related to the use of infinite horizon MPC formulations [117; 122; 25], that are recently gaining increasing attention and can remove the requirements of a terminal set in the MPC problem formulation.

Note that the constraints on the artificial states (5.7e) are tightened ($E$ is the matrix used to select the subset of state constraints where the tightening occurs), compared to (5.7d), by a quantity $\epsilon_{\text{nl}}$, which is computed by the disturbance observer (presented in Section 5.3.2) at each sampling time. This additional tightening allows the controller to take into account the effects of the plant model-mismatches/nonlinearities, which are not modelled in the prediction model (5.7b)

and cannot be modelled as constant disturbances (5.3). Consequently, the pairs $(\hat{x}_t, \hat{u}_t)$ are generated to take into account these plant-model mismatches leading to a *robust* artificial reference generation, without directly affecting the feasible region of the states and control inputs. Note that constraint tightening is a technique used in robust MPC to avoid infeasibility in the presence of disturbances (the interested reader can refer to [105] and the references within).

In general, the MPC controller solves Problem (5.7) online from the plant and returns an optimal sequence of states and control inputs that minimizes the cost (5.7a). Let the optimal sequence be defined as follows:

$$\{\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}\} := \{x_0, \ldots, x_N^*, u_0^*, \ldots, u_{N-1}^*, \theta_0^*, \ldots, \theta_N^*\}. \tag{5.9}$$

Only the first element of $\mathbf{u}$ is implemented in closed-loop, that is, the control law obtained using the MPC controller is given by:

$$\kappa_{\text{MPC}}(\nu, x_{\text{init}}) = u_0^*, \tag{5.10}$$

and the closed-loop system is described by

$$x(t+1) = Ax(t) + B\kappa_{\text{MPC}}(\nu, x_{\text{init}}). \tag{5.11}$$

With this framework in mind, the next section details the interactions between the FD module and the MPC controller to actively detect and diagnose the root cause of jamming faults.



## 5.4 Proposed FD-MPC Design

This section aims to describe the close interactions between the FD module and the MPC controller (described in Sections 5.3.3 and 5.3.4, respectively) in our proposed integrated FTMPC approach. Figure 5.4 summarizes these interactions. In the following, we show how the fault information obtained by the FD module is exploited by the MPC controller and how the MPC controller actively modifies its reconfiguration strategies to assist the FD module in diagnosing the root cause of a detected elevator jamming.

### 5.4.1 Detection

As Figure 5.4 shows, during the detection phase, the FD module constantly monitors each elevator by evaluating its corresponding residual signal $e_{\delta_{\text{e}_i}}$ with $J_i$ in(5.4) ($i \in \mathcal{I}$). If the residual evaluation signal $J_i$ associated with the $i$-th elevator at time $t_{\text{f}_i}$ exceeds the predefined threshold $J_i^{\text{th}}$ or differs from the others as described in Section 5.3.3, the FD module detects that the $i$-th elevator is jammed. At this stage, the root cause of jamming is still unknown. Hence, the FD module sends a message to the MPC controller to activate the first reconfiguration (i.e., reconfiguration for diagnosis in Figure 5.4).

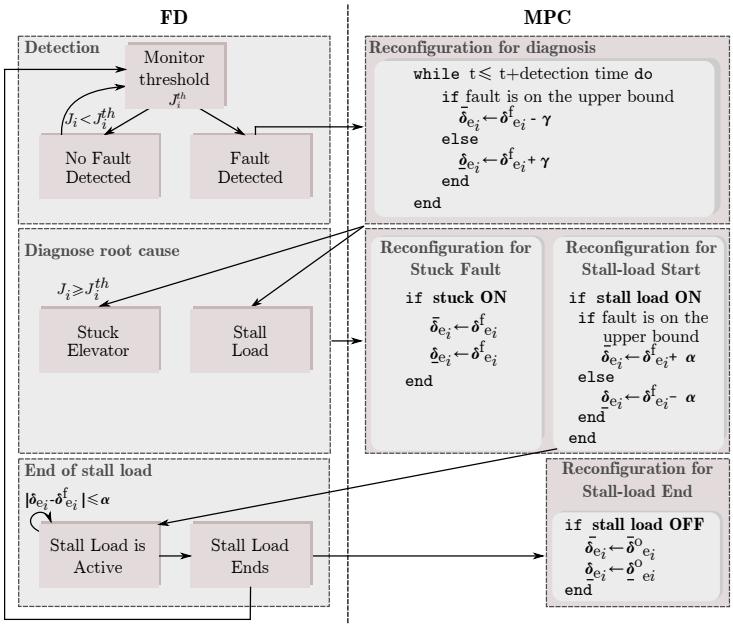


**Figure 5.4:** Proposed FD-MPC design.

### 5.4.2 Reconfiguration for Diagnosis

The aim of the reconfiguration for diagnosis is to help the FD module understand the root cause of the jamming fault. The MPC controller checks the sign of $e_{\delta_{e_i}}$ at time $t_{f_i}$ to decide whether to modify $\overline{\delta}_{e_i}$ or $\underline{\delta}_{e_i}$, that is, the upper or the lower bounds of the $i$-th elevator. Note that this modification in the MPC problem formulation affects only the definition of $g_\theta$ (i.e., the feasible region of the parameters $\theta$ used to generate the artificial reference signal). The idea is to temporarily set the jammed elevator bound to a tightened value $\delta^{f}_{e_i} \pm \gamma$, where $\delta^{f}_{e_i}$ is the measured value of the elevator at time $t_{f_i}$ and $\gamma$ is a positive constant that should be tuned sufficiently small to preserve the performance of the controller, but, at the same time, large enough to allow the size of residual signal to exceed the predefined threshold $J_i^{th}$ for a stuck elevator. Note that the positive or negative ($\pm$) sign depends on the bound that the MPC modifies, according to the description in Figure 5.4. The MPC maintains this new $\gamma$-tightened bound for $\tau$ samples. On one hand, $\tau$ must be selected sufficiently large to ensure that the control commands $u$ have time to adjust to the updated (in terms of feasible region) parameters $\theta$. On the other hand, $\tau$ must be small enough to preserve performance (especially in case of false alarms or stuck faults). It is reasonable to set $\tau$ proportional to the prediction horizon $N$.

### 5.4.3 Diagnosis of the Root Cause

If $J_i(t_{\mathrm{f}_i} + \tau) < J_i^{\mathrm{th}}$ at the end of the diagnosis period, the FD module confirms a *stall load* as the root cause of the jamming fault, because the controller showed (using the reconfiguration for diagnosis) that jammed elevator can still move within its reduced bounds. If $J_i(t_{\mathrm{f}_i} + \tau) \geq J_i^{\mathrm{th}}$, the FD module confirms a *stuck elevator* as the root cause of the jamming fault, because the faulty elevator was unable to reach the tightened bound.

### 5.4.4 Reconfiguration for Stuck Fault

As soon as the FD module communicates the root cause of the jamming fault, the MPC controller performs the second reconfiguration. If the diagnosis is that the elevator is stuck, the MPC controller performs the reconfiguration for the stuck elevator by setting both $\underline{\delta}_{\mathrm{e}_i}$ and $\overline{\delta}_{\mathrm{e}_i}$ in the definition of $g_\theta$ to $\delta_{\mathrm{e}_i}^f$, as Figure 5.4 shows. In this way, the artificial reference is generated to take into account that the $i$-th elevator is permanently stuck at the fault position and adapts the reference for the remaining healthy elevators accordingly. This second reconfiguration is also the last one for the stuck elevator.

### 5.4.5 Reconfiguration for Stall-Load Start

If the diagnosis is stall load on the $i$-th elevator, the MPC controller performs the reconfiguration for stall-load start to allow the detection of the end of the stall load. In this respect, the controller sets the previously modified bound ($\overline{\delta}_{\mathrm{e}_i}$ or $\underline{\delta}_{\mathrm{e}_i}$ depending on the sign of $e_{\delta_{\mathrm{e}_i}}$ at time $t_{\mathrm{f}_i}$) to the new value $\delta_{\mathrm{e}_i}^f \pm \alpha$, that is, the controller allows a $\alpha > 0$ larger bound for the $i$-th elevator, but does not restore the original bound ($\overline{\delta}_{\mathrm{e}_i}^{\mathrm{o}}$ or $\underline{\delta}_{\mathrm{e}_i}^{\mathrm{o}}$) yet. This new limit allows one to detect whether the elevators deviate from the temporarily jammed position at the end of the stall load.

*Remark* 5.4.1. Setting $\alpha = 0$ could prevent the FD module to monitor the end of the stall load because the elevator cannot follow a command that exceeds its reduced bound. The reduced bounds of elevators due to miss detecting the end of a stall load may lead to sever control performance degradation.

### 5.4.6 Detection of the End of Stall Load

During the reconfiguration for stall-load start, the FD module constantly monitors the discrepancy between the measured elevator position $\delta_{\mathrm{e}_i}^{\mathrm{m}}$ and its previously jammed position $\delta_{\mathrm{e}_i}^f$. If $|\delta_{\mathrm{e}_i}^{\mathrm{m}} - \delta_{\mathrm{e}_i}^f| \leq \alpha$, the FD module communicates that the stall load is still active on the $i$-th elevator and the MPC controller maintains its current formulation. When this condition is violated, the FD module communicates the end of the stall load to the controller and returns to monitor the residual value.

### 5.4.7 Reconfiguration for Stall-Load End

When the stall load ends, the MPC must restore the original saturation limit (i.e., $g_\theta = g$), which is the last reconfiguration for the stall load.

*Remark* 5.4.2. The MPC reconfiguration can handle more than one elevator fault at a time, thanks to the decoupled structure of the FD module, which monitors each elevator independently. In this work, however, we consider symmetric faults, that is, if a jamming fault occurs on the left inner elevator, the same fault occurs on the right inner elevator. The reason for this choice is related to the fact that nonsymmetric faults affect the lateral behavior of the aircraft and would require a different (more complex) model to build the MPC predictions.

*Remark* 5.4.3. Compared to [23], all the reconfigurations in the MPC problem formulation do not affect the states and the control commands, but only the feasible region of the parameters $\theta$. These reconfigurations affect the way the artificial reference is generated and allow a smoother transition from the fault-free region to the faulty feasible region (by generating a feasible reference signal for the states and actuators for every problem instance).

### 5.4.8 Discussion

The proposed algorithm relies on the interactions between the FD unit and the MPC controller. In this work, we proposed a simple FD design and an LTI MPC formulation to simplify the presentation of our approach (as pointed out in Remarks 5.3.2 and 5.3.5).

The success of our proposed algorithm depends on the accuracy of the detection and diagnosis. In general, the fault detection and diagnosis accuracy depends mainly on $N_{\text{eval}}$, $J_i^{\text{th}}$ and $\tau$. These parameters determine the delay from fault occurrence to control reconfiguration. On one hand, if we set these parameters so that the delay is short, the FD results are less accurate. Consequently, control performance is sacrificed. On the other hand, if we set those parameters so that the delay is larger, the FD results are more accurate, but the control performance would still be sacrificed (due to the larger delay). This suggests a trade-off in the waiting time for the reconfiguration. Detailed theoretical analysis of such an integration for FD parameter tuning is an open theoretical challenge [143]. Nevertheless, the intuitive understanding above provides a guideline for tuning.

The proposed design is robust to scenarios that might lead to misdetection or misdiagnosis of actuator faults. For example, if the *reconfiguration for diagnosis* is triggered by a misdetection in the FD unit, a temporary reconfiguration of the actuator bounds will be performed leading to $\tau$ time instances of conservative behavior. In most cases, the redundancy in the number of actuators (that allows to reallocate the control action on the healthy control surfaces) will mitigate the conservatism due to the misdetection.

A more severe situation that the proposed algorithm does not address is related to the misdiagnosis of a stuck fault. In particular, suppose that $\tau$ is too short and the residual signal does not have enough time to decrease during the diagnosis phase. In this scenario, a stuck fault for an healthy elevator is diagnosed by our algorithm. This misdetection can seriously affect the performance, especially if all the longitudinal control surfaces are erroneously diagnosed as stuck. The algorithm can be modified to include additional control surfaces (e.g., the ones associated to the lateral dynamics) to compensate for the fault, or techniques to recover from the misdiagnosis of a fault must be implemented for this particular scenario.

## 5.5 Simulation Results

This section presents numerical results of our integrated control strategy on an Airbus passenger aircraft simulator that has been the benchmark model of the RECONFIGURE project [37].

The threshold $J_i^{\text{th}}$ in the FD module is selected according to the guideline of Section 5.3 and is equal to 0.40 for the inner elevators and to 0.65 for the outer elevators (the thresholds are different given the differences between the inner and outer elevator models). In addition, we implemented the detection strategy that exploits redundancy described in Section 5.3.3. In this respect, the FD unit detects a fault on the $i$-th elevator if $J_i >= 4J_j$, $i \neq j$, $i, j \in \mathcal{I}$, that is, when the residual signal of the $i$-th elevator is four times larger than the residual signals of the other elevators. In addition, we selected the time required for the diagnosis of the root cause of the jamming as $\tau = NT_s$ ($N = 20$ is the length of the prediction horizon and $T_s := 0.04$ sec is the sampling time of the system), that is, $\tau$ is selected proportional to the prediction horizon used in the MPC problem formulation. Another parameter that requires a trade-off between performance and accuracy is $\gamma$, used to tighten the faulty-elevator constraints during the *reconfiguration-for-diagnosis* phase. We noticed that a small value of $\gamma$ (e.g., 1% of the maximum allowed control command) is sufficient for the diagnosis. Finally, we selected $\alpha$ sufficiently large (e.g., $3\gamma$) to avoid false alarms in the detection of the stall-load end right after the diagnosis phase.

We trimmed the aircraft at an altitude of $12,500$ feet and calibrated airspeed of $335$ knots (inside the flight envelope) and we used the linearized model of the aircraft at the trimmed operating condition to build the MPC prediction model. Our aim is to track a doublet signal on the vertical load factor, that is, $\nu := n_{z_{\text{ref}}}$. Specifically, we consider the sequence of two doublets of different amplitude. The first doublet starts at 0.04 sec and ends at 20.04 sec and its value exceeds the allowed constraints on the vertical load factor. The second doublet starts at 30.04 sec and ends at 50.04 sec and its value remains within the constraints of the vertical load factor. We study the performance of our integrated design in the following scenarios:

- Stall load occurring at 2.65 sec from the beginning of the simulation on the

inner elevators.

- Stuck fault occurring on the inner elevators at 2.65 sec from the beginning of the simulation.

The baseline to evaluate the performance of the proposed integrated design is the behavior of the system controlled by the MPC controller, in the fault-free case. Note that we simulate the occurrence of the faults during the first doublet when the reference signal starts exceeding the vertical load factor bounds. Furthermore, in the following, recall that all the reconfigurations operate on the feasible region of the artificial reference signal (as discussed in Section 5.4) and do not affect the original feasible region of the states and actuators.

### 5.5.1 Stall Load

Figures 5.5-5.6 present the results obtained using the proposed algorithm (i.e., the integrated FD-MPC design) in case of stall load on the inner elevators. In this scenario the outer elevators are healthy.

Figure 5.5 details the behavior of the vertical load factor. During the first part of the manoeuvre the stall load occurs. The proposed algorithm allows the controller to avoid the constraint violation of the vertical load factor (that would have occurred without a tailored control reconfiguration) with a minor loss of performance (less than 5%) compared to the fault-free case (dot-dashed green line). Figure 5.6[1] details the behavior of the elevators and of the residual signals during the detection and diagnosis of the stall load. Once the fault is detected, the MPC controller immediately updates the lower bound of the faulty inner elevators. Consequently, the outer elevators (second row) compensate for the temporary loss of the inner elevators (first row) leading to an overall control action (third row) that is comparable to the one in the fault-free case.

The detection and diagnosis of the fault is fundamental for the performance of the controller. In particular, as shown in the last row of Figure 5.6, the FD unit alerts the MPC controller as soon as the residual signal $J_i$ of the inner elevators starts to abnormally increase with respect to the one of the outer elevators. When the anomaly is detected the MPC controller proceeds to perform the reconfiguration for fault diagnosis (first row) and adapts the reference signal to maintain feasibility. At the end of the detection time, given that the residual signal is below the threshold, the FD unit notifies the MPC controller of the occurrence of a stall load. Note that, at the end of the detection phase, the inner elevators are no longer in stall, but they remain close (within $\alpha$) to the lower bound. Hence, the FD unit does not immediately declare the end of the stall load and waits that the conditions for the *reconfiguration for stall-load end* are met. As soon as the inner elevators move away from their

[1]The units on the vertical axis of the elevator plots (Figures 5.6, 5.8) have been removed upon request of our industrial partners.

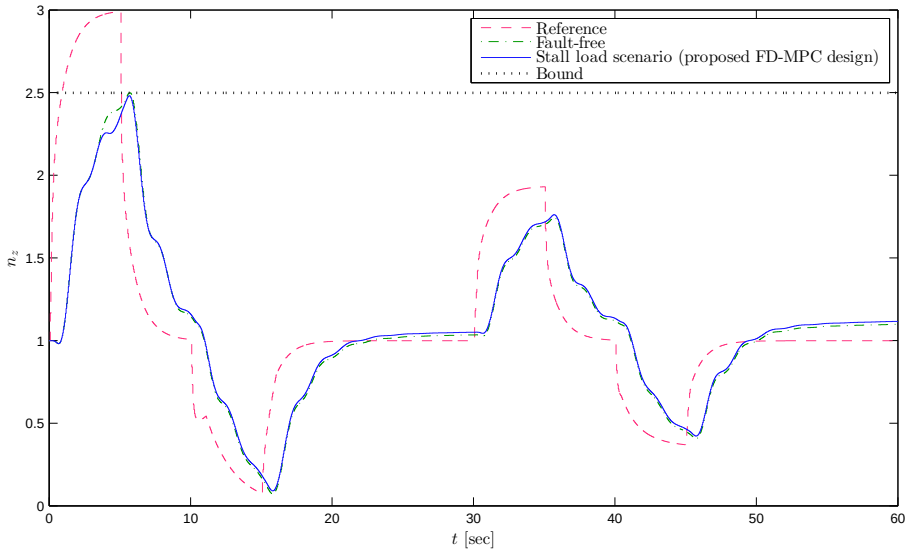


**Figure 5.5:** Comparison of the vertical load factor tracking performance in the fault-free case (dot-dashed green line) and when a stall load on the inner elevators (at 2.65 sec from the beginning of the simulation) is detected and diagnosed using the proposed integrated design (solid blue line).



reduced saturation bounds the stall load ends and the MPC controller restores the original elevator bounds.

### 5.5.2 Stuck Fault

Figures 5.7-5.8 present the results obtained using the proposed integrated control algorithm in case of permanent jamming of the inner elevators. In this scenario the outer elevators are healthy.

Figure 5.7 details the behavior of the vertical load factor. During the first part of the manoeuvre the inner elevators become jammed. The proposed algorithm, thanks to the detection and diagnosis of the root cause of the jamming fault, allows the controller to avoid the constraint violation of the vertical load factor with a minor loss of performance. Note that without the proposed sequence of reconfigurations for detection and diagnosis, due to the severity of the fault, the MPC controller would not be able to maintain the system within its feasible region and ensure stability.

Figure 5.8 presents the behavior of the elevators and of the residual signals during the detection and diagnosis of the stuck fault. Once the fault is detected, the MPC controller immediately performs the first reconfiguration (as done in the previous case for the temporary jamming) to update the bounds associated with the inner elevators in the feasible region of the artificial reference. During the detection phase,

compared to the previous scenario, the residual signal of the inner elevators (solid blue line on the last row of Figure 5.8) increases. At the end of the detection time the residual signal associated with the inner elevators is still above the predefined threshold and the FD module can diagnose the permanent jamming of the inner elevators. After the diagnosis, the MPC controller performs the reconfiguration for stuck fault by updating the upper and lower bound of the faulty elevators in the MPC problem formulation (as also the first row of Figure 5.8 depicts). The tracking performance is maintained (compared to the fault-free case depicted in dash-dotted green lines) with limited loss thanks to the reallocation of the control authority on the healthy outer elevators (second and third row of Figure 5.8). The minor performance loss is due mainly to the inner elevator being stuck to a nonzero value and the presence of physical rate limitations in the actuators that affect the response of the outer elevators to the loss of the inner ones.

## 5.6 Conclusions

We presented a novel fault-tolerant controller tailored to aerospace applications. Our approach relies on the close interaction between a fault-detection (FD) module and a model predictive controller (MPC). The FD module exploits the controller to diagnose the root cause of the elevator jamming and the MPC exploits the information provided by the FD module to better handle the jamming. We showed on an Airbus passenger aircraft simulator the benefits that our strategy can bring to the performance of the control system.

As the numerical example showed, the proposed integrated design provides an effective strategy for the detection and active diagnosis of jamming faults that can occur on the aircraft actuators. Furthermore, the reconfiguration and fault-tolerant reference generation allows one to preserve the tracking performance after the occurrence of the fault.

A limitation of the current approach is related to the definition of the threshold used to activate the diagnosis. Exploiting the information provided by the other actuators helps the early detection of the faults, but if all the control surfaces are affected by a fault (e.g., in case of temporary jamming) the choice of the threshold remains critical. As part of our future work, we plan to investigate different strategies on the threshold selection (for example, by exploring the relationship with the amplitude of the reference signal and disturbances) to improve the detection of the fault.

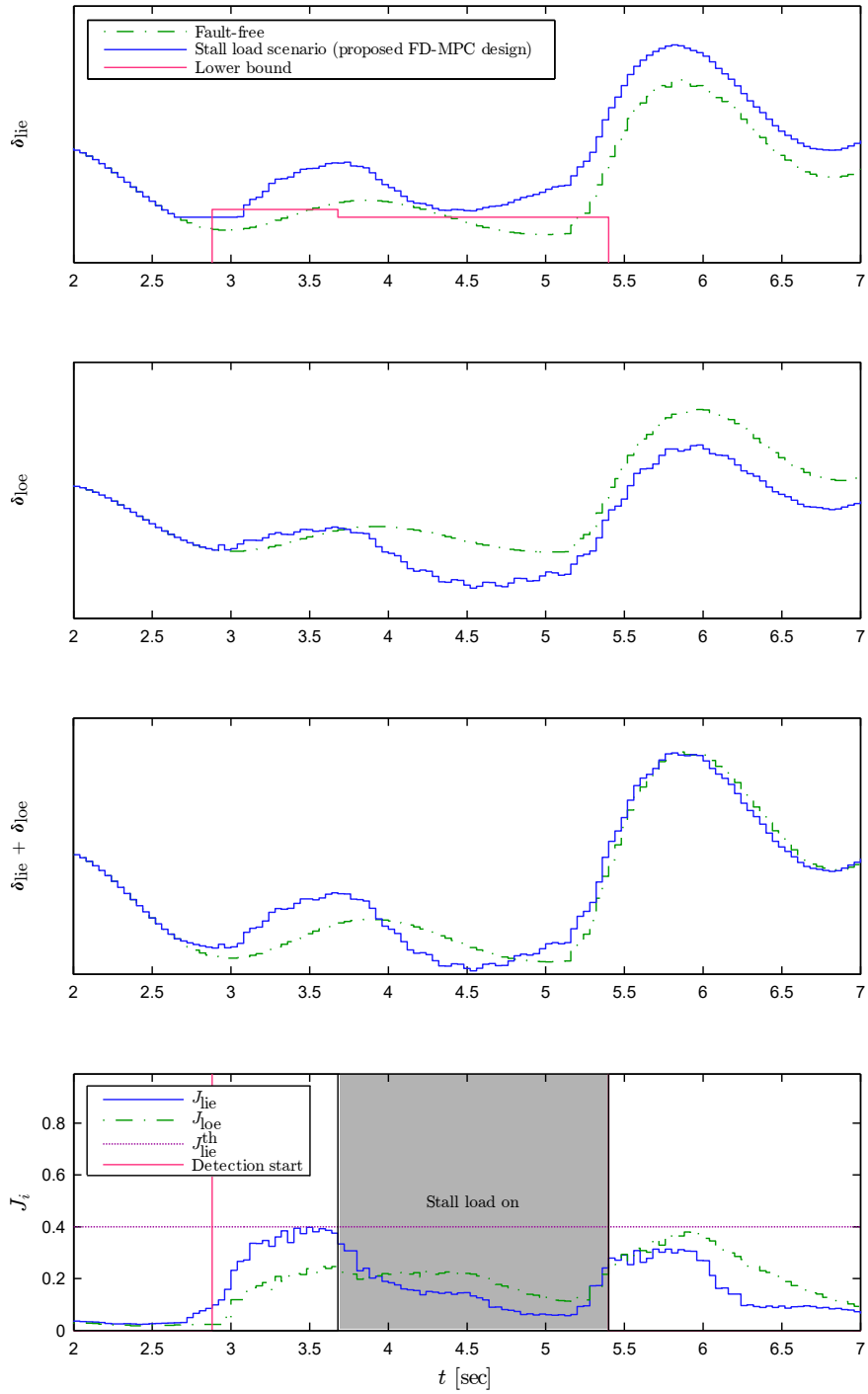


**Figure 5.6:** Comparison of the elevator behaviors (rows 1–3) in the fault-free case (dot-dashed green line) and when a stall load on the inner elevators is detected and diagnosed using the proposed integrated design (solid blue line). The last row depicts the behavior of the residual signals used to detect and diagnose the fault. The grey area highlights the duration of the reconfiguration for stall load start.

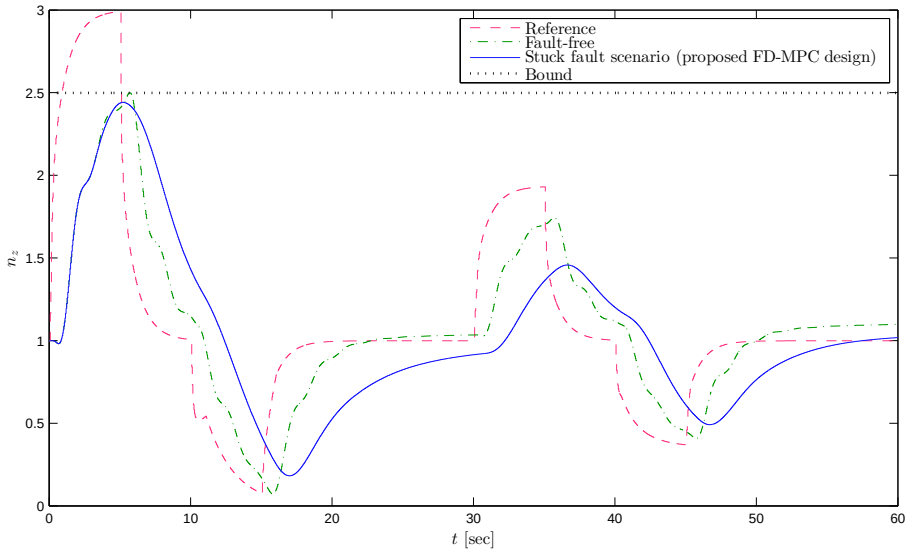


**Figure 5.7:** Comparison of the vertical load factor tracking performance in the fault-free case (dot-dashed green line) and when a permanent jamming of the inner elevators (at 2.65 sec from the beginning of the simulation) is detected and diagnosed using the proposed integrated design (solid blue line).

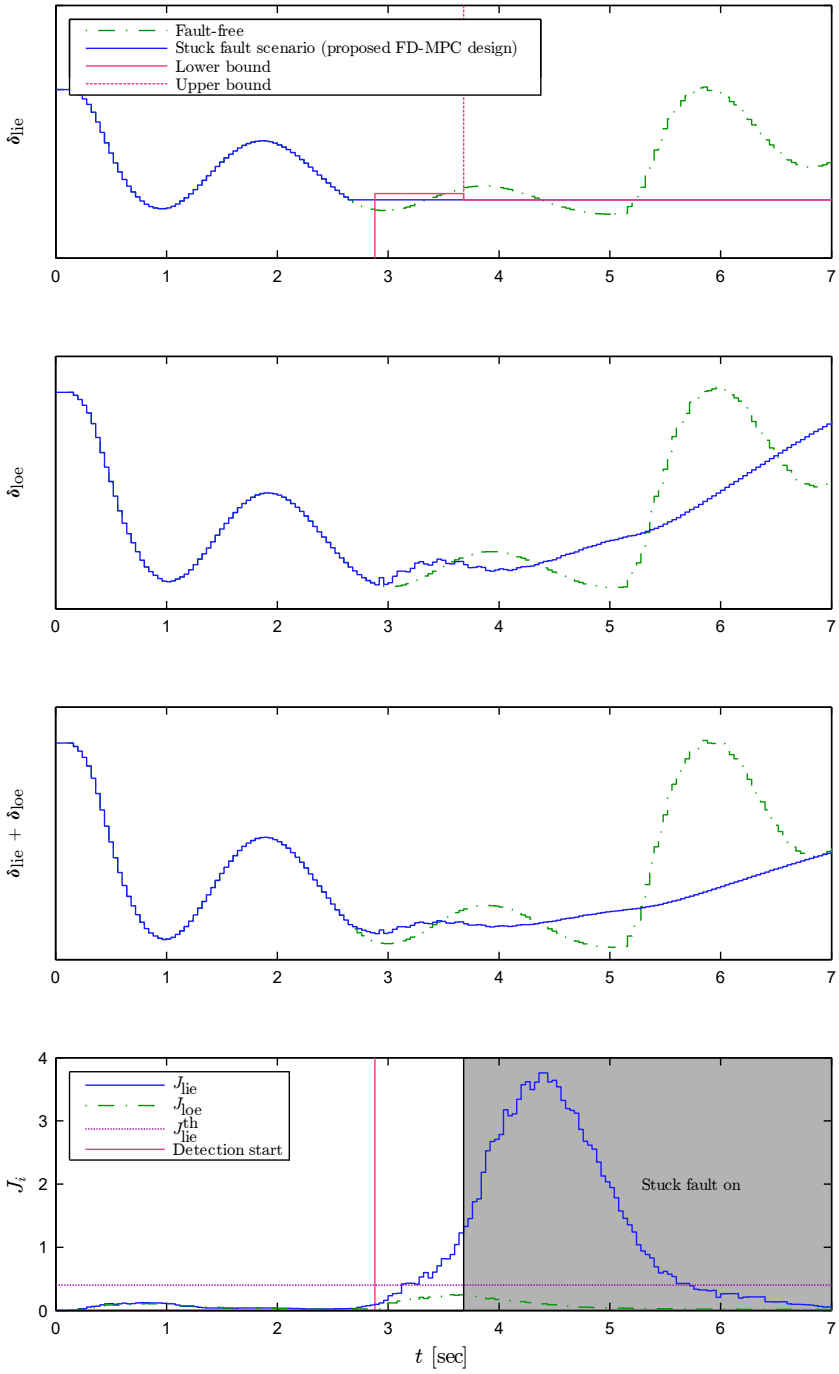


**Figure 5.8:** Comparison of the elevator behaviors (rows 1–3) in the fault-free case (dot-dashed green line) and when a stuck fault on the inner elevators is detected and diagnosed using the proposed integrated design (solid blue line). The last row depicts the behavior of the residual signals used to detect and diagnose the fault. The grey area highlights the reconfiguration for a stuck fault.

# 6 Conclusions

This dissertation focused on the design of first-order methods for MPC in applications with fast dynamics, motivated by the benchmark problem of the EU funded project RECONFIGURE. The main techniques we relied on in this work are proximal-gradient and splitting methods. In the following, we summarize the key results of this dissertation.

- **Adaptive constraint-tightening strategy for MPC.** We showed how to deal with the early termination of a first-order solver to guarantee recursive feasibility and closed-loop stability of the system controlled by the MPC controller. The main idea was to deal with the early termination of the solver in a robust way, that is, by tightening the feasible region of the MPC problem by a quantity proportional to the possible constraint violation. The amount of tightening is selected online to reduce the conservatism in the solution.

- **Parallel constraint-tightening strategy for MPC.** The approach above was initially developed to handle the possible infeasibility and suboptimality of the MPC problem solution obtained from Nesterov's dual fast gradient scheme. This theory was then nontrivially extended and improved by using splitting methods to exploit the structure of the MPC problem. The resulting algorithm is suitable for implementation on parallel hardware architectures. Furthermore, by exploiting the structure of the MPC problem, we showed how to improve the numerical properties of the MPC problem itself and, consequently, the practical convergence of the solver. Furthermore, the initial results were extended to deal with tracking problems.

- **Accelerated versions of the stochastic proximal-gradient method with variance reduction.** We proposed two different acceleration strategies of the stochastic proximal-gradient (Prox-SVRG) method proposed in [136]. Prox-SVRG uses a multistage strategy that consists of two loops: an inner loop (the algorithm

behaves similarly to the stochastic proximal-gradient method) and an outer loop (the algorithm behaves similarly to a proximal-gradient method). First, we showed how to accelerate the inner loop of Prox-SVRG by exploiting its similarity to the inexact proximal-gradient method. Second, we showed how to accelerate the outer loop of the Prox-SVRG by using in the proof classical stability arguments based on dynamical systems theory.

- **Stochastic alternating minimization algorithm with variance reduction.** We proposed a stochastic alternating minimization algorithm with variance reduction (SVR-AMA) suitable for MPC applications. In particular, the proposed algorithm follows from the application of Prox-SVRG to the dual of the MPC problem. Furthermore, we also proposed two accelerated versions of SVR-AMA following a similar approach to the one described above for Prox-SVRG.

- **Numerical results on aerospace applications.** All the proposed methods were tested using aerospace applications (longitudinal flight control problems) to evaluate the performance of the proposed solvers and control algorithms when applied to these systems. The proposed solvers led to significant improvements in terms of quality of the MPC problem solution achievable within the sampling time of the aircraft. Furthermore, the proposed control algorithms guaranteed closed-loop stability and recursive feasibility.

- **Active diagnosis strategy for jamming faults.** We designed a fault detection and diagnosis algorithm that strongly relies on reconfigurable MPC to handle elevator jamming faults. The diagnosis of the fault is active in the sense that the fault detection (FD) unit is able to diagnose the root cause of the jamming by using the information provided by the MPC controller. In this respect, after the FD unit communicates the occurrence of a fault, the MPC controller is commanded to perform a tailored reconfiguration that provides the FD unit with the necessary information for the diagnosis. Without these interactions, passive diagnosis methods alone would not be able to diagnose the root cause of the jamming faults.

- **Fault-tolerant reference generation.** In order to help the FD unit diagnose the root cause of jamming faults, we proposed the use of an artificial reference for the state and the control commands that takes into account the occurrence of faults. This strategy helps deal with changes in the feasible region of the controller, by generating, for each problem instance, a reference signal that is feasible for the available control command.

## Future Directions

1. **Distributed MPC using SVR-AMA.** The proposed SVR-AMA scheme can be a suitable candidate to deal with large-scale distributed MPC applications. In

particular, it would be interesting to investigate the possibility of using the probability distribution of the random updates as a tuning parameter to design the interactions among different agents in the network.

2. **Constrained linear quadratic regulator problem.** The presence of the terminal set to guarantee closed-loop stability in MPC often leads to conservative performance. In addition, the terminal set has usually (a negative) impact on the conditioning of the optimization problem leading to poor performance of the first-order solver (in terms of number of iterations needed to reach a desired level of suboptimality). Hence, it would be interesting to reconsider the constrained linear quadratic regulator problem. Promising results in this direction are provided in [122; 25].

3. **Implementation on embedded systems.** Several studies focused on the issues related to the implementation of solvers for embedded MPC applications ([53; 145; 38; 92] to mention a few). Most of the algorithms proposed in this dissertation have been implemented in MATLAB (with a few exceptions in C) using floating-point architectures. It would be interesting to investigate how these algorithms (especially SVR-AMA) perform on fixed-point units. Furthermore, it would be interesting to implement the proposed designs using the Airbus library, which is a formally verified programming language used to implement the control laws on the on-board control unit and only supports a limited number of operations.

4. **Data-driven designs.** From the control perspective, accurate models are important for MPC to guarantee good performance. For flight control applications, we have to take into account that the aircraft dynamics can vary significantly throughout the flight envelope and are influenced by several parameters and flight modes, such as control surface configurations, landing gear, the center of gravity, the weight of the aircraft or weather conditions (e.g., presence of icing), to mention a few. All these parameters have a significant impact on the dynamics of the aircraft that cannot be captured by a single LTI model. In practice, there is typically a lack of availability of sets of linearized models (corresponding to operating points sampled sufficiently densely from the flight envelope) corresponding to all parameter configurations. This motivates the study of alternative approaches based on data-driven online model adaptation and control design, such as moving horizon estimation (MHE) [102; 45] or subspace predictive control (SPC) problems [18]. In this respect, there are already some studies on SPC for aerospace applications (for example, in [40]), but the work is still limited due to the computation requirements of this technique.

5. **Linking control theory and optimization.** In the proof of the outer acceleration of Prox-SVRG we relied on dynamical systems theory to derive conclusions on

the convergence of the solver. Thanks to this theory we could provide a simple proof of convergence of the algorithm. There is an emerging trend in applying control theory to optimization [127; 133; 132; 41]. In this respect, we can investigate the use of control theory to revise the analysis and improve the convergence of the first-order solvers introduced in this dissertation.

6. **Application domains.** The proposed algorithms have been tested on aerospace applications, motivated by the benchmark model of the RECONFIGURE project. The proposed techniques can be used in other application domains with hard real-time constraints, such as automotive or maritime applications. For example, the use of splitting algorithms is still limited for motion planning and control of autonomous cars or vessels. The proposed algorithms can be of great use to develop distributed real-time planners to handle dynamic environments and avoid collisions.

# References


[1] F. A. De Almeida and D. Leißling. Fault-tolerant model predictive control with flight-test results. *Journal of Guidance, Control, and Dynamics*, 33: 363–375, 2010.

[2] R.A. Bartlett, A. Wachter, and L.T. Biegler. Active set vs. interior point strategies for model predictive control. In *Proceedings of the American Control Conference*, pages 4229–4233, 2000.

[3] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009. DOI:10.1137/080716542.

[4] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002. DOI:10.1016/S0005-1098(01)00174-1.

[5] D.P. Bertsekas and J.N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, Englewood Cliffs, NJ, 1989.

[6] F. Borrelli, A. Bemporad, and M. Morari. *Predictive Control for linear and hybrid systems*. 2015. Available at http://www.mpc.berkeley.edu/mpc-course-material.

[7] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[8] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011. DOI:10.1561/2200000016.

[9] E. Camponogara and H.F. Scherer. Distributed optimization for model predictive control of linear dynamic networks with control-input and output constraints. *IEEE Transactions on Automation Science and Engineering*, 8(1): 233–242, 2011.

[10] K. P. B. Chandra, L. Chen, H. Alwi, and C. Edwards. Actuator faults and blow-down limit detection, and fault tolerant control for the reconfigure benchmark problem. In *2016 IEEE Conference on Control Applications (CCA)*, pages 1544–1549, 2016. DOI:10.1109/CCA.2016.7588020.

[11] J. Cieslak, D. Henry, A. Zolghadri, and P. Goupil. Development of an active fault-tolerant flight control strategy. *Journal of Guidance, Control, and Dynamics*, 31(1):135–147, 2008. DOI:10.2514/1.30551.

[12] C. Conte, T. Summers, M. N. Zeilinger, M. Morari, and C. N. Jones. Computational aspects of distributed optimization in model predictive control. In *Proceedings of 51th IEEE Conference on Decision and Control(CDC)*, pages 6819–6824, December 2012.

[13] M. D. Doan, T. Keviczky, and B. De Schutter. A dual decomposition-based optimization method with guaranteed primal feasibility for hierarchical MPC problems. In *Proceedings of the 18th IFAC World Congress*, pages 392–397, September 2011.

[14] M.D. Doan, T. Keviczky, and B. De Schutter. A distributed optimization-based approach for hierarchical model predictive control of large-scale systems with coupled dynamics and constraints. In *Proceedings of the 50th IEEE Conference on Decision and Control*, pages 5236–5241.

[15] C. Edwards, H. Smaili, and T. Lombaerts. *Fault Tolerant Flight Control: A Benchmark Challenge*. Springer-Verlag, 2010.

[16] P. Falcone, M. Tufo, F. Borrelli, J. Asgari, and H. E. Tseng. A linear time varying model predictive control approach to the integrated vehicle dynamics control problem in autonomous systems. In *Proceedings of the 46th IEEE Conference on Decision and Control*, pages 2980–2985. IEEE, 2007.

[17] D. Famularo, Martino D., and M. Mattei. Constrained control strategies to improve safety and comfort on aircraft. *AIAA Journal of Guidance, Control, and Dynamics*, 31(6):1782 – 1792, 2008.

[18] W. Favoreel and B. De Moor. SPC: Subspace predictive control. In *Proceedings of the IFAC World Congress*, pages 235–240, 1998.

[19] A. Ferramosca, D. Limón, I. Alvarado, T. Alamo, and E. F. Camacho. MPC for tracking with optimal closed-loop performance. *Automatica*, 45(8):1975–1978, 2009. DOI:10.1016/j.automatica.2009.04.007.

[20] L. Ferranti and T. Keviczky. A parallel dual fast gradient method for MPC applications. In *Proceedings of the 54th IEEE Control Conference on Decision and Control*, pages 2406–2413, Dec 2015. DOI:10.1109/CDC.2015.7402568.

[21] L. Ferranti and T. Keviczky. MPC design for the longitudinal motion of a passenger aircraft based on operator-splitting and fast-gradient methods. In *Proceedings of the European Control Conference (ECC'16)*, pages 1562–1567, Jun 2016.

[22] L. Ferranti and T. Keviczky. Operator-splitting and gradient methods for real-time predictive flight control design. *Journal of Guidance, Control, and Dynamics*, pages 1–13, 2016. DOI:10.2514/1.G000288.

[23] L. Ferranti, Y. Wan, and T. Keviczky. Predictive flight control with active diagnosis and reconfiguration for actuator jamming. In *Proceedings of 5th IFAC Conference on Nonlinear Model Predictive Control*, pages 166–171, September 2015. DOI:10.1016/j.ifacol.2015.11.278.

[24] L. Ferranti, Y. Pu, C. N. Jones, and T. Keviczky. Asynchronous splitting design for model predictive control. In *Proceedings of the 55th IEEE Conference on Decision and Control*, 2016.

[25] L. Ferranti, G. Stathopoulos, C. N. Jones, and T. Keviczky. Constrained LQR using online decomposition techniques. In *55th IEEE Conference on Decision and Control (CDC)*, pages 2339–2344, Dec 2016.

[26] L. Ferranti, Y. Pu, C. N. Jones, and T. Keviczky. SVR-AMA: an asynchronous alternating minimization algorithm with variance reduction for model predictive control applications. *Submitted to the IEEE Transactions on Automatic Control*, 2017.

[27] L. Ferranti, Y. Wan, and T. Keviczky. Fault-tolerant reference generation for model predictive control with active diagnosis of elevator jamming faults. *Submitted for review to the International Journal of Robust and Nonlinear Control (IJRNC)*, 2017.

[28] H. J. Ferreau, H. G. Bock, and M. Diehl. An online active set strategy to overcome the limitations of explicit MPC. *International Journal of Robust and Nonlinear Control*, 18(8):816–830, 2008. DOI:10.1002/rnc.1251.

[29] C. E. Garcia and D. M. Prett. Advances in industrial model–predictive control. *Chemical Process Control*, 111:249–294, 1986.

[30] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson. Optimal parameter selection for the alternating direction method of multipliers (ADMM): quadratic problems. *IEEE Transactions on Automatic Control*, 60(3):644–658, 2015.

[31] P. Giselsson. Improving Fast Dual Ascent for MPC-Part II: the Embedded

Case. *arXiv preprint arXiv:1312.3013*, 2013.

[32] P. Giselsson and A. Rantzer. On feasibility, stability and performance in distributed model predictive control. *IEEE Transactions on Automatic Control*, 59(4):1031–1036, 2014.

[33] P. Giselsson, M. D. Doan, T. Keviczky, B. De Schutter, and A. Rantzer. Accelerated gradient methods and dual decomposition in distributed model predictive control. *Automatica*, 49(3):829–833, 2013. DOI:10.1016/j.automatica.2013.01.009.

[34] A. A. Goldstein. On steepest descent. *SIAM Journal on Control and Optimization*, 3(1):147, 1965. DOI:10.1137/0303013.

[35] T. Goldstein, B. O'Donoghue, S. Setzer, and R. Baraniuk. Fast alternating direction optimization methods. *SIAM Journal on Imaging Sciences*, 7(3): 1588–1623, 2014. DOI:10.1137/120896219.

[36] P. Goupil, J. Boada-Bauxell, A. Marcos, E. Cortet, M. Kerr, and H. Costa. AIRBUS efforts towards advanced real-time fault diagnosis and fault tolerant control. In *Proceedings of the 19th IFAC World Congress*, pages 3471–3476, 2014.

[37] P. Goupil, J. Boada-Bauxell, A. Marcos, P. Rosa, M. Kerr, and L. Dalbies. An overview of the FP7 RECONFIGURE project: industrial, scientific and technological objectives. In *Proceedings of the 9th IFAC Symposium on SAFEPROCESS*, pages 976–981, 2015.

[38] A. Guiggiani, I. Kolmanovsky, P. Patrinos, and A. Bemporad. Fixed-point constrained model predictive control of spacecraft attitude. In *Proceedings of the American Control Conference (ACC'15)*, pages 2317–2322, 2015. DOI:10.1109/ACC.2015.7171078.

[39] S. Haghighat, H.H.T. Liu, and J.R.R.A. Martins. Model-predictive gust load alleviation controller for a highly flexible aircraft. *AIAA Journal of Guidance, Control, and Dynamics*, 35(6):1751–1766, 2012.

[40] R. Hallouzi and M. Verhaegen. Fault-tolerant subspace predictive control applied to a boeing 747 model. *Journal of Guidance, Control, and Dynamics (JGCD)*, 31(4):873–88, 2008. DOI:10.2514/1.33256.

[41] M. Hardt, T. Ma, and B. Recht. Gradient descent learns linear dynamical systems. *arXiv preprint arXiv:1609.05191*, 2016.

[42] E. N. Hartley, M. Gallieri, and J. M. Maciejowski. Terminal spacecraft rendezvous and capture with lasso model predictive control. *International Journal of Control*, 86(11):2104–2113, 2013. DOI:10.1080/00207179.2013.789608.

[43] E. N. Hartley, J. L. Jerez, A. Suardi, J. M. Maciejowski, E. C. Kerrigan, and G. A. Constantinides. Predictive control using an FPGA with application to aircraft control. *IEEE Transactions on Control Systems Technology*, 22(3): 1006–1017, May 2014. DOI:10.1109/TCST.2013.2271791.

[44] E.N. Hartley and J.M. Maciejowski. A longitudinal flight control law based on robust MPC and H2 methods to accommodate sensor loss in the reconfigure benchmark. *9th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes (SAFEPROCESS)*, 48(21):1000–1005, 2015. DOI:10.1016/j.ifacol.2015.09.657.

[45] E. L. Haseltine and J. B. Rawlings. Critical evaluation of extended kalman filtering and moving-horizon estimation. *Industrial & Engineering Chemistry Research*, 44(8):2451–2460, 2005.

[46] B. He and X. Yuan. On the $O(1/n)$ convergence rate of the Douglas-Rachford Alternating Direction Method. *SIAM Journal on Numerical Analysis*, 50(2): 700–709, 2012. DOI:10.1137/110836936.

[47] M. Herceg, M. Kvasnica, C.N. Jones, and M. Morari. Multi-Parametric Toolbox 3.0. In *Proceedings of the European Control Conference*, pages 502–510, Zürich, Switzerland, July 17–19 2013. `http://control.ee.ethz.ch/~mpt`.

[48] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms*. Springer-Verlag Berlin Heidelberg, 1993. DOI:10.1007/978-3-662-02796-7.

[49] B. Houska, H.J. Ferreau, and M. Diehl. An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range. *Automatica*, 2011.

[50] D. Hrovat, S. Di Cairano, H. E. Tseng, and I. V. Kolmanovsky. The development of model predictive control in automotive industry: A survey. In *IEEE International Conference on Control Applications*, pages 295–302, October 2012. DOI:10.1109/CCA.2012.6402735.

[51] T. Zhang J. Langford, L. Li. Sparse online learning via truncated gradient. In *Proceedings of Advances in Neural Information Processing Systems*, pages 905–912, 2009.

[52] J. L. Jerez, G. A. Constantinides, E. C. Kerrigan, and K. V. Ling. Parallel MPC for real-time FPGA-based implementation. *Proceedings of the 18th IFAC World Congress*, 18(Part 1):1338–1343, 2011.

[53] J. L. Jerez, P. J. Goulart, S. Richter, G. A. Constantinides, E. C. Kerrigan, and M. Morari. Embedded predictive control on an FPGA using the fast gradient method. In *Proceedings of the European Control Conference (ECC'13)*, pages 3614–3620, 2013.

[54] J.L. Jerez, K.-V. Ling, G.A. Constantinides, and E.C. Kerrigan. Model predictive control for deeply pipelined field-programmable gate array implementation: Algorithms and circuitry. *IET Control Theory Applications*, 6:1029–1041, 2012.

[55] R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in Neural Information Processing Systems*, 26:315–323, 2013.

[56] M. M. Kale and A. J. Chipperfield. Stabilized MPC formulations for robust reconfigurable flight control. *Control Engineering Practice*, 13:771–788, 2005.

[57] T. Keviczky and G. J. Balas. Receding horizon control of an F-16 aircraft: a comparative study. *Control Engineering Practice*, 14(9):1023–1033, 2006. DOI:10.1016/j.conengprac.2005.06.003.

[58] T. Keviczky and G.J. Balas. Flight test of a receding horizon controller for autonomous uav guidance. In *Proceedings of the American Control Conference*, pages 3518–3523, 2005.

[59] T. Keviczky and G.J. Balas. Software-enabled receding horizon control for autonomous UAV guidance. *AIAA Journal of Guidance, Control, and Dynamics*, 29(3):680–694, 2006.

[60] M. Kögel and R. Findeisen. Fast predictive control of linear systems combining Nesterov's gradient method and the method of multipliers. In *Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, pages 501–506, 2011. DOI:10.1109/CDC.2011.6160688.

[61] M. Kögel and R. Findeisen. A fast gradient method for embedded linear predictive control. In *Proceedings of the 18th IFAC World Congress*, pages 1362–1367, 2011.

[62] M. Kögel and R. Findeisen. Cooperative distributed MPC using the alternating direction multiplier method. In *Proceedings of the 8th IFAC Symposium on Advanced Control of Chemical Processes*, pages 445–450, July 2012.

[63] Markus Kögel and Rolf Findeisen. Parallel solution of model predictive control using the alternating direction multiplier method. *Proceedings of the 4th IFAC Nonlinear Model Predictive Control Conference (NMPC)*, pages 369–374, 2012.

[64] W. Langson, I. Chryssochoos, S.V. Raković, and D. Q. Mayne. Robust model predictive control using tubes. *Automatica*, 40(1):125–133, 2004.

[65] J. Lew. Robust predictive control for structures under damage condition. *Journal of Guidance, Control, and Dynamics*, 36:1824–1829, 2013.

[66] D. Limón, I. Alvarado, T. Alamo, and E. F. Camacho. MPC for tracking of piece-wise constant references for constrained linear systems. *Automatica*, 44:2382–2387, 2008. DOI:10.1016/j.automatica.2008.01.023.

[67] D. Luenberger. Observers for multivariable systems. *IEEE Transactions on Automatic Control*, 11(2):190–197, 1966. DOI:10.1109/TAC.1966.1098323.

[68] J. M. Maciejowski. The implicit daisy-chaining property of constrained predictive control. *Applied Mathematics and Computer Science*, 8:695–712, 1998.

[69] J. M. Maciejowski. *Predictive Control: with Constraints*. Pearson Education, 2002.

[70] J. M. Maciejowski and C. N. Jones. MPC fault-tolerant flight control case study: Flight 1862. In *Proceedings of IFAC Safeprocess Sympoisum*, pages 119–124, 2003.

[71] A. Marcos and G. J. Balas. Development of linear-parameter-varying models for aircraft. *Journal of Guidance, Control, and Dynamics*, 27(2):218–228, 2004.

[72] D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000. DOI:10.1016/S0005-1098(99)00214-9.

[73] S. Mehrotra. On the Implementation of a Primal-Dual Interior Point Method. *SIAM Journal on Optimization*, 2(4):575–601, 1992.

[74] I. Necoara and D. Clipici. Efficient parallel coordinate descent algorithm for convex optimization problems with separable constraints: application to distributed MPC. *Journal of Process Control*, 23(3):243–253, 2013.

[75] I. Necoara and V. Nedelcu. Rate analysis of inexact dual first order methods. application to dual decomposition. *IEEE Transactions on Automatic Control*, 59(5):1232–1243, 2014. DOI: 10.1109/TAC.2013.2294614.

[76] I. Necoara and J.A.K. Suykens. Application of a smoothing technique to decomposition in convex optimization. *IEEE Transactions on Automatic Control*, 53(11):2674–2679, 2008.

[77] I. Necoara, V. Nedelcu, and I. Dumitrache. Parallel and distributed optimization methods for estimation and control in networks. *Journal of Process Control*, 21:756–766, 2011.

[78] I. Necoara, L. Ferranti, and T. Keviczky. An adaptive constraint tightening approach to linear model predictive control based on approximation algorithms for optimization. *Optimal Control Applications and Methods*, 36(5):648–666, 2015. ISSN 1099-1514. DOI:10.1002/oca.2121.

[79] V. Nedelcu, I. Necoara, and Q. Tran-Dinh. Computational complexity of inexact gradient augmented Lagrangian methods: application to constrained MPC. *SIAM Journal on Control and Optimization*, 52(5):3109–3134, 2014.

[80] A. Nedic and A. Ozdaglar. Approximate primal solutions and rate analysis for dual subgradient methods. *SIAM Journal on Optimization*, 19(4):1757–1780, 2009. DOI:10.1137/070708111.

[81] Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

[82] Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2005. DOI:10.1007/s10107-004-0552-5.

[83] V. Nevistic and J. A. Primbs. Finite receding horizon linear quadratic control: A unifying theory for stability and performance analysis. Technical report, California Institute of Technology, Pasadena, CA, 1997.

[84] A. Nitanda. Stochastic proximal gradient descent with acceleration techniques. In *Proceedings of Advances in Neural Information Processing Systems*, pages 1574–1582, 2014.

[85] I. Notarnicola and G. Notarstefano. Randomized dual proximal gradient for large-scale distributed optimization. In *Proceedings of the 54th IEEE Conference on Decision and Control*, pages 712–717, 2015.

[86] B. O'Donoghue and E. Candès. Adaptive restart for accelerated gradient schemes. *Foundations of Computational Mathematics*, 15(3):715–732, 2015. DOI:10.1007/s10208-013-9150-3.

[87] D. Ossmann. Fault tolerant control design for the longitudinal aircraft dynamics using quantitative feedback theory. In *AIAA Guidance, Navigation, and Control Conference*, pages 1–16, 2015. DOI:10.2514/6.2015-1310.

[88] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239, 2014. DOI:10.1561/2400000003.

[89] A. Patrascu, I. Necoara, and R. Findeisen. Rate of convergence analysis of a dual fast gradient method for general convex optimization. In *Proceedings of the 54th IEEE Conference on Decision and Control (CDC)*, pages 3311–3316, 2015. DOI:10.1109/CDC.2015.7402717.

[90] P. Patrinos and A. Bemporad. An accelerated dual gradient-projection algorithm for embedded linear model predictive control. *IEEE Transactions on Automatic Control*, 59(1):18–33, 2014. DOI:10.1109/TAC.2013.2275667.

[91] P. Patrinos and P. Bemporad. An accelerated dual gradient-projection algorithm for linear model predictive control. In *Proceedings of the 51st IEEE Conference on Decision and Control*, pages 662–667, 2012.

[92] P. Patrinos, A. Guiggiani, and A. Bemporad. Fixed-point dual gradient projection for embedded model predictive control. In *Proceedings of the European Control Conference (ECC'13)*, pages 3602–3607, 2013.

[93] T. Péni, B. Vanek, Z. Szabò, and J. Bokor. Supervisory fault tolerant control of the GTM UAV using LPV methods. *International Journal of Applied Mathematics and Computer Science*, 25(1):117–131, 2015. DOI:10.1515/amcs-2015-0009.

[94] I. Prodan, S. Olaru, R. Bencatel, J. B. de Sousa, C. Stoica, and S. I. Niculescu. Receding horizon flight control for trajectory tracking of autonomous aerial vehicles. *Control Engineering Practice*, 21(10):1334–1349, 2013. DOI:10.1016/j.conengprac.2013.05.010.

[95] A. I. Propoi. Use of linear programming methods for synthesizing sampled-data automatic systems. *Automation and Remote Control*, 24(7):837–844, 1963.

[96] Y. Pu, M. N. Zeilinger, and C. N. Jones. Fast alternating minimization algorithm for model predictive control. In *Proceedings of the 19th IFAC World Congress*, pages 11980–11986, 2014. DOI:10.3182/20140824-6-ZA-1003.01432.

[97] Y. Pu, C.N. Jones, and M.N. Zeilinger. Inexact alternating minimization algorithm for distributed optimization with an application to distributed MPC. Technical report, EPFL, August 2016. arXiv preprint arXiv:1608.00413.

[98] I. Puncochar, J. Siroky, and M. Simandl. Constrained active fault detection and control. *IEEE Transactions on Automatic Control*, 60:253–258, 2015.

[99] S.J. Qin and T.A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733–764, 2003.

[100] D. M. Raimondo, G. R. Marseglia, R. D. Braatz, and J. K. Scott. Fault-tolerant model predictive control with active fault isolation. In *Proceedings of 2013 Conference on Control and Fault-Tolerant Systems*, Nice, France, 2013.

[101] C. V. Rao, S. J. Wright, and J. B. Rawlings. Application of interior-point methods to model predictive control. *Journal of Optimization Theory and Applications*, 99(3):723–757, 1998. DOI:10.1023/A:1021711402723.

[102] J. B. Rawlings. Moving horizon estimation. *Encyclopedia of Systems and Control*, pages 1–7, 2014.

[103] J.B. Rawlings and D.Q. Mayne. *Model Predictive Control: Theory and Design*. Nob Hill Publishing, 2009.

[104] J. Richalet, A. Rault, J.L. Testud, and J. Papon. Model predictive heuristic control: Applications to industrial processes. *Automatica*, 14(1):413–428,

1978. DOI:10.1016/0005-1098(78)90001-8.

[105] A. Richards and J. How. Robust stable model predictive control with constraint tightening. In *Proceedings of the American Control Conference*, pages 1557–1562. IEEE, 2006.

[106] S. Richter, C.N. Jones, and M. Morari. Real-time input-constrained MPC using fast gradient methods. In *Proceedings of the 48th IEEE CDC held jointly with the 28th CCC*, pages 7387–7393, 2009. DOI:10.1109/CDC.2009.5400619.

[107] S. Richter, M. Morari, and C.N. Jones. Towards computational complexity certification for constrained MPC based on Lagrange relaxation and the fast gradient method. In *Proceedings of the 50th IEEE Conference on Decision and Control*, pages 5223–5229, 2011.

[108] S. Richter, C.N. Jones, and M. Morari. Computational complexity certification for real-time MPC with input constraints based on the fast gradient method. *IEEE Transactions on Automatic Control*, 57:1391–1403, 2012.

[109] S. Richter, C. N. Jones, and M. 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.

[110] P. Rosa, J. Vasconcelos, and M. Kerr. A mixed-$\mu$ approach to the integrated design of an FDI/FTC system applied to a high-fidelity industrial airbus nonlinear simulator. *9th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes (SAFEPROCESS)*, 48(21):988–993, 2015. DOI:10.1016/j.ifacol.2015.09.655.

[111] M. Rubagotti, P. Patrinos, and A. Bemporad. Stabilizing Embedded MPC with Computational Complexity Guarantees. In *Proceedings of the European Control Conference*, pages 3065–3070, July 2013.

[112] M. Rubagotti, P. Patrinos, and A. Bemporad. Stabilizing linear model predictive control under inexact numerical optimization. *IEEE Transactions on Automatic Control*, 59(6):1660–1666, 2014.

[113] R.S. Russell. Nonlinear F-16 simulation using simulink and matlab. Technical report, University of Minnesota, 2003. `https://www.aem.umn.edu/people/faculty/balas/darpa_sec/software/F16Manual.pdf`.

[114] R. Scattolini. Architectures for distributed and hierarchical model predictive control - a review. *Journal of Process Control*, 19(5):723–731, 2009.

[115] C.W. Scherer. LPV control and full block multipliers. *Automatica*, 37(3): 361–375, 2001. DOI:10.1016/S0005-1098(00)00176-X.

[116] M. Schmidt, N. Le Roux, and F. Bach. Convergence rates of inexact proximal-

gradient methods for convex optimization. In *Advances in neural information processing systems (NIPS)*, pages 1458–1466, 2011.

[117] P. O. M. Scokaert and J. B. Rawlings. Constrained linear quadratic regulation. *IEEE Transactions on Automatic Control*, 43(8):1163–1169, 1998.

[118] P.O.M. Scokaert, D.Q. Mayne, and J.B. Rawlings. Suboptimal model predictive control (feasibility implies stability). *IEEE Transactions on Automatic Control*, 44(3):648–654, 1999.

[119] S. Shalev-Shwartz and T. Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14:567–599, 2013.

[120] S. Shalev-Shwartz and T. Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. *Mathematical Programming*, 155:105–145, 2016.

[121] G. Stathopoulos, T. Keviczky, and Y. Wang. A hierarchical time-splitting approach for solving finite-time optimal control problems. In *Proceedings of the Eurpean Control Conference (ECC'13)*, pages 3089–3094. IEEE, 2013.

[122] G. Stathopoulos, M. Korda, and C. N. Jones. Solving the infinite-horizon constrained LQR problem using accelerated dual proximal methods. *IEEE Transactions on Automatic Control (TAC)*, 2016. DOI:10.1109/TAC.2016.2594381.

[123] G. Stathopoulos, H. Shukla, A. Szucs, Y. Pu, and C. N. Jones. Operator splitting methods in control. *Foundations and Trends® in Systems and Control*, 3 (3):249–362, 2016.

[124] B. Stevens, , and F. Lewis. *Aircraft Control and Simulation*. Wiley, New York, 1992.

[125] B.T. Stewart, A.N. Venkat, J.B. Rawlings, S.J. Wright, and G. Pannocchia. Cooperative distributed model predictive control. *Systems & Control Letters*, 59:460–469, 2010.

[126] F. Stoican and S. Olaru. *Set-theoretic Fault-tolerant Control in Multisensor Systems*. John Wiley & Sons, Inc., 2013.

[127] W. Su, S. Boyd, and E. Candès. A differential equation for modeling Nesterov's accelerated gradient method: Theory and insights. In *Advances in Neural Information Processing Systems*, pages 2510–2518, 2014.

[128] P. Tseng. Applications of a splitting algorithm to decomposition in convex programming and variational inequalities. *SIAM Journal on Control and Optimization*, 29(1):119—138, 1991. DOI:10.1137/0329006.

[129] F. Ullmann. A matlab toolbox for C-code generation for first order methods.

Technical report, ETH Zürich, 2011.

[130] M. Voss and R. Eigenmann. Reducing parallel overheads through dynamic serialization. In *Proceedings of the 13th IPDPS*, pages 88–92. IEEE, 1999. DOI:10.1109/IPPS.1999.760440.

[131] Y. Wang and S. Boyd. Fast Model Predictive Control Using Online Optimization. *IEEE Transactions on Control Systems Technology*, 18(2):267–278, 2010. DOI:10.1109/TCST.2009.2017934.

[132] A. Wibisono, A. C. Wilson, and M. I. Jordan. A variational perspective on accelerated methods in optimization. *Proceedings of the National Academy of Sciences*, 113(47):E7351–E7358, 2016. DOI:10.1073/pnas.1614734113.

[133] A. C Wilson, B. Recht, and M. I. Jordan. A lyapunov analysis of momentum methods in optimization. *arXiv preprint arXiv:1611.02635*, 2016.

[134] S. J. Wright. *Primal-Dual Interior-Point Methods*. SIAM Publications, 1997.

[135] S. J. Wright. Applying New Optimization Algorithms to Model Predictive Control. *AIChE Symposium Series*, 93(316):147–155, 1997.

[136] L. Xiao and T. Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014. DOI:10.1137/140961791.

[137] F. Xu, S. Olaru, V. Puig, C. Ocampo-Martínez, and S. Niculescu. Sensor-fault tolerance using robust MPC with set-based state estimation and active fault isolation. In *Proceedings of the 53rd Conference on Decision and Control*, pages 4953–4958, Los Angeles, CA, 2014.

[138] J.C. Duchi Y. Singer. Efficient learning using forward-backward splitting. In *Proceedings of Advances in Neural Information Processing Systems*, pages 495–503, 2009.

[139] A. Yetendje, M. M. Seron, and J. A. De Doná. Robust multiactuator fault-tolerant MPC design for constrained systems. *International Journal of Robust and Nonlinear Control*, 23:1828–1845, 2013.

[140] X. Yu, Z. Liu, and Y. Zhang. Fault-tolerant flight control design with finite-time adaptation under actuator stuck failures. *IEEE Transactions on Control Systems Technology*, 25(4):1431–1440, 2017. DOI:10.1109/TCST.2016.2603072.

[141] M. N. Zeilinger, N. J. Colin, D. M. Raimondo, and M. Morari. Real-time MPC-Stability through Robust MPC design. In *Proceedings of the 48th IEEE Conference on Decision and Control held jointly with the 28th Chinese Control Conference*, pages 3980–3986, 2009. DOI:10.1109/CDC.2009.5400903.

[142] Y. Zhang and J. Jiang. Fault tolerant control system design with

explicit consideration of performance degradation. *IEEE Transactions on Aerospace and Electronic Systems*, 39(3):838–848, 2003. DOI:10.1109/TAES.2003.1238740.

[143] Y. Zhang and J. Jiang. Issues on integration of fault diagnosis and reconfigurable control in active fault-tolerant control systems. *IFAC Proceedings Volumes*, 39(13):1437–1448, 2006.

[144] Y. Zhang and J. Jiang. Bibliographical review on reconfigurable fault-tolerant control systems. *Annual Reviews in Control*, 32(2):229 – 252, 2008. DOI:10.1016/j.arcontrol.2008.03.008.

[145] P. Zometa, M. Koegel, T. Faulwasser, and R. Findeisen. Implementation aspects of model predictive control for embedded systems. In *Proceedings of the American Control Conference*, pages 1205–1210, 2012.

# Curriculum Vitae

Laura Ferranti was born in Rome in 1986. She received her B.Sc. degree in Control Engineering from the University of Rome "Tor Vergata", Rome, Italy, in 2009, and her M.Sc. degree in Control Engineering from the University of Rome "Tor Vergata", Rome, Italy, in 2012. Her M.Sc. thesis project was carried out at ULg in collaboration with Schneider-Toshiba Inverter Europe, France. After her M.Sc. graduation, she was a visiting scholar at UC Berkeley, CA, in 2012. She conducted the research that led to this dissertation in the Delft Center for Systems and Control (DCSC), Delft University of Technology, Delft, The Netherlands, from 2013 to 2017. During her Ph.D. studies, she was a vising researcher at EPFL, Lausanne, Switzerland, at the beginning of 2016.

Her research interests include optimization and optimal control, model predictive control, embedded optimization-based control with application in flight control.

# Publications

This dissertation consists of a number of research papers, as published in the following journals (or as recently submitted for review) [1]:

I. Necoara, L. Ferranti, and T. Keviczky. An adaptive constraint tightening approach to linear model predictive control based on approximation algorithms for optimization[2]. *Journal of Optimal Control Applications and Methods (OCAM)*, DOI:10.1002/oca.2121, 2015.

L. Ferranti and T. Keviczky. Operator-splitting and gradient methods for real-time predictive flight control design[3]. *AIAA Journal of Guidance, Control, and Dynamics (JGCD)*, DOI:10.2514/1.G000288, 2016.

L. Ferranti, Y. Pu, C. N. Jones, and T. Keviczky. SVR-AMA: an asynchronous alternating minimization algorithm with variance reduction for model predictive control applications [4]. Submitted for review to the *IEEE Transactions on Automatic Control (TAC)*, 2017.

L. Ferranti, Y. Wan, and T. Keviczky. Fault-tolerant reference generation for model predictive control with active diagnosis of elevator jamming faults[5]. Submitted for review to the *International Journal of Robust and Nonlinear Control (IJRNC)*, 2017.

[1] The text may slightly differ from the original version in minor editorial changes made to improve readability.

[2] Appears in Chapter 2.

[3] Appears in Chapter 3.

[4] Appears in Chapter 4.

[5] Appears in Chapter 5.

Related publications not included in the dissertation are listed in the following:

L. Ferranti, Y. Wan, and T. Keviczky. Predictive flight control with active diagnosis and reconfiguration for actuator jamming In *Proceedings of the 5th IFAC Conference on Nonlinear Model Predictive Control (NMPC)*, vol. 48, n. 23, pp. 166-171, Seville, 2015.

L. Ferranti and T. Keviczky. A parallel dual fast gradient method for MPC applications. In *Proceedings of the 54th IEEE Conference on Decision and Control (CDC)*, pp. 2406-2413, Osaka, 2015.

L. Ferranti and T. Keviczky. MPC design for the longitudinal motion of a passenger aircraft based on operator-splitting and fast-gradient methods. In *Proceedings of the European Control Conference (ECC)*, Aalborg, 2016.

L. Ferranti, Y. Pu, C. N. Jones, and T. Keviczky. Asynchronous splitting design for model predictive control In *Proceedings of the 55th IEEE Conference on Decision and Control (CDC), Las Vegas, 2016.*

L. Ferranti, G. Stathopoulos, C. N. Jones, and T. Keviczky. Constrained LQR using online decomposition techniques In *Proceedings of the 55th IEEE Conference on Decision and Control (CDC), Las Vegas, 2016.*

X. Zhang, L. Ferranti, and T. Keviczky. An improved primal-dual interior-point solver for model predictive control *Accepted for publication in the proceedings of the the 56th IEEE Conference on Decision and Control (CDC), Melbourne, 2017.*