IMT Institute for Advanced Studies, Lucca

Lucca, Italy

Embedded Model Predictive Control: Finite Precision Arithmetic and Aerospace Applications

PhD Program in Computer Science and Engineering

XXVII Cycle

by

Alberto Guiggiani

2015

The dissertation of Alberto Guiggiani has been approved.

Program Coordinator: Prof. Alberto Bemporad,
IMT Institute for Advanced Studies, Lucca (IT).

Advisor: Prof. Alberto Bemporad,
IMT Institute for Advanced Studies, Lucca (IT).

Co-advisor: Prof. Panagiotis Patrinos,
IMT Institute for Advanced Studies, Lucca (IT).

Co-advisor: Prof. Ilya Kolmanovsky,
University of Michigan, Ann Arbor, MI (US).

The dissertation of Alberto Guiggiani has been reviewed by:

Prof. Ion Necoara,
University Politehnica of Bucharest (RO).

Dr. Samir Bennani,
European Space Agency (ESA).

IMT Institute for Advanced Studies, Lucca

2015

alla mia famiglia

Table of Contents

List of Figures

List of Tables

Acknowledgements

First of all, I would like to thank my advisors Alberto Bemporad, Panos Patrinos, and Ilya Kolmanovsky. Alberto, your skills and expertise have always been an inspiration, and your help for my career has been invaluable. Panos, most of this thesis results wouldn't have been possible without the passion and dedication that you put in your work. Ilya, I really appreciated your hospitality at the University of Michigan, you made my visiting period an enriching experience and gave an essential contribution to this thesis work.

Regarding the visiting period, I wish also to thank Eric Tseng and Davor Hrovat for hosting me at Ford Research in Dearborn.

I wish to express my gratitude to Ion Necoara and Samir Bennani for their efforts in reviewing this thesis, and for giving constructive feedbacks that helped me to improve this work.

I would like to thank all my friends and colleagues at IMT Lucca, with whom I had the pleasure to spend enjoyable moments through the three years of PhD. A special thanks goes to Matteo, Daniele, Pantelis, Lorenzo, Andreas, Gionata, Ajay, Carlo Alberto, and all the past and current Dynamical Systems, Control, and Optimization research unit fellows, that were always available for help and willing to engage in constructive discussions.

Un ringraziamento speciale alla mia famiglia per il supporto offerto in ogni istante di questo viaggio iniziato molti anni fa. Questa tesi è dedicata a voi.

Vita

Feb. 17, 1985 Born, Bagno a Ripoli (Florence), Italy

Education

2012-15 Ph.D., *IMT Institute for Advanced Studies*, Lucca.
Dynamical Systems, Control and Optimization Research Unit.
Advisor: Prof. Alberto Bemporad.

2005-11 B.Eng. + M.Eng., *University of Florence*, Florence (IT).
Electrical and Automation Engineering degree.
Final Mark: 110/110 cum Laude.
Thesis title: *Project and implementation of a real-time control system in an optical tweezer for neuronal stimulation measurements.*

Studies Abroad

2014 Visiting, *University of Michigan - Aerospace Engineering Dept.*, Ann Arbor (MI).

2008 Erasmus, *University of Lancaster*, Lancaster (UK).

Experience

2014 **Internship**, *Ford Research and Advanced Engineering*, Dearborn (MI).
Topic: Research on automotive control.

2012 **Collaborator**, *National Research Council of Italy (CNR) - Institute for Complex Systems (ISC)*, Florence (IT).
Topic: Development of a real-time control system for Atomic Force Microscopy (AFM) imaging.

2011 **Internship**, *Italian Institute of Technology (IIT) - Neuroscience and Brain Technologies Dept. (NBT)*, Genoa (IT).
Topic: Project and implementation of a control system for optical trap to study effects of mechanical stimuli on neuronal cells in vitro.

Projects

HYCON2 Matlab Toolbox, `dysco.imtlucca.it/h2t`.
An unified environment developed within the HYCON2 European Network of Excellence to promote and integrate MATLAB toolboxes for networked control design.

RealTime Suite, `rtaixml.net/realtime-suite`.
Tools and guides to set up a Linux-RTAI real-time machine within a CACSD environment.

Manuale di Automazione, `manualeautomazione.netsons.org`.
Notes (in italian language) covering many control theory and applications topics, including system modeling and identification, stability analysis, control systems synthesis, laboratories, and more.

Publications

Journal Papers

2015 M. Rubagotti, P. Patrinos, A. Guiggiani, A. Bemporad. "Real-Time Model Predictive Control Based on Dual Gradient Projection: Theory and Fixed-Point FPGA Implementation". *International Journal of Robust and Nonlinear Control*.

2015 P. Patrinos, A. Guiggiani, A. Bemporad. "A Dual Gradient Projection Algorithm for Model Predictive Control in Fixed-Point Arithmetic". *Automatica*, 55.

2011 A. Guiggiani, B. Torre, A. Contestabile, F. Benfenati, M. Basso, M. Vassalli, , and F. Difato. "Long-range and long-term interferometric tracking by static and dynamic force-clamp optical tweezers". *Optics Express*, 19(23).

Conference Proceedings

2015 A. Guiggiani, I. Kolmanovsky, P. Patrinos, and A. Bemporad. "Constrained Model Predictive Control of Spacecraft Attitude with Reaction Wheels Desaturation". In *Proc. European Control Conference*, Linz, Austria.

2015 A. Guiggiani, I. Kolmanovsky, P. Patrinos, and A. Bemporad. "Fixed-Point Constrained Model Predictive Control of Spacecraft Attitude". In *Proc. American Control Conference*, Chicago, Illinois.

2014 A. Guiggiani, P. Patrinos, and A. Bemporad. "Fixed-Point Implementation of a Proximal Newton Method for Embedded Model Predictive Control". In *Proc. IFAC World Congress*, Cape Town, South Africa.

2013 P. Patrinos, A. Guiggiani, and A. Bemporad. "Fixed-point dual gradient projection for embedded model predictive control". In *Proc. 12th European Control Conference*, Zurich, Switzerland.

2012 F. Difato, H. Tsushima, M. Pesce, A. Guiggiani, F. Benfenati, A. Blau, M. Basso, M. Vassalli, and E. Chieregatti. "Axonal regeneration of cultured mouse hippocampal neurons studied by an optical nano-surgery system". In *Proc. Conference on Photonic Therapeutics and Diagnostics*, San Francisco, California.

2011 A. Guiggiani, M. Basso, M. Vassalli, and F. Difato. "RealTime Suite: a step-by-step introduction to the world of real-time signal acquisition and conditioning". In Proc. Real Time Linux Workshop, Prague, Czech Republic.

Abstract

Model Predictive Control (MPC) is a multivariable advanced control technique widely popular in many industrial applications due to its ability to explicitly optimize performance, straightforwardly handling constraints on system variables.

However, MPC requires solving a Quadratic Programming (QP) optimization problem at each sampling step. This has slowed down its diffusion in embedded applications, in which fast sampling rates are paired with scarce computational capabilities, as in the automotive and aerospace industries.

This thesis proposes optimization techniques and controller formulations specifically tailored to embedded applications. First, fixed-point implementations of Dual Gradient Projection (DGP) and Proximal Newton methods are introduced. Detailed convergence analysis in the presence of round-off errors and algorithm optimizations are presented, and concrete guidelines for selecting the minimum number of fractional and integer bits that guarantee convergence are provided.

Moreover, extensive simulations and experimental tests on embedded devices, supported by general-purpose processing units and FPGAs, are reported to demonstrate the feasibility of the proposed solvers, and to expose the benefits of fixed-point arithmetic in terms of computation speeds and memory requirements.

Finally, an embedded MPC application to spacecraft attitude control with reaction wheels actuators is presented. A lightweight controller with specific optimizations is developed, and its good performance evaluated in simulations. Moreover, special MPC formulations that address the problem of reaction wheel desaturation are discussed, where the constraint handling property of MPC is exploited to achieve desaturation without the need of fuel-consuming devices such as thrusters.

Notation and Abbreviations

The notation and abbreviations adopted through the thesis can be resumed as follows.

Notation

$\mathbb{R}$	Set of real numbers.
$\mathbb{R}_+$	Set of nonnegative real numbers.
$\mathbb{R}^n$	Set of column real vectors of length n.
$\mathbb{R}^{m \times n}$	Set of m by n real matrices.
$\mathbf{I}^n$	Identity matrix of rank n.
$\mathbf{0}^{m \times n}$	m by n matrix of all zeros.
A'	Transpose of matrix A.
$[z]_+$	Euclidean projection of vector z on the nonnegative orthant.
$[z]_{\mathcal{Z}}$	Euclidean projection of vector z on the set $\mathcal{Z}$.
$\|z\|$	Euclidean norm of vector z.
$\|A\|$	Spectral norm of matrix A.
$\lambda(A)$	Set of eigenvalues of matrix A.
A^i	*i-th* column of matrix A.
z^i	*i-th* element of vector z.
$z_{(\nu)}$	Vector z at algorithm iteration ν.
$x(t)$	Dependency of vector x on continuous time instant t.
x_t	Dependency of vector x on discrete time step t.
$\texttt{fi}\,(\xi)$	Fixed-point representation of number ξ.

Abbreviations

ASIC	Application-Specific Integrated Circuit.
DGP	Dual Gradient Projection.
FPGA	Field-Programmable Gate Array.
GEVP	Generalized EigenValue Problem.
GP	Gradient Projection.
LQR	Linear Quadratic Regulator.
LVLH	Local-Vertical Local-Horizontal.
MIQP	Mixed-Integer Quadratic Programming/Program.
MPC	Model Predictive Control.
QP	Quadratic Programming/Program.
SDP	Semi-Definite Programming.

Chapter 1

Introduction

This chapter details the fundamental concepts behind Model Predictive Control (Section 1.1) and Quadratic Programming (Section 1.2), and surveys some of the key publications in those fields. Moreover, the basics of fixed-point computations are covered in Section 1.3. The issues and previous attempts in embedding MPC on hardware platforms, and the problem of achieving offset-free control, are detailed in Section 1.4 and Section 1.5, respectively. Section 1.6 presents existing literature on MPC for aerospace applications. Finally, Section 1.7 states the contributions given by this thesis.

1.1 Model Predictive Control

Model Predictive Control, also known as *Receding Horizon Control*, is a control strategy based on the knowledge of a plant model to predict future system states and compute an optimal input sequence. At each time step, a Quadratic Programming problem is solved in order to find the optimal control action which minimizes a combination between the actuation effort and the tracking error, over a fixed prediction horizon (Figure 1.1). Then, only the first computed input is fed to the system and the whole process is repeated at the subsequent time step, reinitializing the optimization problem with the new state measurement.

MPC controllers can be interfaced to plants described by dynamical systems with an arbitrary number of states and inputs, provided that sufficient computing power is available to evaluate the control input within the sampling period; in addition to that, they can inherently handle constraints on system states and inputs. This is a fact of uttermost relevance which greatly supported MPC diffusion in industries, since nearly every physical system is subject to constraints: actuators can exert forces in a limited range, system states can have security boundaries (e.g., temperature or pressure in a chemical plant), and so on. Moreover, MPC formulations can be modified and expanded to interact with a vast diversity of control problems, including nonlinear, stochastic or hybrid systems, large-scale and distributed scenarios, and embedded applications.

Model Predictive Control raised a significant interest in the

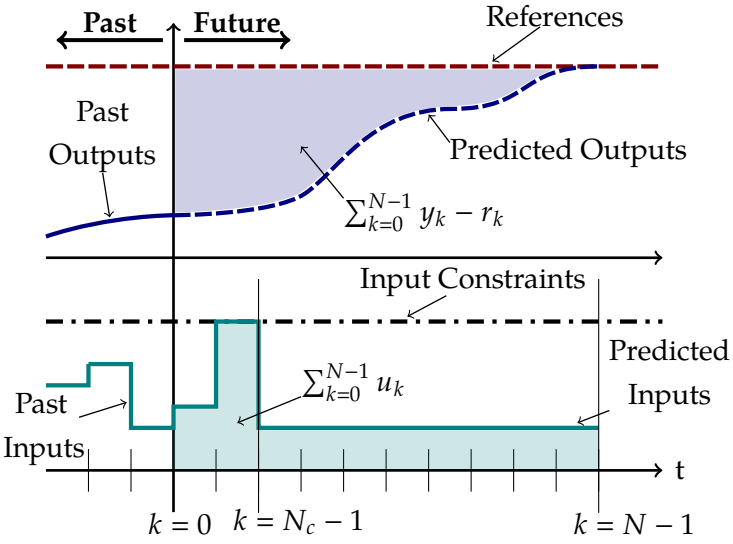

Figure 1.1: Graphical representation of Model Predictive Control. At current time step, a number N_c of future control moves u_k are computed such that the predicted system outputs y_k over N time steps are driven to desired reference signals r_k, while minimizing a weighted sum of the actuation effort and the tracking error (shaded areas).

control community for decades, and the literature is vast and elaborate. For classic reference books, refer to [1], [2], and [3]. Tutorial [4] introduces predictive control techniques for linear and nonlinear constrained system, with a non-expert target audience in mind. Notable surveys include [5], focused on research trends in stability and optimality, [6], investigating the diffusion of MPC as industrial technology, and [7], where also hybrid and explicit

setups are included.

The roots of the fundamental concepts underlying MPC can be found in the optimal control theories emerged in the "control revolution" of the 1960s. Indeed, the Model Predictive Control key idea links the two major research themes of that period.

The first comes from the *Dynamic Programming* theories of Hamilton, Jacobi and Bellman, which provided a way to evaluate an optimal *feedback* control in the form

$$u = K(x).$$

The second is the *maximum principle*, which allows the computation of an optimal *open-loop* control sequence

$$u^*(k, x) \text{ for } k = 0, 1, \ldots$$

The link is implicit in the Bellman's principle of optimality [8], which states:

> *"An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."*

More explicitly, the link becomes

$$K(x) = u^*(0, x)$$

as in Lee and Markus [9]:

"One technique for obtaining a feedback controller synthesis from knowledge of open-loop controllers is to measure the current control process state and then compute very rapidly for the open-loop control function. The first portion of this function is then used during a short time interval, after which a new measurement of the process state is made and a new open-loop control function is computed for the new measurement. The procedure is then repeated."

1.1.1 Linear Tracking Formulation

The basic linear Model Predictive Control formulation for reference tracking can be detailed as follows. Consider a physical system whose dynamics are described by a set of differential and output equation in the form

$$\begin{aligned} \dot{x}(t) &= f\left(x(t), u(t)\right), \\ y(t) &= g\left(x(t)\right), \end{aligned} \tag{1.1}$$

where $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ and $g : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ are (possibly nonlinear) functions, $x(t) \in \mathbb{R}^{n_x}$ and $u(t) \in \mathbb{R}^{n_u}$ are,respectively, the vectors of system states and control inputs at time t, and $y(t) \in \mathbb{R}^{n_y}$ is the vector of measured system outputs.

The control objective is to steer the system outputs to asymptotically track desired reference signals r, i.e.,

$$y(t) \to r(t) \text{ for } t \to \infty. \tag{1.2}$$

In order to be employed as prediction model in the MPC setup, the system dynamics (1.1) needs to be formulated in the discrete-time, linear state-space form

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t, \\ y_t &= Cx_t, \end{aligned} \tag{1.3}$$

Going from (1.1) to (1.3) is a critical step that often requires appropriate linearization or system identification procedures. The goal is to find a proper trade-off between *simplicity* and *exhaustivity*, i.e., the prediction model needs to be simple enough to allow for fast online computations, but complex enough to capture the significant plant dynamics.

The linear Model Predictive Control strategy for reference tracking can be summarized as follows.

1. At current time t, obtain the system output measurement y_t. Feed y_t to a state observer, generally based on a Kalman filter, to compute the current system state estimation $\hat{x}_t$.

2. Solve the following constrained optimization problem (1.4)

$$\min_{\left[\Delta u_0 \,..\, \Delta u_{N_c-1}\right]'} \|y_N - r_N\|^2_{W_N} + \sum_{k=0}^{N-1} \|y_k - r_k\|^2_{W_y} + \|\Delta u_k\|^2_{W_u}$$

$$\begin{aligned}
\text{subject to} \quad & x_{k+1} = Ax_k + Bu_k, \\
& y_k = Cx_k, \\
& u_k = u_{k-1} + \Delta u_k, \ k < N_c, \\
& u_k = u_{k-1}, \ k \geq N_c, \\
& u_{-1} = u_{t-1}, \ x_0 = \hat{x}_t, \\
& (y_k, u_k) \in \mathcal{Z},
\end{aligned}$$

where, over a prediction horizon of length N, a number N_c of free control moves are chosen to minimize a weighted sum of the tracking errors and the actuation efforts, with weights given by the matrices W_u, W_x, and W_N, while maintaining the input and output trajectories constrained in the set $\mathcal{Z} \in \mathbb{R}^{n_u+n_y}$. The computation of the prediction is initialized with the control action at the previous time step u_{t-1} and the current state estimate $\hat{x}_t$. The number of free control moves N_c is usually chosen significantly smaller than the prediction horizon N in order to reduce the computational complexity. The terminal cost $\|y_N - r_N\|_{W_N}$ is optional and generally used to prove closed-loop stability (see [10] for a seminal work in the context of predictive control of unconstrained linear systems, and [11, 12] for key extensions to the constrained case).

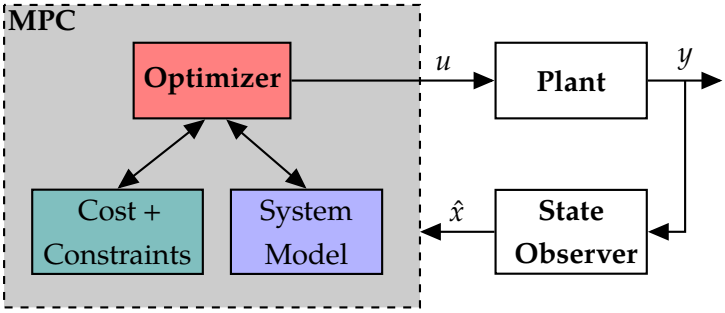

Figure 1.2: Typical Model Predictive Control loop. An optimization problem, built using the knowledge of the plant model, is solved to compute the control action u that minimizes a given performance index while satisfying actuator and state constraints. The problem is parametrized with respect of the current state estimate $\hat{x}$, obtained from a state observer.

3. Apply only the first optimal input u_0^* and discard the rest of the sequence. Then, repeat Steps 1-2 when the new system output measurement y_{t+1} is available.

A typical Model Predictive Control setup is depicted in Figure 1.2.

1.1.2 Extensions

Many modifications and extensions to the linear tracking formulation (1.4) have been proposed in the literature for specific control problems. Here follows a brief overview with some of the most significant.

Nonlinear MPC When performance specifications get tighter, safety or environmental considerations arise in importance, and system is driven close to constraints boundaries, can happen that the controller is required to take into account of process nonlinearities. Although the MPC framework outlined in this section has closed-loop nonlinear dynamics due to the presence of constraints, it is supported by a linear prediction model. On the other hand, a nonlinear MPC framework can be described as in (1.4), but with the state update equation is replaced by a nonlinear equation [13]. It now happens to be a constrained nonlinear parametric program, of which the solution may be hard (if not impossible) to compute online. In this case, an alternative approach is to linearize the system at each sampling step around the current operating point and build a new linear MPC problem, obtaining the so called time-varying MPC. For additional reading, refer to [14], [15], and [16].

Hybrid MPC The term Hybrid MPC refers to predictive control of system described by *hybrid models*. In hybrid models differential (difference) equations, for continuous (discrete) state evolutions, are possibly paired with switching dynamics, boolean inputs, finite state machines, and logic constraints. A framework for modeling (and controlling) hybrid systems was proposed in [17]; is based on describing the hybrid dynamics in term of *Mixed Logical Dynamical* (MLD) systems, which are computationally tractable as prediction models resulting in Mixed-Integer Quadratic Programming (MIQP). Stability of hybrid MPC formulations is investigated in [18].

Explicit MPC Explicit Model Predictive Control was first introduced in [19] (see also [20]) and has the objective of overcoming the online computational burden that comes in the optimization process by giving an *explicit* piecewise affine form for the controller, which can be computed offline by partitioning the input and state space. Then, online one has to evaluate the current region in the state space and compute an affine function. The key strength point of explicit MPC approach is the (generally) low computational effort required online at each sampling step. This is a fact of uttermost relevance for embedded MPC applications, where usually the controller must run on low-consumption, low-power devices. On the other hand, for larger problems the number of regions may explode, requiring a lot of memory to store the control laws and making the time required to evaluate the active region comparable to the time needed to solve online the optimization problem, thus neglecting the advantages of the explicit approach.

Robust MPC The basic linear MPC formulation (1.4) does not take into account of mismatches between the system model and the actual plant dynamics, and of any unmeasured disturbances acting on the system. In practical applications, both of those issues are present; this is the reason that motivated research in the *robust* MPC field. Preliminary investigations about robustness in model predictive control can be found in [21]; then, the main research direction was related to min-max optimal control problem, e.g., where an optimization of the worst-case scenario is performed. The biggest drawback of this approach is an exponential growth of the

complexity with respect to the prediction horizon; this issue was partially addressed in [22]. Additional key literature include [23], [24], and [25].

Stochastic MPC The robust approach is designed to address a worst-case scenario. However, for many applications where a strong and intrinsic stochasticity affects the model, robust control techniques may become too conservative. Sample applications belonging to this category include control of smart grids of multiple power generators and optimal interface to the energy market [26], vehicle adaptive cruise control with driver behavior modeling [27], networked control systems subject to time-varying transmission intervals, delays and dropouts [28]. The stochastic predictive control, instead of considering the worst-case disturbances, tries to embed a stochastic model of those uncertainties directly into the prediction model of the plant. This can be done by making the system matrices dependent on a set of discrete disturbances that vary with time according to proper dynamics (e.g., a Markov chain).

1.2 Quadratic Programming for Model Predictive Control

Quadratic Programming is a class of mathematical problems that falls into the category of convex optimization problems. The definition of a quadratic program relies on the concepts of *convex sets* and *convex functions*.

Definition 1.2.1 (Convex Set)**.** A set $C \subset \mathbb{R}^n$ is *convex* if, for any x_1, $x_2 \in C$, the set

$$\mathcal{P} = \lambda x_1 + (1 - \lambda)x_2, \quad \lambda \in [0, 1]$$

is is entirely contained in C. In other words, the line drawn between any two points of the set still belongs to the set.

Definition 1.2.2 (Convex Function)**.** A function $V : C \to \mathbb{R}$ is *convex* if the set C is convex and, for any $x_1, x_2 \in C$, the following inequality holds

$$V\left(\lambda x_1 + (1 - \lambda)x_2\right) \leq \lambda V(x_1) + (1 - \lambda)V(x_2), \quad \lambda \in [0, 1]\,.$$

In other words, the function always remains below the line traced between any two points of the function itself.

Definition 1.2.3 (Strongly Convex Function)**.** A function $V : C \to \mathbb{R}$ is *strongly convex* with modulus c if, for any $x_1, x_2 \in C$ and $\lambda \in [0, 1]$, it holds

$$V\left(\lambda x_1 + (1 - \lambda)x_2\right) \leq \lambda V(x_1) + (1 - \lambda)V(x_2) - c\lambda(1 - \lambda)(x_1 - x_2)^2. \tag{1.5}$$

For V differentiable, (1.5) can be rewritten as

$$(\nabla V(x_1) - \nabla V(x_2))'(x_1 - x_2) \geq \kappa_V \|x_1 - x_2\|^2,$$

where $\kappa_V = 2c$.

Definition 1.2.4 (Convex Optimization Problem)**.** A convex optimization problem is the problem of minimizing a convex function V with respect to an optimization vector z, which is constrained to belong to a convex set C:

$$\begin{aligned} \min_{z} \quad & V(z) \\ \text{subject to} \quad & z \in C. \end{aligned} \tag{1.6}$$

Problem (1.6) is also a *strongly* convex optimization problem if $V(z)$ is strongly convex.

Definition 1.2.5 (Quadratic Programming)**.** Quadratic Programming is a class of convex optimization problems in the form (1.6), where the cost function $V : C \to \mathbb{R}$ is quadratic with respect to the optimization vector $z \in \mathbb{R}^n$, i.e.,

$$V(z) = \frac{1}{2} z'Qz + c'z,$$

with $Q \in \mathbb{R}^{n \times n}$ (*Hessian* matrix) and $c \in \mathbb{R}^n$, and C is a polyhedron.

Literature on convex optimization is vast and covers over one century of research. However, the last 30 years have seen interest growing in this field due to two main reasons:

1. new algorithms and techniques were developed to solve efficiently even complex and large-scale problems;

2. applications in many and heterogeneous sciences were discovered to rely on convex problems; this includes predictive control, where a quadratic programming problem has to be solved at each sampling step.

For an excellent reference book on convex optimization theory and applications see [29].

1.2.1 Lagrangian Duality

Consider a convex optimization problem in the form

$$\begin{aligned} \mathbb{P}: \quad & \min_{z} \quad V(z) \\ & \text{subject to} \quad g(z) \leq 0 \end{aligned} \tag{1.7}$$

where $z \in \mathbb{R}^n$, $V : \mathbb{R}^n \to \mathbb{R}$ is a convex function, and $g(z) : \mathbb{R}^n \to \mathbb{R}^m$ describes a convex set. We call (1.7) the *primal* problem $\mathbb{P}$.

Definition 1.2.6 (Lagrangian Function)**.** Consider a constrained convex optimization problem as in (1.7). Its *Lagrangian function* $\mathcal{L}$ is defined as

$$\mathcal{L}(z, y) = V(z) + y' g(z),$$

where $y \in \mathbb{R}^m$ is a vector of *Lagrange multipliers*.

Using the Lagrangian function one can formulate a *dual* problem $\mathbb{D}$ as follows

$$\begin{aligned} \mathbb{D}: \quad & \max_{y} \quad \Phi(y) \\ & \text{subject to} \quad y \geq 0, \end{aligned} \tag{1.8}$$

where

$$\Phi(y) = \inf_{z \in \mathbb{R}^n} \mathcal{L}(z, y). \tag{1.9}$$

Intuitively, the dual problem is obtained by *relaxing* the primal constraints. Their violation is now permitted, but penalized in the cost function and weighted by positive coefficients (the Lagrange multipliers). A simple graphical representation of the Lagrangian duality is depicted in Figure 1.3.

Under the assumption that the primal problem (1.7) is convex and feasible, i.e., the set of solutions z^* is non-empty, then the primal and dual problem have the same optimal value of the cost function ($V^* = \Phi^*$). This is a key result of the Lagrangian duality theory, and is exploited by many algorithms that work on the dual problem to ultimately find a solution for the primal problem.

For further reading on the duality theory, we refer to [30–32].

1.2.2 Methods

The theoretical foundations of Quadratic Programming are dated back in the 1950s [33–35]; since then, many methods and algorithms have been investigated in the literature. Today, we can identify three

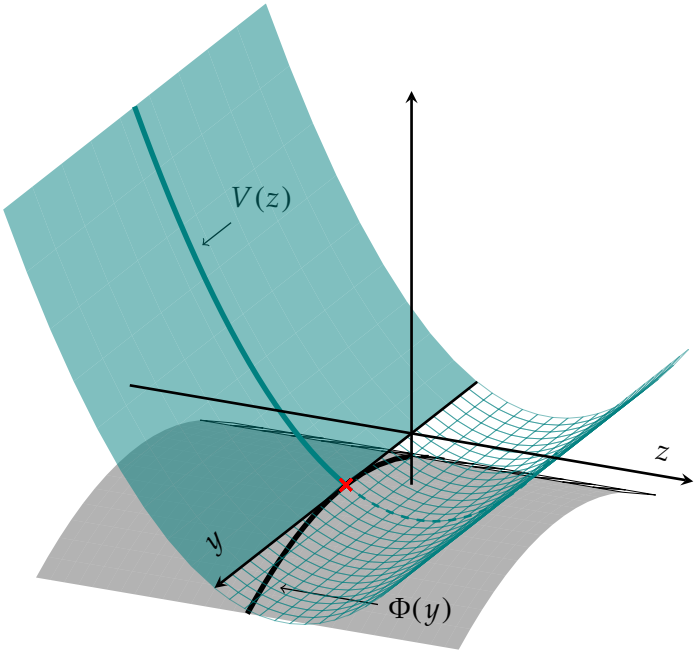

Figure 1.3: One-dimensional graphical example of Lagrangian duality. The primal problem (1.7) is formulated with $V(z) = z^2 - 2z$ and $g(z) = z$. The dual cost in (1.8) becomes $\Phi(y) = -\frac{1}{4}y^2 + y - 1$. Solid lines are cost functions inside constraints, dashed lines are cost functions outside constraints. Notice how the minimum of the primal cost function $V(z)$ coincides with the maximum of the dual cost function $\Phi(y)$ (denoted by a cross).

most popular classes of algorithms to solve problem arising in Model Predictive Control applications: *active set* methods, *interior point* methods, and *gradient projection* methods.

Active Set Methods The term *active set* refers to the subset of constraints that are satisfied as equalities by the current estimate of the optimization vector; this class of methods is named after this term since the computations performed at each iteration are based on updating the active set.

Active set methods are divided into *primal feasible* methods and *dual feasible* methods. In the primal approach, computation is divided into two phases. The first phase requires to find a feasible point for the primal, while in the second phase the objective function is decreased by updating a Karush-Kuhn-Tucker (KKT) matrix and the set of active constraints. Algorithm terminates when the current estimate is feasible for the dual problem.

Dual approaches work in a similar way, searching feasible points for the dual problem and increasing the objective function by a KKT matrix update. With respect to the primal feasible approaches, they generally require less computations to find a feasible point, since dual constraints are simpler (non-negativity constraints). On the other hand, they require that the primal problem is strongly convex.

Active-set general purpose solvers are adopted in MPC applications for small-to-medium scale problems, since problem complexity is $O(N^3)$, i.e., grows cubicly with respect to the prediction horizon.

Examples of active-set methods are found in literature since the early 1980s [36, 37]. For notable subsequent advancements and algorithm formulations see [38–40]. Finally, for an active set method specifically tailored for fast MPC applications refer to [41].

Interior Point Methods The roots of modern interior point methods are found in the *barrier* methods, popular during the 1960s to solve constrained programming [42].

Consider an optimization problem in the form (1.7). Barrier methods approximate it into an unconstrained problem by adding a barrier function to the cost function that penalizes the candidate solution for being infeasible. A sample barrier function is given by the logarithmic barrier; the resulting problem then becomes

$$\min_{z} \quad V(z) + \sum_{i=i}^{m} -\lambda^{-1} \log\left(-g_i(z)\right) \tag{1.10}$$

where λ is an user-defined parameter that regulates a trade-off between a good solution approximation ($\lambda \to \infty$) and a more numerically stable Hessian ($\lambda \to 0$). The approximated problem (1.10) can then be solved with Newton methods [43, 44].

Interest in barrier methods quickly faded when, in the late 1960s, it was shown that the Hessian of the barrier function can become increasingly ill-conditioned as the solution is approached [45, 46]. The mainstream optimization community then abandoned barrier methods in favor to more promising - for that time - approaches, like augmented Lagrangian and sequential quadratic programming.

Everything changed with the interior-point revolution of 1984, started with the Karmarkar announcement of a new linear programming method able to solve the problem in polynomial time [47]. Just one year later it was demonstrated that there was a formal equivalence between Karmarkar's algorithm and classical barrier methods [48], which then gained again popularity amongst the optimization community.

Subsequent works bridged classical barrier methods to modern optimization, extending polynomial-time complexity to new convex optimization problems [49]. Today, modern interior-points methods are routinely used in many applications, including Model Predictive Control [50]. Since problem complexity grows as $O(N)$, i.e., linearly with the prediction horizon, interior-point solvers are the standard for large-scale problems.

Gradient Projection Methods Gradient projection methods are a class of *first-order* methods, i.e., where only first-order informations of the function to be minimized, which include the function value and its gradient, are required. The first examples of gradient projection method are dated back in the late 1950s [51–53].

The simple idea behind this class of algorithms is the following. Consider a constrained optimization problem in the form (1.6), where the cost function $V(z)$ has a *Lipschitz-continuous* gradient with constant L, i.e.,

$$\|\nabla V(z_1) - \nabla V(z_2)\| \leq L\|z_1 - z_2\|,$$

for all $z_1, z_2 \in C$. Then, gradient projection algorithms in their basic implementation perform, at each iteration, a *descent* step and a *projection* step.

In the descent step, the current solution estimate is updated by performing a step in the direction of the negative function gradient:

$$z_{(\nu)} \leftarrow z_{(\nu)} - \lambda \nabla V(z), \tag{1.11}$$

where ν is the iteration counter and parameter λ has to be properly cosen to ensure convergence.

Then, in the projection step the solution estimate is projected on the constraint set, obtaining the new solution candidate

$$z_{(\nu+1)} = \left[z_{(\nu)}\right]_C, \tag{1.12}$$

which preserves feasibility. Note that gradient methods are of interest if computing the projection on the set C is *simple*.

If a solution accuracy ε is required for the objective function, algorithm (1.11)-(1.12) requires $O\left(\frac{L}{\varepsilon}\right)$ iterations. This speed bound was broken in Fast Gradient Methods (FGM), introduced in [54] and extended in [55–57], which exploit acceleration techniques based on variable step sizes and reach a solution accuracy ε in $O\left(\sqrt{\frac{L}{\varepsilon}}\right)$ iterations.

1.2.3 From Model Predictive Control to Quadratic Programming

Consider a Model Predictive Control problem in the regulation form

$$
\begin{aligned}
&\min_{\left[u_0 \,..\, u_{N_c-1}\right]'} \|x_N\|^2_{W_N} + \sum_{k=0}^{N-1} \|x_k\|^2_{W_x} + \|u_k\|^2_{W_u} \\
&\text{subject to} \quad x_{k+1} = Ax_k + Bu_k, \\
&\qquad\qquad\quad u_k = u_{k-1}, \quad k \geq N_c, \\
&\qquad\qquad\quad x_0 = \hat{x}_t, \\
&\qquad\qquad\quad \underline{x} \leq x_k \leq \bar{x}, \quad \underline{u} \leq u_k \leq \bar{u}.
\end{aligned}
\tag{1.13}
$$

The goal is to formulate problem (1.13) in a condensed quadratic programming problem in the form

$$
\begin{aligned}
&\min_{z} \quad \frac{1}{2} z'Qz + c'z \\
&\text{subject to} \quad Gz + g \leq 0,
\end{aligned}
\tag{1.14}
$$

where $z = \left[u_0 \,..\, u_{N_c-1}\right]'$ is the stacked vector of control inputs over the control horizon. This can be achieved by means of the following procedure.

1. Build the extended weight matrices

$$\mathcal{W}_x = \overbrace{\begin{bmatrix} W_x & 0 & \dots & 0 \\ 0 & \ddots & \dots & 0 \\ \vdots & \vdots & W_x & \vdots \\ 0 & 0 & \dots & W_N \end{bmatrix}}^{N\ blocks}, \quad \mathcal{W}_u = \overbrace{\begin{bmatrix} W_u & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & W_u \end{bmatrix}}^{N_u\ blocks}.$$

2. Build the state update matrices

$$\mathcal{A} = \begin{bmatrix} A' & (A^2)' & \dots & (A^N)' \end{bmatrix}'$$

$$\mathcal{B} = \overbrace{\begin{bmatrix} B & 0 & \dots & \dots & 0 \\ AB & B & \dots & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A^{N_u-2}B & A^{N_u-3}B & \dots & B & 0 \\ \vdots & \vdots & \dots & AB & B \\ \vdots & \vdots & \dots & \vdots & B\sum_{i=0}^{1} A^i \\ \vdots & \vdots & \dots & \vdots & \vdots \\ A^{N-1}B & A^{N-2} & \dots & A^{N-N_u+1}B & B\sum_{i=0}^{N-Nu} A^i \end{bmatrix}}^{N_u\ blocks}.$$

3. Build the constraints matrices

$$G = \begin{bmatrix} \mathbf{I}^{N_u n_u} \\ -\mathbf{I}^{N_u n_u} \\ \mathcal{B} \\ -\mathcal{B} \end{bmatrix}, \quad S = \begin{bmatrix} \mathbf{0}^{2N_u n_u \times n_x} \\ -\mathcal{A} \\ \mathcal{A} \end{bmatrix}, \quad w = \begin{bmatrix} \bar{U} \\ -\underline{U} \\ \bar{X} \\ -\underline{X} \end{bmatrix}$$

where $\bar{U}$, $\underline{U}$ (and $\bar{X}$, $\underline{X}$) are the stacked vectors of upper and lower bounds on inputs (states), repeated N_u (N) times, and $\mathbf{I}^{\eta}$ is the identity matrix of size η.

4. Finally, the QP problem (1.14) is defined by

$$\begin{aligned} Q &= 2\left(\mathcal{B}'\mathcal{W}_x\mathcal{B} + \mathcal{W}_u\right), \\ c &= 2\mathcal{B}'\mathcal{W}_x\mathcal{A}\hat{x}_t, \\ g &= -S\hat{x}_t - w. \end{aligned}$$

The procedure to build the QP problem starting from a tracking MPC setup, as in (1.4), is equivalent to the one detailed in this section, but applied to an extended system where the control inputs are included as states, i.e.,

$$\begin{cases} \begin{bmatrix} x_{t+1} \\ u_{t+1} \end{bmatrix} &= \begin{bmatrix} A & B \\ \mathbf{0}^{n_u \times n_x} & \mathbf{I}^{n_u} \end{bmatrix} \begin{bmatrix} x_t \\ u_t \end{bmatrix} + \begin{bmatrix} B \\ \mathbf{I}^{n_u} \end{bmatrix} \Delta u_t \\ y_t &= \begin{bmatrix} C & \mathbf{0}^{n_y \times n_u} \end{bmatrix} \begin{bmatrix} x_t \\ u_t \end{bmatrix}. \end{cases} \tag{1.15}$$

Table 1.1: Fixed-point vs. floating-point arithmetics.

Fixed-Point	**Floating-Point**
Pros	*Pros*
Faster computations.	Extended range.
Broader hardware support.	Enhanced precision.
Lower power consumption.	Higher flexibility.
Cons	*Cons*
Limited range and precision.	Slower computations.
Round-off errors.	Limited hardware support.
Overflow errors.	Larger power consumption.

1.3 Fixed-Point Computations

When performing computations on digital hardware, numbers are stored into *binary words*, sequences of bits (either one or zero) of fixed length. How these sequences are to be interpreted to get the *real-world* number depends on the data type; it can be either *fixed-point* or *floating-point*. A synthetic comparison of the two is presented in Table 1.1.

In the floating-point format, part of the available bits are reserved to the *exponent*, and part to the *mantissa* (significand), plus one sign bit. To retrieve the stored number it is sufficient to perform the operation

$$\text{real-world number} = \text{mantissa} \times 2^{\text{exponent}}.$$

Being able to control the exponent value, and therefore the scaling, enables a trade-off between range and precision. According to the

needs, floating-point format allows to represent very large values, or small values with high precision. For this reason, floating-point representations are the standard in the majority of computing applications.

Efficiently executing operation between floating-point numbers on digital calculators requires the processor to be equipped with a *floating-point unit* (FPU). While nearly all modern, desktop-class CPUs come equipped with hardware support for floating-point computations, this is not true for most of the microprocessors employed in embedded control applications, such as automotive and aerospace, and for programmable devices such as *Field-Programmable Gate Arrays* (FPGAs) and *Application-Specific Integrated Circuits* (ASICs). As demonstrated in [58], performing operations in floating-point arithmetic on such devices can lead to slower computations and larger power consumption with respect to fixed-point arithmetic.

When trying to minimize computational effort, power consumption, and chip size, a great positive impact is given by the choice of fixed-point number representation, especially in embedded applications [59].

Fixed-point data types, as depicted in Figure 1.4, are defined by:

- a total number w of bits (*word length*),
- r bits for the integer part,
- p bits for the fractional part,

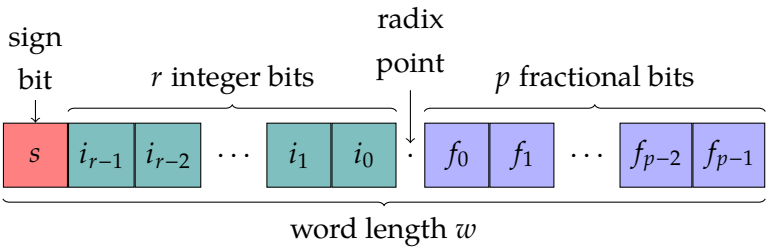

Figure 1.4: Structure of a fixed-point number. A total number w of bits (word length) is split between r bits for the integer part, p bits for the factional part, and one (optional) sign bit. Unlike floating-point number representations, the radix point is fixed.

- a fixed radix point, and
- an optional sign bit.

The real-world number can be restored from the fixed-point representation with the following operation:

$$\text{real-world value} = \text{stored value} \times 2^{-p}$$

Switching to fixed-point format in embedded applications can lead to a significant improvement in computational performance; however, this comes at the price of *overflow* errors and *round-off* errors (Figure 1.5).

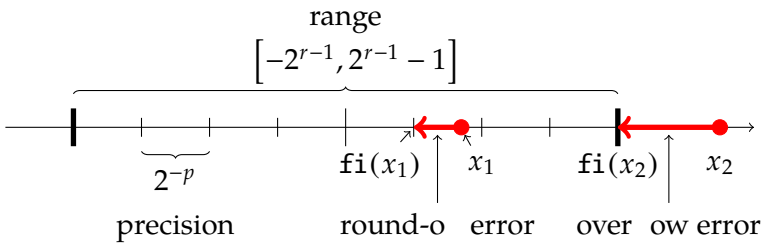

Figure 1.5: Fixed-point errors. Because of finite precision, real-world number x_1 is rounded to the representable value $\mathtt{fi}(x_1)$ committing a round-off error. Because of limited range, real-world number x_2, which lies out of admissible range, is stored as the maximum representable number $\mathtt{fi}(x_2)$, committing an overflow error.

1.3.1 Overflow Errors

A fixed-point data type can represent numbers in a limited range. For a signed fixed-point number, this corresponds to

$$\text{range} = \left[-2^{r-1}, 2^{r-1}-1\right],$$

where the asymmetry comes for the representation of the zero-element.

If the result of a mathematical operation is a number which lies outside this range, an overflow error occurs. Then, according to the architecture, the number can be projected to the maximum representable value (*saturation*), or cast back into the representable space (*modulo* arithmetic). In both cases it is in a dangerous situation that can lead to totally unexpected computation results. For this reason, it is of uttermost importance to ensure that the number r of

integer bits is large enough such that every operation output lies within the admissible range.

1.3.2 Round-off Errors

In fixed-point computations, alongside with limited range comes limited precision. The precision is the difference between successive values representable by fixed-point data type, which is equal to the value of its least significant bit. The precision is linked to the number of fractional bits, i.e.,

$$\text{precision} = 2^{-p}$$

If a number cannot be represented exactly by the specified data type and scaling, a rounding method is used to cast the value to a representable number and a round-off error occurs. This error becomes particularly dangerous in iterative algorithms, that can suffer of error accumulation. Embedding the error dynamics in iterative algorithms for Quadratic Programming is a challenging task and is one of the main contributions of Chapter 2.

An excellent reference book on fixed-point error analysis is [60], where round-off errors coming from fixed-point and floating-point number representations are analyzed, and their effect in many algebraic operations is investigated. Other notable literature on the topic can be found in [61–64].

1.3.3 Errors due to Mathematical Operations

Suppose that an algorithm is running with fixed-point arithmetic with a scaling factor 2^{-p}, i.e., with a number $p \in \mathbb{N}_+$ of fractional bits, and assume that real numbers are represented in fixed-point by rounding to the closest value. Therefore, the resolution (i.e., the smallest representable non-zero magnitude) of a fixed-point number is equal to $2^{-(p+1)}$.

Performing additions and subtractions between fixed-point numbers does not result in any loss of accuracy due to rounding.

On the other hand, the operation of multiplication can generate a round-off error in the result even if the operands are represented exactly. In specific, multiplying two fixed-point numbers $\zeta = \gamma\xi$ leads to the fixed-point representation $\mathtt{fi}(\zeta)$ of ζ, with

$$|\zeta - \mathtt{fi}(\zeta)| \leq 2^{-(p+1)}.$$

For $x, y \in \mathbb{R}^n$ let

$$\mathtt{fi}(x'y) \triangleq \sum_{i=1}^{n} \mathtt{fi}(x_i y_i).$$

Then, the round-off error for the inner product of x and y can be bounded as follows

$$|x'y - \mathtt{fi}(x'y)| \leq 2^{-(p+1)} n. \tag{1.16}$$

If A is an $m \times n$ matrix and x is an n-vector, then

$$\|Ax - \mathtt{fi}(Ax)\|_\infty \leq 2^{-(p+1)} n. \tag{1.17}$$

1.4 Embedded Model Predictive Control

In computing applications, the term *embedded systems* refers to small computers programmed to perform specific tasks within a larger system. With respect to general-purpose computers, embedded devices are characterized by lower power consumption and per-unit cost.

Embedded control applications usually distinguish for fast system dynamics, leading to high sampling frequencies and tight real-time constraints. At the same time, embedded controllers often dispose of scarce computing capabilities, e.g., low clock frequencies, limited memory, and lack of hardware support for floating-point computations.

For those reasons, implementing a Model Predictive Controller on embedded platforms poses quite a few challenges, both from a system-theoretic and an optimization point of view. Specifically, the main requirements that make a Quadratic Programming solver suitable for embedded MPC are the following.

1. The algorithm should be *simple* enough to be implemented on simple hardware platforms, for example requiring no divisions or dynamic memory allocation.

2. One must be able to compute a bound on its worst-case execution time for computing a (reasonably good) solution; this is a key requirement to guarantee that real-time constraints are satisfied.

3. Stability and invariance guarantees for the resulting closed-loop system must be provided despite a degree of suboptimality and/or infeasibility of the solution.

4. The algorithm should be robust to low precision arithmetic, i.e., the effect of round-off errors should be small (avoiding error accumulations between algorithm iterations), and no overflow should occur, by means on known bounds on all the computed variables. Moreover, one should be able to determine *a priori* the behavior of the algorithm under such hypotheses.

In recent years, significant interest amongst researchers and practitioners raised for extending advanced control techniques, such as Model Predictive Control, to fast embedded applications. This interest is supported and accentuated by two key factors:

1. new convex optimization algorithms and software tools which allow computing times for small-to-medium size problems in the range of milliseconds, even with relatively small computing capabilities, and

2. technological advancements in the production processes that brought embedded systems with faster microcontrollers and larger memory sizes, at lower per-unit cost and power consumption.

In [65] an FPGA implementation of an interior-point method for solving the QP problem is presented, showing that the "MPC-

on-a-chip" idea is indeed viable. In [66], an active-set QP solver for ASIC and FPGA is proposed and tested for Model Predictive Control of nonlinear systems. In [67, 68], a "QP-on-a-chip" controller is implemented on FPGA with an iterative linear solver and tested with hardware-in-the-loop experiments for aircraft control. In [69] an implementation of a modified interior-point solver in fixed-point is presented.

Recently, the use of first-order methods, and in particular Fast Gradient Methods developed by Nesterov [56], has been advocated as a viable candidate for embedded optimization-based control [70–77]. These methods can compute a suboptimal solution in a finite number of iterations, which can be bounded a priori, and they are simple enough (usually requiring only matrix-vector products) for hardware implementation. In particular, the accelerated Dual Gradient Projection method proposed in [71, 76], called GPAD, can be applied to linear MPC problems with general polyhedral constraints and with guaranteed global primal convergence rates. In [78] results of [76] are exploited to show how GPAD can be used in Model Predictive Control to provide invariance, stability and performance guarantees in a finite number of iterations for the closed-loop system.

Moreover, in [79] the authors derive convergence estimates for dual first order methods in general convex problems, under inexact dual information. The results of this paper are then extended in [80] to prove closed-loop stability in the context of embedded MPC.

1.5 Offset-Free Model Predictive Control

When Model Predictive Controllers are used for reference tracking, one fundamental issue is to deal with constant tracking errors that persist in steady-state.

The computation of the control action is based on a prediction model whose dynamics cannot fully capture the behavior of the real physical process. This might be due to modeling errors that affect the steady-state, such as in the presence of nonlinear DC gains, or of constant disturbances that represent the steady-state value of unmodeled dynamics affecting the controlled system.

An "integral action" is needed so that desired reference signals are tracked without a residual bias, even in the presence of model mismatches and unknown disturbances. The goal is to asymptotically track a vector $\bar{r}_t \in \mathbb{R}^{n_y}$ of desired output references, namely

$$y_t \to \bar{r}_t \text{ for } t \to \infty. \tag{1.18}$$

1.5.1 Methods for Offset-Free Model Predictive Control

In this section we review the most important methods for offset-free tracking in Model Predictive Control, i.e., modifications or extensions to (1.4) such that condition (1.18) is verified. Those classic methods include: augmenting the prediction model with states that integrate the tracking error; adopting disturbance models to be estimated together with system states; converting the prediction model to the so-called velocity form, where system states are updated in

incremental form. Moreover, a novel approach based on adding the integral of the tracking error to the reference signal is proposed.

Integral States A first method in MPC schemes to eliminate steady-state offsets in set-point tracking is to augment the dynamical model used for prediction with the integral of the tracking error. This technique, well-known for PID control, was proposed for state-feedback LQR control in [81], and can be successfully adopted in several state-feedback strategies such as MPC.

The formulation of an optimal, receding-horizon control problem that includes the explicit integration of the tracking error becomes the following:

(1.19)

$$\begin{aligned}
\min_{\left[\Delta u_0 \,..\, \Delta u_{N_c-1}\right]'} \quad & \|y_N - r_N\|^2_{W_N} + \sum_{k=0}^{N-1} \|y_k - r_k\|^2_{W_y} + \|\Delta u_k\|^2_{W_u} + \|i_k\|^2_{W_i} \\
\text{subject to} \quad & x_{k+1} = Ax_k + Bu_k, \\
& y_k = Cx_k, \\
& i_{k+1} = i_k + T_s\left(y_k - r_k\right), \\
& u_k = u_{k-1} + \Delta u_k,\ k < N_c, \\
& u_k = u_{k-1},\ k \geq N_c, \\
& u_{-1} = u_{t-1},\ x_0 = x_t, \\
& (x_k, u_k) \in \mathbb{Z}.
\end{aligned}$$

The term $i_k \in \mathbb{R}^{n_y}$ in (1.19) collects the additional states where integral of the tracking error is accumulated; those states also enter the cost function with a weight given by the matrix $W_i \in \mathbb{R}^{n_y \times n_y}$.

The relative values of W_u, W_y, W_i, that is the weights on input increments, outputs and integral of tracking errors, respectively, are the main tuning knobs to shape the closed-loop performance of the controller.

This method is simple to implement, as it requires only a slight modification of the cost function of the QP problem associated with (1.19). The additional on-line computation load introduced by the integral state i_k is very mild, as it only affects the computation of the linear term of the cost function, while it does not change the number of optimization variables and constraints in the QP, nor it affects the state observer.

The main drawback of the method, as in PID control, is potential *controller windup*. Controller windup is a well-know phenomenon that occurs when the required control action is larger than the actual control action fed to the plant due to the actuators saturation (e.g., in the transient right after a big step of the reference signal). This leads to an accumulation of the integral states, and eventually to large overshoots in the outputs and even instability [82]. A detailed review of anti-windup methods, along with a general framework to describe them, is presented in [83].

Within the proposed MPC framework, a large initial value i_0 in the prediction in (1.19) causes the term

$$\|i_k\|_{W_i} = i_k' W_i i_k = (i_0 + T_s \sum_{j=1}^{k-1} y_k - r_k)' W_i (i_0 + T_s \sum_{j=1}^{k-1} y_k - r_k)$$

where T_s is the sampling time, to create the following (possibly

large) linear term

$$2T_s i_0' W_i \sum_{j=1}^{k-1} y_k - r_k$$

in the cost function that is minimized with respect to Δu. Therefore, a large i_0 may bias the entire MPC function. Such a term i_0 may no be diminishable quickly because of limited actuation authority.

Extensions of this method for output feedback and tracking of nonlinear, possibly continuous time systems are discussed in [84, 85].

Disturbance Models One of the most popular and effective methods for achieving offset-free model predictive control involves augmenting the model with artificial, a-priori unknown, constant disturbances that are estimated together with the system states by an observer.

Seminal works on the topic include [86–88]; then, numerous extensions and refinements can be found in the literature. In [89] robust constraint satisfaction is ensured in offset-free control in presence of time-varying setpoints. A method for combined disturbance model and observer design is discussed in [90]. [91] analyzes the case where then number of measured variables is larger than the number of tracked variables. Finally, an extension to general nonlinear models is discussed in [92].

The method can be implemented by means of the following procedure.

1. Augment the prediction model with constant disturbance states

$$\begin{aligned} x_{k+1} &= Ax_k + B_d d_k + Bu_k, \\ d_{k+1} &= d_k, \\ y_k &= Cx + k + C_d d_k, \end{aligned} \tag{1.20}$$

   where $d_k \in \mathbb{R}^{n_y}$, i.e., the number of disturbances equals the number of measured outputs.

2. Choose B_d and C_d such that the augmented system (1.20) is observable. A necessary and sufficient condition for observability is

$$\mathtt{rank} \begin{bmatrix} A - \mathbf{I}^{n_x} & B_d \\ C & C_d \end{bmatrix} = n_x + n_y. \tag{1.21}$$

3. Ensure that the pair (A, B) is controllable and

$$\mathtt{rank} \begin{bmatrix} A - \mathbf{I}^{n_x} & B \\ C & 0 \end{bmatrix} = n_x + n_y. \tag{1.22}$$

   This requires that the number of manipulated variables is at least equal to the number of controlled outputs, which is a rather intuitive condition for being able to achieve perfect tracking of all output references.

4. Design an asymptotically stable linear observer

$$\begin{aligned} \hat{x}_{k+1} &= A\hat{x}_k + B_d\hat{d}_k + Bu(k) + L_x\left(C\hat{x}_k + C_d\hat{d}_k - y_k^p\right) \\ \hat{d}_{k+1} &= \hat{d}_k + L_d\left(C\hat{x}_k + C_d\hat{d}_k - y_k^p\right). \end{aligned} \tag{1.23}$$

In most practical cases, assuming that the overall system remains observable, one choses $B_d = 0$ and $C_d = I$, so that d_k asymptotically compensates for mismatches of the DC gain of the model and the real (possibly nonlinear) DC gain of the system. This is the default choice for example in the Model Predictive Control Toolbox for MATLAB [93].

A Model Predictive Controller built with an augmented prediction model as in (1.20), paired with a state observer as in (1.23), guarantees zero-offset tracking in the steady-state, that is $y_k \to r_k$ as $k \to \infty$. An intuitive argument for this is the following: in steady-state, the observer (1.23) makes the predicted output equal to the measured output, the MPC controller makes the predicted output equal to the set-point, and as a result the measured output coincides with the set-point.

Velocity Forms The *velocity form* is a reformulation for linear systems that, adopted in the prediction model used by MPC, leads to offset-free tracking in the presence of constant disturbances. The key principle behind velocity forms is to move to an enlarged state composed by the *increments* of the original system states, plus the output errors.

Velocity forms were initially applied to LQ regulators [94], and then extended to Model Predictive Control frameworks [95]. In [96], an offset-free MPC formulation based on velocity form is proposed, along with stability and feasibility results even for unfeasible desired references. An extension that includes the case

of non-constant but bounded disturbances, relying on a tube-based approach, is presented in [97].

A velocity formulation for a linear system in the form

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + \omega_k, \\ y_k &= Cx_k, \end{aligned} \tag{1.24}$$

where the number of outputs n_y equals the number of inputs n_u and the disturbance $\omega \in \mathcal{W}$ is bounded, can be computed by means of the following procedure.

1. Ensure that the pair (A, B) is reachable and

$$\mathtt{rank} \begin{bmatrix} \mathbf{I}^{n_x} - A & -B \\ C & 0 \end{bmatrix} = n_x + n_y. \tag{1.25}$$

2. Define the extended system state vector as

$$\xi_k \triangleq \begin{bmatrix} \Delta x_k \\ \varepsilon_k \end{bmatrix} = \begin{bmatrix} x_k - x_{k-1} \\ y_k - r_k \end{bmatrix}, \tag{1.26}$$

for which the system dynamics becomes

$$\xi_{k+1} = \begin{bmatrix} A & 0 \\ CA & \mathbf{I}^{n_u} \end{bmatrix} \xi_k + \begin{bmatrix} B & CB \end{bmatrix} \Delta u_k + \begin{bmatrix} \mathbf{I}^{n_x} \\ C \end{bmatrix} \Delta \omega_k. \tag{1.27}$$

3. Map the constraints on the original states and inputs vectors x, u into constraints on the enlarged state ξ.

One then derives a nominal model by neglecting the disturbance term in the system equation (1.27). Using the resulting nominal system as the prediction model in the MPC formulation, paired with new the state constraints defined as in Step 3, grants offset-free reference tracking.

Regarding mapping the original constraints into the new model, a detailed (and nontrivial) procedure can be found in [97, Sec. II.B]. Here, we propose a new simpler approach to enforce input and output constraints. We extend further the model in (1.27) by adding u_{k-1} (for input constraints) and x_k (for output constraints) as extra states in the overall model, and u_k, $y_k = Cx_k$ as constrained outputs. Hence, the overall model becomes

$$\begin{aligned} \xi_{k+1} &= \begin{bmatrix} A & 0 & 0 & 0 \\ CA & \mathbf{I}^{n_u} & 0 & 0 \\ 0 & 0 & \mathbf{I}^{n_u} & 0 \\ \mathbf{I}^{n_x} & 0 & 0 & \mathbf{I}^{n_x} \end{bmatrix} \xi_k + \begin{bmatrix} B \\ CB \\ \mathbf{I}^{n_u} \\ 0 \end{bmatrix} \Delta u_k + \begin{bmatrix} \mathbf{I}^{n_x} \\ C \\ 0 \\ 0 \end{bmatrix} \Delta \omega_k \\ z_k &= \begin{bmatrix} 0 & 0 & \mathbf{I}^{n_x} & 0 \\ 0 & 0 & 0 & C \end{bmatrix} \xi_k \end{aligned} \tag{1.28}$$

where

$$\xi_k \triangleq \begin{bmatrix} \Delta x_k \\ \varepsilon_k \\ u_{k-1} \\ x_k \end{bmatrix}, \quad z_k \triangleq \begin{bmatrix} u_k \\ y_k \end{bmatrix}.$$

Reference Governor The main idea of the so-called "reference governors" is to manipulate the reference signal before feeding it to

the controller (usually an existing linear controller), so to achieve desired properties such as constraint satisfaction [98–100].

Along the idea of manipulating the reference signal, we propose here a reference governor method that achieves offset-free tracking by adding the integration of the tracking errors on the reference signals. This means that the actual reference vector $r(t)$ fed to the controller is different from the desired reference vector $\bar{r}(t)$, and $r(t)$ is computed as follows:

$$r(t) = \bar{r}(t) - \int_0^t \left(y(\tau) - \bar{r}(\tau)\right) \, \mathrm{d}\tau, \tag{1.29}$$

where $y(t)$ are the measurements of the system outputs to be tracked. The rationale is rather intuitive: if the output $y(t)$ is smaller than the desired $\bar{r}(t)$, the reference signal given to the MPC controller is increased, or vice versa decreased, until a steady-state value $r(t)$ is achieved (possibly $r(t) \neq \bar{r}(t)$) that makes $y(t) = \bar{r}(t)$.

A possible extension is to limit the rate of change of the tracking error signal fed to the integrator. This prevents an undesired charge of the integral state in the transient after a large reference change, that could otherwise lead to large overshoots. The overall control loop is depicted in Figure 1.6.

1.5.2 Simulation

In this section we detail a closed-loop simulation based on a real-world control scenario where integral action is needed in order to achieve offset-free control. The simulation is repeated for each of

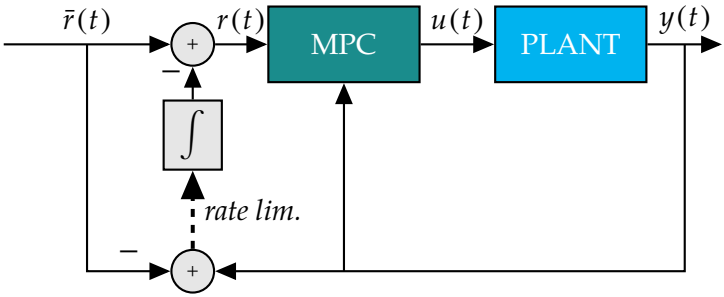

Figure 1.6: Scheme for integral action on the reference signal. $\bar{r}(t)$ is the desired reference vector; $r(t)$ is the actual reference vector fed to the controller; $u(t)$ is the control action; $y(t)$ is the vector of measured system outputs.

the integral action techniques introduced in this section in order to assess their performance.

Gasoline engines for vehicles are required to satisfy CO_2 emission regulations, which are becoming stricter and stricter. At the same time, consumers demand high performance and low fuel consumption. This challenge paves the way for advanced control techniques, with Model Predictive Control being a suitable candidate due to the support for multiple-input, multiple-output systems and the optimal constraint handling.

The control objective is to regulate the engine airflow and the boost pressure generated by the compressor to maintain an optimal stoichiometric fuel-to-air ratio and therefore minimizing the emissions and fuel consumption, while guaranteeing the required torque output. This is achieved by manipulating the throttle placed

at the engine intake manifold inlet, and the wastegate valve which deviates a fraction of the exhaust gases back into the turbine.

For a comprehensive formulation of the engine model that includes the nonlinear equations, the reader is referred to [101].

The simulation is performed at a sampling time $T_s = 20\,ms$. The system state is estimated from the measured variables using a Kalman filter.

The throttle control input is constrained within the $[0,\ 100]$ range with a maximum rate of 200 per second, while the wastegate input is constrained in the $[0,\ 0.4]$ interval with a maximum rate of 2.5 per second. An additional constraint is imposed on the wastegate state, which need to be in the range $[0,\ 0.5]$.

In order to properly test the techniques for offset-free control, a mismatch between the prediction model and the plant model generating the output data is introduced. This is obtained by varying the parameter denoting the sensitivity of the engine flow with respect to the intake pressure.

The test scenario simulates a tip-in maneuver, with a sudden increase in the torque request coming from the driver. The step is sufficiently large to cause actuator saturation and therefore require a proper anti-windup formulation when adopting the integral states method. Simulation results are shown in Figure 1.7, where the engine flow closed-loop trajectories are compared for a baseline controller without integral action, and controllers modified according to the techniques introduced in this section. With the baseline controller a

steady-state offset is present; on the other hand, all the modified controllers are able to achieve offset-free tracking, although with varying closed-loop performance.

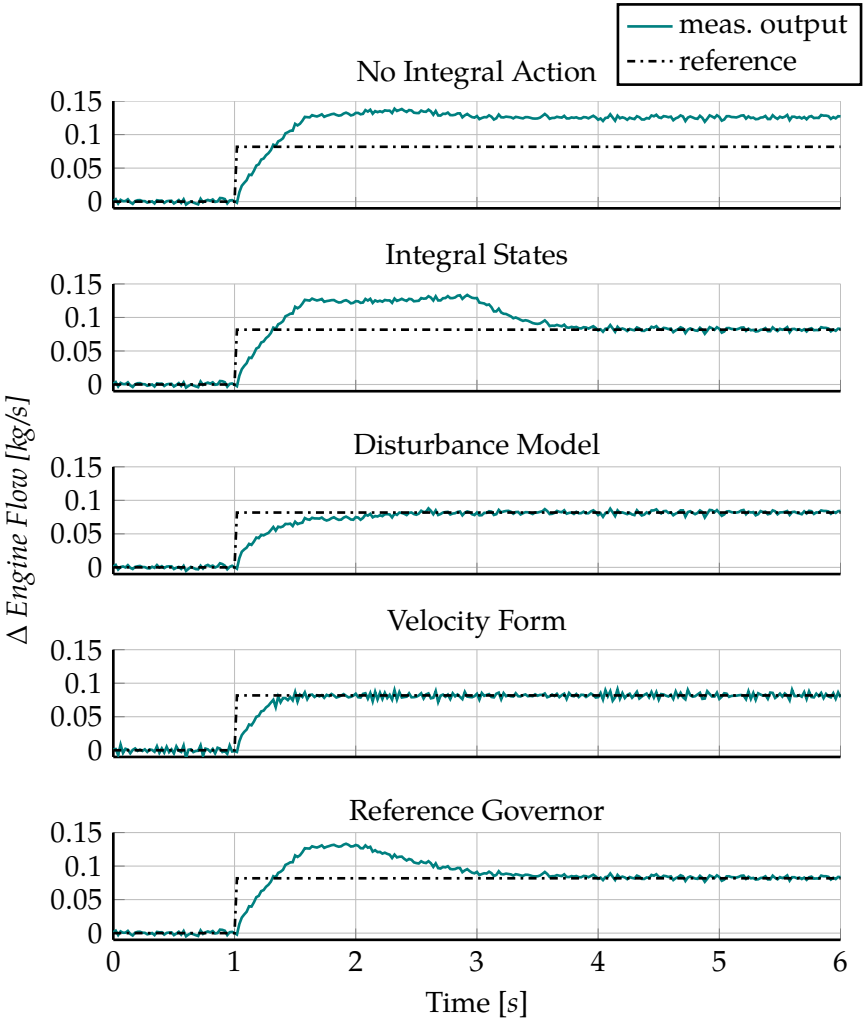

Figure 1.7: Closed-loop simulation of integral action techniques for tracked variable engine flow during a tip-in maneuver. Baseline controller performance without integral action (*top*) is compared with modified controllers according to the offset-free techniques introduced in Section 1.5.

1.6 Model Predictive Control for Aerospace Applications

In aerospace applications, Model Predictive Control has been shown to be an effective approach for problems of rendezvous and proximity maneuvers [102–105]. MPC was a natural choice because of its systematic handling of constraints such as maintaining a spacecraft position within line of sight of the docking port, or terminal velocity constraints such that spacecraft and port velocities match when docking.

Another relevant aerospace application for MPC is given by the problem of attitude control. Recent developments include [106] where an explicit solution is derived from a linearized spacecraft model, [107] where MPC is applied to spacecraft attitude control using magnetic actuators, [108] which demonstrates a global in orientation attitude stabilization using a Lie group variational integrator-based model, and [109] where a fast real time MPC implementation for low level Guidance/Navigation/Control functions on a low power embedded computing platform is presented.

In Chapter 5 we investigate the problem of attitude control for spacecraft equipped with reaction wheels actuators. Reaction wheels, a type of momentum-exchange devices, are a common way to control the attitude in a spacecraft, requiring only electrical power to operate [110, 111]. However, the presence of external disturbances can lead to a constant increase of the wheels rotational speeds, and ultimately to a saturation. Typically, this problem is

solved by periodically activating mass-expulsion devices such as thrusters [112, 113], or by using magnetic coils [114, 115].

1.7 Motivation and Contribution

Technological advancements in recent years brought us enhanced computing capabilities on processing units for embedded systems, together with cheaper per-unit cost, lower power consumption, and larger memory sizes. This paved the way for advanced control techniques in a broader range of domains, including applications with fast dynamics and tight real-time constraints such as automotive and aerospace.

Amongst such advanced techniques stand out Model Predictive Control; interest in embedded MPC implementations is growing both in academic communities and industry practitioners. This is justified by the flexibility of control solutions based on MPC, that allow for optimization of arbitrary performance indexes, while inherently accounting for input and output constraints.

However, embedding Model Predictive Control on hardware platforms poses quite a few challenges, both from the theoretical point of view, with the need for proper optimization techniques, and from the implementation aspects. Those challenges are the motivation behind the work presented in this thesis.

The main contributions of this thesis are the analysis of quadratic programming methods specifically tailored for fixed-point arithmetic, targeting embedded MPC applications. For these methods, we analyze round-off error propagation, and give specific guidelines to avoid the occurrence of overflow errors. Moreover, we conduct experimental tests to assess algorithms performance when

implemented on hardware. Finally, we investigate the problem of spacecraft attitude control with reaction wheels actuator, and demonstrate the viability and good performance of an embedded MPC controller supported by one of the fixed-point QP solvers introduced.

Here follows a detailed description of the contributions given by each chapter of the thesis.

Chapter 2: Gradient Projection Methods in Finite Precision Arithmetic

When running Quadratic Programming solvers on embedded devices, switching to fixed-point arithmetic may lead to significantly faster computation, smaller chip size, and lower power consumption, as shown in Section 1.3.

Nevertheless, the improvement in performance comes at the price of reduced range in which numbers can be represented, and round-off errors due to finite-precision computations. Because of this, quadratic programming algorithms that perform well in floating-point may perform much worse (even completely wrongly) in fixed-point. Therefore, additional challenges arise when dealing with fixed-point arithmetic, mainly studying round-off error accumulation during algorithm iterations, and establishing bounds on the magnitude of the computed variables to avoid overflows.

In Chapter 2, we propose a Dual Gradient Projection method, which can be seen as a simplified (non-accelerated) version of GPAD algorithm (cf. [76]), specifically tailored for fixed-point implementation.

The main contribution in this chapter is a detailed convergence rate and asymptotic error analysis in terms of primal cost and primal feasibility in the presence of round-off errors due to fixed point arithmetic. In addition to that, specific guidelines are given on the number of decimal bits that certify the convergence to a target suboptimal solution, as well as on the number of integer bits to

avoid overflow errors. The machinery used to perform the analysis is based on the notion of the inexact oracle proposed in [116].

The work of Chapter 2 has been conducted in cooperation with P. Patrinos (*IMT Lucca*) and A. Bemporad (*IMT Lucca*), and is based on publications [117, 118]:

- P. Patrinos, A. Guiggiani, and A. Bemporad, "Fixed-Point Dual Gradient Projection for Embedded Model Predictive Control", in *Proc. European Control Conference*, 2013, pp. 3602-3607.
- P. Patrinos, A. Guiggiani, and A. Bemporad, "A Dual Gradient-Projection Algorithm for Model Predictive Control in Fixed-Point Arithmetic", *Automatica*, vol. 55, pp. 226-235, 2015.

Chapter 3: Proximal Newton Methods in Finite Precision Arithmetic

In Chapter 3 is presented an implementation of a Quadratic Programming solver based on the Proximal Newton method of [119]. With motivations similar to the ones behind the work in Chapter 2, emphasis is put on issues arising from fixed-point arithmetics. An analysis of the accumulation of round-off errors is performed, and guidelines for the number of integer bits are provided such that occurrence of overflow errors is avoided.

Algorithm optimizations to enhance efficiency and robustness are also proposed, with focus on the problems of preconditioning and division-free computations. Moreover, an in-depth comparison against gradient-based approaches, both in terms of computational complexity and solution accuracy, is presented. Finally, algorithm performance when employed as a solver in an MPC controller is evaluated.

The work of Chapter 3 has been conducted in cooperation with P. Patrinos (*IMT Lucca*) and A. Bemporad (*IMT Lucca*), and is based on publication [120]:

- A. Guiggiani, P. Patrinos, and A. Bemporad, "Fixed-Point Implementation of a Proximal Newton Method for Embedded Model Predictive Control", in *Proc. IFAC World Congress*, 2014, pp. 2921-2926.

Chapter 4: Experimental Tests

The goal of Chapter 4 is to evaluate performance of the algorithms introduced in Chapter 2 and Chapter 3, when implemented on hardware platforms with similar computing capabilities and memory amounts to the ones generally employed in actual embedded control systems.

The main contribution of the chapter comes from the implementation of the fixed-point Dual Gradient Projection method on a reconfigurable hardware platform based on FPGA. Thanks to the efficiency of these devices, computation times for the QP solution are shown to lie in the sub-millisecond range. Other contributions in the chapter are experimental tests for both the Gradient Projection method and the Proximal Newton method for implementation on low-power, general purpose embedded devices based on ARM Cortex processing units, with emphasis on the performance gains obtained by switching from floating-point to fixed-point arithmetic.

The work of Chapter 4 has been conducted in cooperation with P. Patrinos (*IMT Lucca*), M. Rubagotti (*Nazarbayev University*) and A. Bemporad (*IMT Lucca*), and is based on publication [121]:

- M. Rubagotti, P. Patrinos, A. Guiggiani, and A. Bemporad, "Real-Time Model Predictive Control Based on Dual Gradient Projection: Theory and Fixed-Point FPGA Implementation", *International Journal of Robust and Nonlinear Control*,

along with publications [117, 118, 120].

Chapter 5: Aerospace Applications

The work detailed in Chapter 5 introduces the application of spacecraft attitude control with reaction wheels actuators. The control setup is characterized by multiple actuators and multiple tracked variables, with input and (possibly) state constraints; it is therefore very well suited for approaches based on embedded MPC.

The contribution in Chapter 5 is given by an embedded MPC formulation for spacecraft attitude control. The controller relies on the fixed-point Dual Gradient Projection algorithm introduced in Chapter 2 for the solution of the QP problem, equipped with the integral action technique based on reference governor detailed in Section 1.5. A special control model formulation allows for extended domain of attraction while maintaining the QP problem small. The controller complexity is evaluated, and its performance tested in closed-loop simulations.

Moreover, the problem of reaction wheel desaturation is addressed. Two special controller formulations, that exploit gravity gradient effects or the Earth magnetic field to achieve desaturation without the need of fuel-consuming devices, are presented. Emphasis is posed on the controller performance gains derived from MPC approaches, mainly due to the constraints handling.

The work of Chapter 5 has been conducted in cooperation with I. Kolmanovsky (*University of Michigan*), P. Patrinos (*IMT Lucca*), and A. Bemporad (*IMT Lucca*), and is based on publications [122, 123]:

- A. Guiggiani, I. Kolmanovsky, P. Patrinos, and A. Bemporad, "Fixed-Point Constrained Model Predictive Control of Spacecraft Attitude", in *Proc. American Control Conference*, 2015.
- A. Guiggiani, I. Kolmanovsky, P. Patrinos, and A. Bemporad, "Constrained Model Predictive Control of Spacecraft Attitude with Reaction Wheels Desaturation", in *Proc. European Control Conference*, 2015.

Chapter 2

Gradient Projection Methods in Finite Precision Arithmetic

A Dual Gradient Projection method is proposed in this chapter. Section 2.1 gives general theoretical results when the algorithm runs with an inexact oracle. In Section 2.2 an inexact DGP method is applied to a modified version of the dual problem and its convergence rate with respect to primal suboptimality and infeasibility is analyzed. In Section 2.3, the general results of the proposed inexact DGP method are applied to the case of QP based on a fixed-point implementation. Finally, simulation results are presented in Section 2.4.

2.1 Inexact Gradient Projection

Consider a Quadratic Programming problem in the form

$$\begin{aligned} &\min_{y} \quad \Phi(y) \\ &\text{subject to} \quad y \in \mathcal{Y}, \end{aligned} \tag{2.1}$$

where $\mathcal{Y}$ is a nonempty closed convex subset of $\mathbb{R}^m$, and $\Phi : \mathbb{R}^m \to \mathbb{R}$ is convex, L_Φ-smooth, i.e., there exists a $L_\Phi > 0$ such that

$$\|\nabla\Phi(y) - \nabla\Phi(w)\| \leq L_\Phi \|y - w\|, \qquad y, w \in \mathbb{R}^m.$$

Assumption 2.1.1. *Problem* (2.1) *admits solution, i.e.,*

$$\Phi^\star \triangleq \inf_{y \in \mathcal{Y}} \Phi(y)$$

is finite and

$$\mathcal{Y}^\star \triangleq \operatorname*{argmin}_{y \in \mathcal{Y}} \Phi(y)$$

is non-empty.

The goal is to find an approximate solution of (2.1) by applying the Gradient Projection method that, at each iteration ν, updates the current solution estimate as follows

$$y_{(\nu+1)} = \left[y_{(\nu)} - \tfrac{1}{L_\Phi} \nabla\Phi(y_{(\nu)}) \right]_{\mathcal{Y}}, \tag{2.2}$$

where the operator $[\cdot]_{\mathcal{Y}}$ denotes a projection on the set $\mathcal{Y}$.

However, it is assumed that the gradient of Φ cannot be computed exactly. At every iterate, instead of the gradient $\nabla\Phi(y_{(\nu)})$, is

only available an *inexact oracle* providing an *approximate* first-order information of the cost function. The notion of inexact oracle was introduced in [116] and is defined as follows.

Definition 2.1.1 (Inexact Oracle)**.** $\Phi : \mathbb{R}^m \to \mathbb{R}$ is equipped with a *first-order* (δ, L)*-oracle* if for any $w \in \mathbb{R}^m$ one can compute a pair $(\Phi_{\delta,L}(w), s_{\delta,L}(w)) \in \mathbb{R} \times \mathbb{R}^m$ such that

$$0 \leq \Delta_{\delta,L}(y; w) \leq \frac{L}{2}\|y - w\|^2 + \delta, \quad \forall y \in \mathbb{R}^m, \tag{2.3}$$

where

$$\begin{aligned} \Delta_{\delta,L}(y; w) &\triangleq \Phi(y) - \ell_{\delta,L}(y; w), \\ \ell_{\delta,L}(y; w) &\triangleq \Phi_{\delta,L}(w) + s_{\delta,L}(w)'(y - w). \end{aligned}$$

We call the first-order $(0, L_\Phi)$-oracle, $(\Phi(y), \nabla\Phi(y))$, the *exact oracle*.

The implementation of a Gradient Projection method with inexact oracle becomes

$$y_{(\nu+1)} = \left[y_{(\nu)} - \tfrac{1}{L} s_{\delta,L}(y_{(\nu)}) \right]_{\mathcal{Y}}. \tag{2.4}$$

Comparing the exact Gradient Projection method (2.2) with the inexact counterpart (2.4), two key differences need to be noted:

1. in the inexact method, $s_{\delta,L}(y_{(\nu)})$ is used instead of the exact gradient $\nabla\Phi(y_{(\nu)})$;

2. in the inexact method, the step size is determined by the new constant L which is different from the Lipschitz constant L_Φ.

We will now introduce two lemmas, essential in proving convergence rates for both primal and dual versions of the inexact Gradient Projection algorithm; the first is a direct extension of [124, Lemma 3.2] in the inexact case[1].

Lemma 2.1.1. *Let* $\{y_{(\nu)}\}$ *be generated by iterating* (2.4) *from any* $y_{(0)} \in \mathcal{Y}$.

Then, for any $y \in \mathcal{Y}$ *and* $\nu \in \mathbb{N}$ *the following inequality holds*

$$\ell_{\delta,L}(y_{(\nu+1)}; y_{(\nu)}) + \tfrac{L}{2}\|y_{(\nu+1)} - y_{(\nu)}\|^2 \leq \ell_{\delta,L}(y; y_{(\nu)}) + \tfrac{L}{2}\|y_{(\nu)} - y\|^2 - \tfrac{L}{2}\|y_{(\nu+1)} - y\|^2. \tag{2.5}$$

Proof. The single algorithm iteration

$$y_{(\nu+1)} = \left[y_{(\nu)} - \tfrac{1}{L} s_{\delta,L}(y_{(\nu)})\right]_{\mathcal{Y}}$$

can be rewritten as

$$y_{(\nu+1)} = \operatorname*{argmin}_{w \in \mathcal{Y}} \{\ell_{\delta,L}(w; y_{(\nu)}) + \tfrac{L}{2}\|w - y_{(\nu)}\|^2\}. \tag{2.6}$$

Writing down the optimality conditions for (2.6) (see e.g., [126, Cor. 26.3], or [32, Prop. 5.4.7]), we obtain

$$\tfrac{1}{L}\ell_{\delta,L}(y_{(\nu+1)}; y_{(\nu)}) \leq \tfrac{1}{L}\ell_{\delta,L}(y; y_{(\nu)}) + (y_{(\nu+1)} - y_{(\nu)})'(y - y_{(\nu+1)}), \tag{2.7}$$

[1] In [124, Lemma 3.2] the property is proved for general Bregman distances, see also [32, 125].

for all $y \in \mathcal{Y}$. Now, by rearranging terms,

$$\begin{aligned}(2.8)\quad &\tfrac{1}{L}\ell_{\delta,L}(y_{(\nu+1)}; y_{(\nu)}) - y'_{(\nu)}(y_{(\nu+1)} - y_{(\nu)}) + \\ &- y'_{(\nu+1)}(y - y_{(\nu+1)}) \le \tfrac{1}{L}\ell_{\delta,L}(y; y_{(\nu)}) - y'_{(\nu)}(y - y_{(\nu)})\end{aligned}$$

After some simple algebraic manipulations, and adding $\frac{1}{2}\|y\|^2 - \frac{1}{2}\|y_{(\nu)}\|^2$, we get (2.5). □

Lemma 2.1.2. *Let* $\{y_{(\nu)}\}$ *be generated by iterating* (2.4) *from any* $y_{(0)} \in \mathcal{Y}$.

Then, for any $y \in Y$ *and* $\nu \in \mathbb{N}$ *the following inequality holds*

$$\begin{aligned}(2.9)\quad &\sum_{i=0}^{\nu}(\Phi(y_{(i+1)}) - \Phi(y)) + \sum_{i=0}^{\nu}\Delta_{\delta,L}(y; y_{(i)}) + \\ &+ \tfrac{L}{2}\|y - y_{(\nu+1)}\|^2 \le \tfrac{L}{2}\|y - y_{(0)}\|^2 + (\nu+1)\delta.\end{aligned}$$

Proof. By the second part of (2.3) and Lemma 2.1.1

$$\begin{aligned}(2.10)\quad \Phi(y_{(\nu+1)}) &\le \ell_{\delta,L}(y_{(\nu+1)}; y_{(\nu)}) + \tfrac{L}{2}\|y_{(\nu+1)} - y_{(\nu)}\|^2 + \delta \\ &\le \ell_{\delta,L}(y; y_{(\nu)}) + \tfrac{L}{2}\|y - y_{(\nu)}\|^2 - \tfrac{L}{2}\|y - y_{(\nu+1)}\|^2 + \delta,\end{aligned}$$

or

$$\begin{aligned}(2.11)\quad &\Phi(y_{(\nu+1)}) - \Phi(y) + \Delta_{\delta,L}(y; y_{(\nu)}) + \tfrac{L}{2}\|y - y_{(\nu+1)}\|^2 \le \\ &\tfrac{L}{2}\|y - y_{(\nu)}\|^2 + \delta.\end{aligned}$$

Combining (2.11) with

$$\Delta(y; y_{(\nu)}) - \tfrac{L_\Psi}{2}\|y - y_{(\nu)}\|^2 \le \Delta_{\delta,L}(y; y_{(\nu)})$$

we arrive at

$$\Phi(y_{(\nu+1)}) - \Phi(y) + \Delta(y; y_{(\nu)}) + \\ + \tfrac{L}{2}\|y - y_{(\nu+1)}\|^2 \leq \tfrac{L+L_\Phi}{2}\|y - y_{(\nu)}\|^2 + \delta \tag{2.12}$$

Summing over $0, \ldots, \nu$ we prove (2.9). □

Remark 2.1.1. Lemma 2.1.2 is the main difference of the analysis proposed in this section compared to that of [116]. It provides the inequality (2.9) that plays a key role in deriving convergence rate estimates not only for the primal version of the inexact Gradient Projection (as done in [116]) but also for its dual counterpart. This will ultimately allow to deduce convergence rate estimates for primal feasibility and optimality in fixed-point implementations of the Dual Gradient Projection algorithm for Model Predictive Control problems.

The next theorem provides convergence rate estimates for the inexact primal Gradient Projection scheme defined in (2.4).

Theorem 2.1.3 (Convergence of Inexact Gradient Projection Method)**.** *Let* $\{y_{(\nu)}\}_{\nu\in\mathbb{N}}$ *be generated by iterating* (2.4) *from any* $y_{(0)}\in\mathcal{Y}$ *and let*

$$\bar{y}_{(\nu+1)} \triangleq \tfrac{1}{\nu+1}\sum_{i=0}^{\nu} y_{(i+1)}.$$

Then, the convergence of the cost function Φ *to the optimal value* $\Phi^\star$ *is bounded as follows*

$$\Phi(\bar{y}_{(\nu+1)}) - \Phi^\star \leq \tfrac{L}{2(\nu+1)}\|y^\star - y_{(\nu)}\|^2 + \delta. \tag{2.13}$$

Proof. Putting $y = y^\star$ in (2.9), dropping the terms $\sum_{i=0}^{\nu} \Delta_{\delta,L}(y^\star; y_{(i)})$ and $\frac{L}{2}\|y^\star - y_{(\nu+1)}\|^2$ since they are nonnegative, and dividing by $(\nu+1)$, we arrive at

$$\frac{1}{(\nu+1)} \sum_{i=0}^{\nu} (\Phi(y_{(i+1)}) - \Phi^\star) \leq \frac{L}{2(\nu+1)} \|y^\star - y_{(0)}\|^2 + \delta. \tag{2.14}$$

Since Φ is convex, the following holds

$$\Phi(\bar{y}_{(\nu+1)}) \leq \frac{1}{(\nu+1)} \sum_{i=0}^{\nu} \Phi(y_{(i+1)}). \tag{2.15}$$

Finally, by substituting (2.15) in (2.14) we prove (2.13). □

2.2 Inexact Dual Gradient Projection

Consider the problem

$$\begin{aligned} \mathbb{P}: \quad & \min_{z} \quad V(z) \\ & \text{subject to} \quad g(z) \leq 0 \end{aligned} \tag{2.16}$$

We call (2.16) the *primal problem* (recall the notions on Lagrangian Duality in Section 1.2.1). We assume that (2.16) is be feasible.

$V : \mathbb{R}^n \to \mathbb{R}$ is a differentiable, strongly convex function with convexity parameter κ_V, i.e.,

$$(\nabla V(z_1) - \nabla V(z_2))'(z_1 - z_2) \geq \kappa_V \|z_1 - z_2\|^2$$

for all $z_1, z_2 \in \mathbb{R}^n$, and $g(z) = Gz - b$, $b \in \mathbb{R}^m$. The unique solution of (2.16) is denoted by $z^\star$.

The ultimate goal is to compute an $(\varepsilon_V, \varepsilon_g)$-optimal solution for (2.16), defined as follows.

Definition 2.2.1. [$(\varepsilon_V, \varepsilon_g)$-optimal solution] Consider two nonnegative constants ε_V, ε_g. Vector z is an $(\varepsilon_V, \varepsilon_g)$-optimal solution for (2.16) if

$$V(z) - V^\star \leq \varepsilon_V \tag{2.17a}$$

$$\|[g(z)]_+\|_\infty \leq \varepsilon_g, \tag{2.17b}$$

where (2.17a) is a bound on the solution suboptimality, i.e. the discrepancy between the optimal and the achieved cost function values, and (2.17b) is a bound on the solution infeasibility, i.e. the maximal constraint violation.

Next, consider the *Lagrangian function* of problem (2.16)

$$\mathcal{L}(z, y) = V(z) + y'g(z).$$

The (negative of the) *dual problem* of (2.16) can be expressed as (2.1), with the convex function $\Phi : \mathbb{R}^m \to \mathbb{R}$ given by

$$\Phi(y) = -\inf_{z \in \mathbb{R}^n} \mathcal{L}(z, y) \tag{2.18}$$

and $\mathcal{Y} = \mathbb{R}^m_+$, i.e., the nonnegative orthant.

Assume that there is no duality gap, i.e., $V^\star = -\Phi^\star$. This assumption is fulfilled if, for example, Problem (2.16) is a convex quadratic program that is feasible, or the Slater condition holds [30–32].

Since V is strongly convex,

$$z_y^\star = \operatorname*{argmin}_{z \in \mathbb{R}^n} \mathcal{L}(z, y)$$

is unique for any $y \geq 0$, and Φ is L_Φ-smooth with $L_\Phi = \|G\|^2/\kappa_V$ [57].

The gradient of Φ is given by

$$\nabla\Phi(y) = -g(z_y^\star). \tag{2.19}$$

Furthermore, we can obtain the unique optimal solution of (2.16) from any dual optimal solution $y^\star \in \mathcal{Y}^\star$ via

$$z^\star = \operatorname*{argmin}_{z \in \mathbb{R}^n} \mathcal{L}(z, y^\star). \tag{2.20}$$

The gradient projection algorithm applied to the dual problem (2.18) becomes (for a given $y_{(0)}$)

$$z_{(\nu)} = \operatorname{argmin}_{z \in \mathbb{R}^n} \mathcal{L}(z, y_{(\nu)}) \tag{2.21a}$$

$$y_{(\nu+1)} = \left[y_\nu + \tfrac{1}{L_\Phi} g(z_{(\nu)}) \right]_+ \tag{2.21b}$$

where $[\cdot]_+$ denotes the projection on the nonnegative orthant.

Next, assume that for every $y \in \mathcal{Y}$, instead of the exact gradient $\nabla\Phi(y) = -g(z_y^\star)$, one can only calculate an approximate gradient

$$\tilde{\nabla}\Phi(y) = -g(z_y) + \xi, \tag{2.22}$$

where

$$\begin{aligned} \|z_y - z_y^\star\| &\leq \epsilon_z, \\ \|\xi\| &\leq \epsilon_\xi, \end{aligned} \tag{2.23}$$

for given positive constants ϵ_z, ϵ_ξ.

2.2.1 Modified Primal-Dual Pair

The goal is to construct a first-oder inexact oracle, introduced in Definition 2.1.1, for Φ (cf. (2.18)) with $s_{\delta,L}(y) = \tilde{\nabla}\Phi(y)$.

Convergence rate results for Gradient Projection methods in the presence of an additive disturbance ξ require the constraint set $\mathcal{Y}$ of the dual problem (2.1) to be bounded [127, 128]. For this reason, the dual problem (2.1) will be modified in order to have a bounded constraint set.

Let $d \in \mathbb{R}^m$ be such that its i-th element satisfies

$$d_i \geq \max\{y_i^\star, 1\} \tag{2.24}$$

for some $y^\star \in Y^\star$, and

$$\mathcal{Y}_\alpha \triangleq \{y \in \mathbb{R}^m \mid 0 \leq y \leq \alpha d\}, \quad \alpha \geq 1. \tag{2.25}$$

For a discussion about the choice of the parameter α we refer the reader to Section 2.2.4.

Furthermore, let

$$\begin{aligned} D &\triangleq \|d\|, \\ D_\alpha &\triangleq \max_{y_1, y_2 \in \mathcal{Y}_\alpha} \|y_1 - y_2\| = \alpha D, \end{aligned}$$

the diameter of $\mathcal{Y}_\alpha$. Notice that $\mathcal{Y}_\alpha \subseteq \mathcal{Y}$ and $\mathcal{Y}_\alpha \cap \mathcal{Y}^\star \neq \emptyset$.

Next, consider the following modified dual problem

$$\begin{aligned} \min \quad & \Phi(y) \\ \text{subject to} \quad & y \in \mathcal{Y}_\alpha, \end{aligned} \tag{2.26}$$

where the non-negativity constraint has been replaced by the constraint set $\mathcal{Y}_\alpha$. Obviously it holds that

$$\mathcal{Y}_\alpha^\star \triangleq \operatorname*{argmin}_{y \in \mathcal{Y}_\alpha} \Phi(y) \subseteq \mathcal{Y}^\star,$$

therefore any optimal solution of the modified dual problem (2.26) is also a solution of the original dual problem. Hence, one can compute an optimal solution for the primal problem (2.26) and recover the optimal solution for (2.16) via (2.20).

Remark 2.2.1. In principle, determining a vector d such that (2.24) holds requires one to know bounds on the elements of a dual optimal solution. If (2.16) is a parametric QP, as in embedded linear MPC, then tight uniform bounds (valid for every admissible parameter vector) can be computed using techniques described in [76]. In fact, one has to compute such bounds anyway, since they are imperative for determining the worst-case number of iterations, and consequently the worst-case running time of the algorithm, a central concern in embedded optimization applications (see, e.g., [71, 129, 130]).

2.2.2 Inexact Oracle

We are now ready to derive an inexact oracle for Φ on $\mathcal{Y}_\alpha$ under assumptions (2.22), (2.23). The next proof follows the lines in [116].

Proposition 2.2.1. *Consider* Φ *given by* (2.18). *The pair*

$$\Phi_{\delta,L}(y) = -\mathcal{L}(z_y, y) - \alpha D\epsilon_\xi, \tag{2.27a}$$

$$s_{\delta,L}(y) = \tilde{\nabla}\Phi(y) = -g(z_y) + \xi \tag{2.27b}$$

furnishes a (δ_α, L)*-oracle for* Φ *on* Y_α, *where*

$$\delta_\alpha \triangleq L_V \epsilon_z^2 + 2\alpha D\epsilon_\xi,$$
$$L \triangleq \tfrac{2}{\kappa_V}\|G\|^2.$$

Proof. Since $\mathcal{L}(\cdot, y)$ is L_V-smooth and $z_y^\star$ is its unconstrained minimum, we have that

$$\mathcal{L}(z_y, y) - \mathcal{L}(z_y^\star, y) \leq \tfrac{L_V}{2}\|z_y - z_y^\star\|^2.$$

Therefore, $\|z_y - z^\star\| \leq \epsilon_z$ implies

$$\mathcal{L}(z_y, y) - \mathcal{L}(z_y^\star, y) \leq \tfrac{L_V}{2}\epsilon_z^2. \tag{2.28}$$

In [116, Sec. 3.2] it is shown that if for every $y \in \mathcal{Y}_\alpha$ one is able to compute a z_y such that (2.28) is satisfied, then $(-\mathcal{L}(z_y, y), -g(z_y))$ is a $(L_V\epsilon_z^2, L)$-oracle for Φ.

Next, consider any $w, y \in \mathcal{Y}_\alpha$ and ξ such that $\|\xi\| \leq \epsilon_\xi$. We have

$$\begin{aligned}
\Phi(w) &= -\mathcal{L}(z_w^\star, w) \\
&\geq -\mathcal{L}(z_y, w) \\
&\geq -\mathcal{L}(z_y, y) - g(z_y)'(w - y) \\
&= -\mathcal{L}(z_y, y) + (-g(z_y) + \xi)'(w - y) - \xi'(w - y) \\
&\geq \Phi_{\delta,L}(y) + s_{\delta,L}(y)'(w - y),
\end{aligned} \tag{2.29}$$

where the first inequality follows from (2.18) and $z_w^\star = \operatorname{argmin}_{z\in\mathbb{R}^n} \mathcal{L}(z, w)$, the second inequality by the fact that $(-\mathcal{L}(z_y, y), -g(z_y))$ is a $(L_V\epsilon_z^2, L)$-oracle for Φ and the left part of (2.3), and the last inequality by Cauchy-Schwarz and (2.27).

On the other hand,

$$\begin{aligned}
\Phi(w) &\leq -\mathcal{L}(z_y, y) - g(z_y)'(w - y) + \tfrac{L}{2}\|w - y\|^2 + L_V\epsilon_z^2 \\
&\leq -\mathcal{L}(z_y, y) + (-g(z_y) + \xi)'(w - y) + \\
&\quad - \xi'(w - y) + \tfrac{L}{2}\|w - y\|^2 + L_V\epsilon_z^2 \\
&\leq -\mathcal{L}(z_y, y) + (-g(z_y) + \xi)'(w - y) + \\
&\quad + \tfrac{L}{2}\|w - y\|^2 + L_V\epsilon_z^2 + \alpha D\epsilon_\xi \\
&= \Phi_{\delta,L}(y) + s_{\delta,L}(y)'(w - y) + \tfrac{L}{2}\|w - y\|^2 + \\
&\quad + L_V\epsilon_z^2 + 2\alpha D\epsilon_\xi.
\end{aligned} \tag{2.30}$$

where the first inequality follows from the fact that $(-\mathcal{L}(z_y, y), -g(z_y))$ is a $(L_V \epsilon_z^2, L)$-oracle for Φ and the right part of (2.3), the third inequality by Cauchy-Schwarz, and the equality by (2.27). Therefore, $(\Phi_{\delta,L}(y), s_{\delta,L}(y))$ given by (2.27) is a (δ_α, L)-oracle for Φ on $\mathcal{Y}_\alpha$. □

Notice that the oracle error δ_α decreases with α, achieving its minimum value for $\alpha = 1$. Furthermore, the bounding of the dual feasible set is essential (cf. (2.25)), otherwise it would not be possible to bound quantities such as $\|w - y\|$, for any $w, y \geq 0$.

2.2.3 Primal Convergence Rates

Under the assumptions imposed by (2.22), (2.23), the ν-th iteration of the Inexact Dual Gradient Projection scheme applied to Problem (2.26) with the first-order oracle given by Proposition 2.2.1 is

$$y_{(\nu+1)} = \left[y_{(\nu)} + \tfrac{1}{L}(g(z_{(\nu)}) + \xi_{(\nu)}) \right]_{\mathcal{Y}_\alpha}, \tag{2.31}$$

with $z_{(\nu)}$, $\xi_{(\nu)}$ subject to

$$\begin{aligned} \|z_{(\nu)} - z^\star_{y_{(\nu)}}\| &\leq \epsilon_z, \\ \|\xi_{(\nu)}\| &\leq \epsilon_\xi. \end{aligned}$$

The Euclidean projection onto $\mathcal{Y}_\alpha$ is very easy to compute, since for $w \in \mathbb{R}^m$ we have

$$[w]_{\mathcal{Y}_\alpha} = \max\{\min\{w, \alpha d\}, 0\}.$$

We will next derive global convergence rates to primal optimality and primal feasibility for the ergodic primal iterates

$$\bar{z}_{(\nu)} \triangleq \frac{1}{\nu+1} \sum_{i=0}^{\nu} z_{(i)}. \tag{2.32}$$

First, the following lemma is needed.

Lemma 2.2.2. *Let $\{y_{(\nu)}, z_{(\nu)}\}$ be generated by iterating the Inexact Dual Gradient Projection algorithm* (2.31) *from any $y_{(0)} \in \mathcal{Y}_\alpha$.*

Then, for any $y \in \mathcal{Y}_\alpha$ and $\nu \in \mathbb{N}$, the following inequality holds

$$\mathcal{L}(\bar{z}_{(\nu)}, y) - V^\star \leq \frac{L}{2(\nu+1)} \|y - y_{(0)}\|^2 + \delta_\alpha. \tag{2.33}$$

Proof. For any $y \in \mathcal{Y}_\alpha$, one has

$$\begin{aligned} \Delta_{\delta,L}(y; y_{(\nu)}) &= \Phi(y) + \mathcal{L}(z_{(\nu)}, y_{(\nu)}) + \\ &\quad + \alpha D \epsilon_\xi + (g(z_{(\nu)}) - \xi_{(\nu)})'(y - y_{(\nu)}) \\ &\geq \Phi(y) + V(z_{(\nu)}) + \alpha D \epsilon_\xi + g(z_{(\nu)})' y + \\ &\quad - \|\xi_{(\nu)}\| \|y - y_{(\nu)}\| \\ &\geq \Phi(y) + \mathcal{L}(z_{(\nu)}, y), \end{aligned} \tag{2.34}$$

where the equality follows from (2.27), the first inequality by Cauchy-Schwartz, and the second inequality by (2.23) and the fact that $y_{(\nu)}$ belongs to $\mathcal{Y}_\alpha$.

Summing over $0, \ldots, \nu$,

$$\begin{aligned} \sum_{i=0}^{\nu} \Delta_{\delta,L}(y; y_{(i)}) &= (\nu+1)\Phi(y) + \sum_{i=0}^{\nu} \mathcal{L}(z_{(\nu)}, y) \\ &\geq (\nu+1)(\Phi(y) + \mathcal{L}(\bar{z}_{(\nu)}, y)), \end{aligned} \tag{2.35}$$

where the inequality follows by convexity of $\mathcal{L}(\cdot, y)$ for any fixed nonnegative $y \in \mathcal{Y}_\alpha$.

Dropping $\frac{L}{2}\|y - y_{(\nu+1)}\|^2$ from (2.9) since nonnegative, using (2.35), and the convexity of Φ, we obtain

$$\Phi(\bar{y}_{(\nu+1)}) + \mathcal{L}(\bar{z}_{(\nu)}, y) \leq \tfrac{L}{2(\nu+1)}\|y - y_{(0)}\|^2 + \delta_\alpha. \tag{2.36}$$

Finally, since $\Phi(\bar{y}_{(\nu+1)}) \geq -V^\star$ we prove (2.33). □

The next theorem gives the convergence rate towards primal feasibility for the ergodic primal iterates generated by the Inexact Dual Gradient Projection algorithm (2.31).

Theorem 2.2.3 (Bound on Primal Infeasibility)**.** *Let* $\{y_{(\nu)}, z_{(\nu)}\}$ *be generated by iterating ine Inexact Dual Gradient Projection algorithm* (2.31) *from any* $y_{(0)} \in \mathcal{Y}_\alpha$.

If $\alpha > 1$, *then for any* $\nu \in \mathbb{N}$ *the primal infeasibility is bounded as follows*

$$\|[g(\bar{z}_{(\nu)})]_+\|_\infty \leq \tfrac{\alpha^2}{\alpha-1}\tfrac{LD^2}{2(\nu+1)} + \delta_\alpha^g, \tag{2.37}$$

where

$$\delta_\alpha^g \triangleq \tfrac{1}{\alpha-1}L_V\epsilon_z^2 + \tfrac{\alpha}{\alpha-1}2D\epsilon_\xi.$$

Proof. Maximizing both sides of (2.33) with respect to $y \in \mathcal{Y}_\alpha$ and using

$$\max_{y\in\mathcal{Y}_\alpha} \mathcal{L}(\bar{z}_{(\nu)}, y) = V(\bar{z}_{(\nu)}) + \alpha \sum_{i=1}^{m} d^i[g^i(\bar{z}_{(\nu)})]_+ \tag{2.38}$$

we obtain

$$V(\bar{z}_{(\nu)}) - V^\star + \alpha \sum_{i=1}^{m} d^i [g^i(\bar{z}_{(\nu)})]_+ \leq \tfrac{LD^2}{2(\nu+1)}\alpha^2 + \delta_\alpha. \tag{2.39}$$

Now choose any $y^\star \in Y_\alpha^\star$ such that $y^\star \leq d$ (it exists by definition of d). By the saddle-point inequality, we have that

$$V^\star = \mathcal{L}(z^\star, y^\star) \leq \mathcal{L}(\bar{z}_{(\nu)}, y^\star),$$

or

$$V(\bar{z}_{(\nu)}) - V^\star \geq -g(\bar{z}_{(\nu)})' y^\star \geq -[g(\bar{z}_{(\nu)})]_+' y^\star. \tag{2.40}$$

Using (2.40) in (2.39), we arrive at

$$\sum_{i=1}^{m} (\alpha d^i - y^{i\star})[g^i(\bar{z}_{(\nu)})]_+ \leq \tfrac{LD^2}{2(\nu+1)}\alpha^2 + \delta_\alpha. \tag{2.41}$$

Since $\alpha > 1$ and $y^\star \leq d$,

(2.42)

$$\begin{aligned}\sum_{i=0}^{m} (\alpha d^i - y^{i\star})[g^i(\bar{z}_{(\nu)})]_+ &\geq (\alpha - 1) \min_{i \in \mathbb{N}_{[1,m]}} \{d^i\} \cdot \textstyle\sum_{i=1}^{m} [g^i(\bar{z}_{(\nu)})]_+ \\ &\geq (\alpha - 1) \|[g^i(\bar{z}_{(\nu)})]_+\|_\infty,\end{aligned}$$

where the last inequality follows from (2.24).

Therefore,

$$\|[g(\bar{z}_{(\nu)})]_+\|_\infty \leq \tfrac{\alpha^2}{\alpha-1} \tfrac{LD^2}{2(\nu+1)} + \tfrac{\delta_\alpha}{(\alpha-1)}. \tag{2.43}$$

□

Notice that, by tuning the parameter α, one is able to perform a trade-off in (2.37) between the constant of the $O(1/\nu)$ term determining the convergence rate to feasibility, and the maximum level of infeasibility that one is able to tolerate, asymptotically. As $\alpha \to \infty$, δ_α^g approaches its minimum, $2D\epsilon_\xi$, but at the same time $\frac{\alpha^2}{\alpha-1} \to \infty$, therefore slowing down the convergence rate.

By choosing $\alpha = 2$ (the one that minimizes $\frac{\alpha^2}{\alpha-1}$, thus achieving optimal convergence rate) we arrive at

$$\|[g(\bar{z}_{(\nu)})]_+\|_\infty \leq \tfrac{2LD^2}{\nu+1} + L_V\epsilon_z^2 + 4D\epsilon_\xi. \tag{2.44}$$

The next theorem gives the convergence rate towards primal suboptimality for the ergodic primal iterates generated by the Inexact Dual Gradient Projection algorithm (2.31).

Theorem 2.2.4 (Bound on Primal Suboptimality)**.** *Let* $\{y_{(\nu)}, z_{(\nu)}\}$ *be generated by iterating the Inexact Dual Gradient Projection algorithm* (2.31) *from any* $y_{(0)} \in \mathcal{Y}_\alpha$.

Then, the primal suboptimality is bounded as follows

$$V(\bar{z}_{(\nu)}) - V^\star \leq \tfrac{L}{2(\nu+1)}(\|y^\star\|^2 + \|y_{(0)}\|^2) + \delta_\alpha, \tag{2.45a}$$

$$V(\bar{z}_{(\nu)}) - V^\star \geq -\left(\tfrac{\alpha^2}{\alpha-1}\tfrac{LD^2}{2(\nu+1)} + \delta_\alpha^g\right) D. \tag{2.45b}$$

Proof. Choose $y^\star \in \mathcal{Y}_\alpha$ with $y^\star \leq d$. By substituting $y = \bar{y}^\star \geq 0$ in (2.33), where

$$\bar{y}^{i\star} = \begin{cases} y^{i\star}, & \text{if } g^i(\bar{z}_{(\nu)}) \geq 0, \\ 0, & \text{if } g^i(\bar{z}_{(\nu)}) < 0, \end{cases}$$

and dropping the term $g(\bar{z}_{(\nu)})'\bar{y}^\star$ since it is nonnegative, we obtain

$$V(\bar{z}_{(\nu)}) - V^\star \leq \tfrac{L}{2(\nu+1)}\|\bar{y}^\star - y_{(0)}\|^2 + \delta_\alpha. \tag{2.46}$$

Now notice that

$$\begin{aligned}\|\bar{y}^\star - y_{(0)}\|^2 &= \|\bar{y}^\star\|^2 - 2y'_{(0)}\bar{y}^\star + \|y_{(0)}\|^2 \\ &\leq \|y^\star\|^2 + \|y_{(0)}\|^2,\end{aligned} \tag{2.47}$$

since $\|\bar{y}^\star\| \leq \|y^\star\|$ and $2y'_{(0)}\bar{y}^\star \geq 0$.

Therefore, using (2.47) in (2.46), we prove at (2.45a).

To prove (2.45b), using (2.40) and Cauchy-Schwartz we obtain

$$V(\bar{z}_{(\nu)}) - V^\star \geq -\|[g(\bar{z}_{(\nu)})]_+\| \ \|y^\star\|. \tag{2.48}$$

Because of (2.37), and choosing any $y^\star$ with $\|y^\star\| \leq d$, we get (2.45b). □

Notice that the constant of the $O(1/\nu)$ term in (2.45a) is independent of α. In fact, if iterations (2.31) start from $y_{(0)} = 0$, then the cost $V(\bar{z}_{(\nu)})$ is always lower than $V^\star + \delta_\alpha$, the best achievable by the corresponding scheme asymptotically, as it is shown below. In that case, one has to worry only about feasibility.

Corollary 2.2.5. *Let $\{y_{(\nu)}, z_{(\nu)}\}$ be generated by iterating the Inexact Dual Gradient Projection algorithm* (2.31) *starting from $y_{(0)} = 0$.*

Then, the primal suboptimality is bounded as follows

$$V(\bar{z}_{(\nu)}) - V^\star \leq \delta_\alpha, \quad \forall \nu \in \mathbb{N}. \tag{2.49}$$

Proof. Simply put $y = 0$ in (2.33). □

2.2.4 Optimal Choice of α for Fixed Oracle Errors ϵ_z, ϵ_ξ

We will next derive the value of the user-defined parameter α that achieves the fastest convergence rate to an $(\varepsilon_V, \varepsilon_g)$-solution, given oracle parameters ϵ_z, ϵ_ξ.

For simplicity and without loss of generality, we assume that the initial iterate is equal to the zero vector, i.e., $y_{(0)} = 0$. In that case one should only worry about convergence to primal feasibility since, due to Corollary 2.2.5, $V(\bar{z}_{(\nu)}) - V^\star \leq \delta_\alpha$, for every $\nu \in \mathbb{N}_+$.

First, one must have $\varepsilon_V \geq \delta_\alpha$, or

$$\alpha \leq \frac{\varepsilon_V - L_V \epsilon_z^2}{2D\epsilon_\xi}. \tag{2.50}$$

Regarding ε_g, for sure it must be larger than $2D\epsilon_\xi$, the infimum of δ_α^g. Furthermore, by (2.37) it must satisfy $\varepsilon_g \geq \delta_\alpha^g$, implying that α must satisfy

$$\alpha > \frac{\varepsilon_g + L_V \epsilon_z^2}{\varepsilon_g - 2D\epsilon_\xi}. \tag{2.51}$$

Notice that the right hand-side of (2.51) is greater than one, since $\varepsilon_g > 2D\epsilon_\xi$. Equations (2.50), (2.51), pose the following restriction

$$\varepsilon_V > \frac{\varepsilon_g (L_V \epsilon_z^2 + 2D\epsilon_\xi)}{\varepsilon_g - 2D\epsilon_\xi}. \tag{2.52}$$

2.2.5 Bound of the Number of Iterations

The next theorem provides an upper bound on the number of iterations needed by the Inexact Dual Gradient Projection algorithm

(2.31) such that the obtained solution is at least $(\varepsilon_V, \varepsilon_g)$-optimal. This is a crucial certification to run the algorithm in embedded applications with tight real-time constraints.

The tightest upper-bound on the number of iterations is given by the next theorem.

Theorem 2.2.6 (Bound of the Number of Iterations)**.** *Suppose that* $\varepsilon_g > 2D\epsilon_\xi$*, and let* ε_V *satisfy* (2.52)*.*

Then, an $(\varepsilon_V, \varepsilon_g)$*-solution, i.e., a solution that guarantees a primal suboptimality lower than* ε_V *and a primal infeasibility lower than* ε_g*, is obtained by iterating the Inexact Gradient Projection Algorithm* (2.31) *from* $y_{(0)} = 0$ *with* $\alpha = \alpha^\star$*,*

$$\alpha^\star \triangleq \min\left\{\frac{2(\varepsilon_g + L_V\epsilon_z^2)}{\varepsilon_g - 2D\epsilon_\xi}, \frac{\varepsilon_V - L_V\epsilon_z^2}{2D\epsilon_\xi}\right\}, \tag{2.53}$$

no more than $\nu(\alpha^\star)$ *times, where* $\nu(\alpha)$ *is given by*

$$\nu(\alpha) = \frac{LD^2\alpha^2}{2(\varepsilon_g - 2D\epsilon_\xi)\alpha - 2(\varepsilon_g + L_V\epsilon_z^2)} - 1. \tag{2.54}$$

Proof. Let

$$\begin{aligned} c_1 &= LD^2, \\ c_2 &= 2(\varepsilon_g - 2D\epsilon_\xi), \\ c_3 &= 2(\varepsilon_g + L_V\epsilon_z^2), \end{aligned}$$

then, $\nu(\alpha) = \frac{c_1\alpha^2}{c_2\alpha - c_3}$.

The fastest convergence rate is achieved when

$$\alpha = \alpha^\star \triangleq \operatorname{argmin}\{\nu(\alpha) | \alpha \in [\underline{\alpha}, \overline{\alpha}]\},$$

where

$$\underline{\alpha} = \frac{c_3}{c_2}, \quad \overline{\alpha} = \frac{\varepsilon_V - L_V \epsilon_z^2}{2D\epsilon_\xi}.$$

Function $v(\alpha)$ is convex on $[\underline{\alpha}, \overline{\alpha}]$ since $v''(\alpha) = \frac{2c_1 c_3^2}{(c_2\alpha - c_3)^3} \geq 0$, for $\alpha \geq \underline{\alpha}$. Setting its derivative equal to zero, we find $\alpha = \frac{2c_3}{c_2}$, which is the first term in the min operator in (2.53). □

For the nominal case ($\epsilon_z = \epsilon_\xi = 0$), from (2.53) we obtain $\alpha^\star = 2$. If $2D\epsilon_\xi \ll \varepsilon_g$ and $\kappa_V \epsilon_z^2 \ll \min\{\varepsilon_g, \varepsilon_V\}$, then $\alpha^\star \approx 2$.

2.2.6 Maximum Admissible Oracle Errors ϵ_z, ϵ_ξ

Based on the results of Section 2.2.3, we will give explicit formulae of the maximum admissible oracle errors ϵ_z, ϵ_ξ as a function of solution accuracy $\varepsilon_V, \varepsilon_g$, and consequently of the number of iterations that are executed.

The question just posed is of significant importance in embedded optimization-based control, like MPC, for the following reason. Given hard real-time constraints dictated by hardware specifications and sampling time, as well as sufficiently small values for $\varepsilon_V, \varepsilon_g$ such that closed-loop stability is guaranteed (see [78]), one wants to determine the smallest allowable oracle precision (maximum allowable values for ϵ_z, ϵ_ξ) that achieves the aforementioned requirements.

As it will become clear in Section 2.3, the values of ϵ_z, ϵ_ξ can be linked to round-off errors occurring due to fixed-point arithmetic,

which in turn depend on the number of fractional bits chosen (cf. Section 1.3). The smaller the number of fractional bits is (i.e., the larger the maximum allowable oracle errors), the smaller the execution time and the power consumption of the hardware device will be.

Corollary 2.2.7. *Suppose that $\epsilon_\xi = \beta\epsilon_z$, for some $\beta > 0$, and set $\alpha = 2$, which is a good approximation for small ϵ_z, ϵ_ξ.*

Then, in order to converge asymptotically to an $(\varepsilon_V, \varepsilon_g)$-solution, the Inexact Gradient Projection Algorithm (2.31) *must have a maximum oracle error ϵ_z bounded by*

$$\epsilon_z < \sqrt{\frac{\varepsilon}{L_V} + \left(\frac{2D\beta}{L_V}\right)^2} - \frac{2D\beta}{L_V}, \tag{2.55}$$

where $\varepsilon = \min\{\varepsilon_g, \varepsilon_V\}$.

Proof. First, from (2.44), we have

$$\frac{2LD^2}{\nu+1} + L_V\epsilon_z^2 + 4D\beta\epsilon_z \leq \varepsilon_g. \tag{2.56}$$

By solving with respect to ϵ_z, we arrive at

$$\epsilon_z \leq \sqrt{\frac{\varepsilon_g}{L_V} + \left(\frac{2D\beta}{L_V}\right)^2 - \frac{2LD^2}{L_V(\nu+1)}} - \frac{2D\beta}{L_V}. \tag{2.57}$$

Due to Corollary 2.2.5, one must have $\delta_2 \leq \varepsilon_V$, or

$$\epsilon_z \leq \sqrt{\frac{\varepsilon_V}{L_V} + \left(\frac{2D\beta}{L_V}\right)^2} - \frac{2D\beta}{L_V}. \tag{2.58}$$

Letting $\nu \to \infty$ in (2.57), and taking into account that $\sqrt{\varepsilon/L_V + (2D\beta/L_V)^2}$ is increasing as a function of ε, we prove (2.55). □

Note that the assumption $\epsilon_\xi = \beta\epsilon_z$ does not cause a loos of generality in the case that the oracle errors are generated by fixed-point arithmetic, as it will be justified in Section 2.3.

Finally, it is worth mentioning that the maximum oracle error ϵ_z, that allows one to reach accuracy ε, decreases as $O(\sqrt{\varepsilon})$, slower than $O(\varepsilon)$ for $\varepsilon < 1$ (which is usually the case of interest).

2.3 Fixed-Point Dual Gradient Projection for Quadratic Programs

The theory presented in Section 2.2 allows to analyze the fixed-point implementation of the Dual Gradient Projection algorithm defined by (2.21) for strictly convex quadratic programs, such as the ones arising in Model Predictive Control applications. This is achieved by linking the errors committed due to fixed-point arithmetic to the oracle errors ϵ_z and ϵ_ξ.

Consider the optimization problem (2.16) with

$$V(z) = \tfrac{1}{2} z' Q z + c' z,$$

and let

$$\kappa_V = \lambda_{\min}(Q) > 0,$$

where $\lambda(Q)$ denotes the set of eigenvalues of the matrix Q.

The dual problem (modulo a sign change) is (2.1) with objective function

$$\Phi(y) = \tfrac{1}{2} y' H y + h' y,$$

and constraint set $\mathcal{Y} = \mathbb{R}^m_+$, where

$$H = G Q^{-1} G',$$
$$h = G Q^{-1} c + b.$$

Furthermore, the unconstrained solution $z_y^\star$ of the primal problem takes the analytical form

$$z_y^\star = E y + e, \tag{2.59}$$

where

$$E = -Q^{-1}G',$$
$$e = -Q^{-1}c.$$

Therefore, the Dual Gradient Projection iterations (2.21) with target solution infeasibility ε_g lead to Algorithm 1.

Algorithm 1 Dual Gradient Projection algorithm for Quadratic Programs.

Input: $E, e, G, b, L, \varepsilon_g$.

$y_{(0)} \leftarrow 0$
$\nu \leftarrow 0$
repeat
  $\nu \leftarrow \nu + 1$
  $z_{(\nu)} \leftarrow Ey_{(\nu)} + e$
  $g_{(\nu)} \leftarrow Gz_{(\nu)} - b$
  $y_{(\nu+1)} \leftarrow \left[y_{(\nu)} + \frac{1}{L} g_{(\nu)} \right]_+$
  $\bar{z}_{(\nu)} \leftarrow \frac{1}{\nu+1} \sum_{i=0}^{\nu} z_{(i)}$
until $\|[G\bar{z}_{(\nu)} - b]_+\|_\infty \leq \varepsilon_g$

Output: z

2.3.1 Fixed-point Implementation

Let Algorithm 1 be embedded on a fixed-point hardware with a scaling factor 2^{-p}, where $p \in \mathbb{N}_+$ is the number of fractional bits.

Assume that real numbers are represented in fixed-point by rounding to the closest value. Therefore, the resolution (i.e., the

smallest representable non-zero magnitude) of a fixed-point number is equal to $2^{-(p+1)}$.

In a fixed-point architecture, for a given $y \in \mathbb{R}^m$, instead of the gradient $\nabla\Phi(y) = -g(z_y^\star)$, where $z_y^\star$ is given by (2.59), we have access to an approximate gradient $\tilde{\nabla}\Phi(y)$ of the form (2.22), with

$$\begin{aligned} z_y &= \texttt{fi}(Ey + e), \\ \xi &= g(z_y) - \texttt{fi}(Gz_y - b). \end{aligned}$$

Due to error propagation in matrix-vector products (cf. (1.17)), the vectors ξ, z_y satisfy (2.23) with

$$\epsilon_z = 2^{-(p+1)} m\sqrt{n}, \tag{2.60a}$$

$$\epsilon_\xi = 2^{-(p+1)} n\sqrt{m}. \tag{2.60b}$$

Since $L = \frac{2}{\kappa_V}\|G\|^2$, by properly scaling the problem matrices we can assume that $L = 1$, therefore there is no round-off error in computing the product $\frac{1}{L} g_{(\nu)}$.

According to Proposition 2.2.1, the pair $(\Phi_{\delta_\alpha, L}(y), \tilde{\nabla}\Phi(y))$ given by (2.27) is a (δ_α, L)-oracle for Φ on $\mathcal{Y}_\alpha$.

The Inexact Dual Gradient Projection scheme (2.31) implemented on the fixed-point hardware platform becomes Algorithm 2. Notice the modification of the projection step which now requires the upper bound d on the dual variables, as defined in (2.24).

2.3.2 Guidelines for the Number of Fractional Bits

We now provide explicit bounds on the number of fractional bits required to grant convergence of Algorithm 2 to a target primal

Algorithm 2 Dual Gradient Projection algorithm for Quadratic Programs in fixed-point arithmetic.

Input: $E, e, G, b, L, \varepsilon_g, d$.

$y_{(0)} \leftarrow 0$
$\nu \leftarrow 0$
repeat
  $\nu \leftarrow \nu + 1$
  $z_{(\nu)} \leftarrow \texttt{fi}(Ey_{(\nu)} + e)$
  $g_{(\nu)} \leftarrow \texttt{fi}(Gz_{(\nu)} - b)$
  $y_{(\nu+1)} \leftarrow \max\{\min\{y_{(\nu)} + \frac{1}{L} g_{(\nu)}, \alpha d\}, 0\}$
  $\bar{z}_{(\nu)} \leftarrow \frac{1}{\nu+1} \sum_{i=0}^{\nu} z_{(i)}$
until $\|[G\bar{z}_{(\nu)} - b]_+\|_\infty \leq \varepsilon_g$

Output: z

suboptimal solution satisfying the bound on maximal primal infeasibility

$$\|[G\bar{z}_{(\nu)} - b]_+\|_\infty \leq \varepsilon_g.$$

Corollary 2.3.1 (Bound on the Number of Fractional Bits)**.** *Let* $\{z_{(\nu)}\}$ *be generated by Algorithm 2, with* $y_{(0)} = 0$*. Assume that real numbers are rounded to the closest fixed-point value.*

Then, the algorithm converges asymptotically to an $\left(\varepsilon_g, \varepsilon_V\right)$*-solution, with* $\varepsilon_g > 2D\epsilon_\xi$ *and* ε_V *satisfying* (2.52)*, if the number of fractional bits* p *is such that*

$$p \geq \log_2 \frac{m\sqrt{n}}{\sqrt{\frac{\varepsilon}{L_V} + \frac{n}{m}\left(\frac{2D}{L_V}\right)^2} - \sqrt{\frac{n}{m}}\frac{2D}{L_V}} - 1, \tag{2.61}$$

where $\varepsilon = \min\{\varepsilon_g, \varepsilon_V\}$.

Proof. Substitute (2.60a) and (2.60b) in (2.55). □

2.3.3 Guidelines for the Number of Integer Bits

Together with round-off errors, another key issue that arises while embedding computations on fixed-point architectures is the occurrence of overflow errors given by the limited range for number representation, as detailed in Section 1.3. In particular, if the number of bits for the integer part equals r, the computed variables can only assume values in $\left[-2^{r-1}, 2^{r-1}-1\right]$.

The next Corollary will set precise guidelines for choosing a number of integer bits that is sufficiently large to avoid overflows.

Corollary 2.3.2 (Bound on the Number of Fractional Bits)**.** *Let the iterations in Algorithm 2 be run on a fixed-point architecture with r bits for the integer part and $y_{(0)} = 0$.*

Then, occurrence of overflow errors is avoided if r is chosen such that

$$r \geq \log_2 \left(\max\left\{\hat{y}, \hat{z}, \hat{g}\right\} + 1\right) + 1, \tag{2.62}$$

where

$$\begin{aligned} \hat{y} &= \alpha \|d\|_\infty, \\ \hat{z} &= \|E\|_\infty \hat{y} + \|e\|_\infty, \\ \hat{g} &= \|G\|_\infty \hat{z} + \|b\|_\infty. \end{aligned}$$

Proof. A real number γ lies within the admissible range representable in fixed-point with r integer bits if $r \geq \log_2(\gamma + 1) + 1$. Since $\log_2(\cdot)$ is strictly increasing, r must be such that

$$r \geq \log_2(\max\{\|y_{(\nu)}\|_\infty, \|z_{(\nu)}\|_\infty, \|g_{(\nu)}\|_\infty\} + 1) + 1,$$

for all $\nu \in \mathbb{N}$.

Using Algorithm 2, one obtains $\|y_{(\nu)}\|_\infty \leq \hat{y}$, $\|z_{(\nu)}\|_\infty \leq \hat{z}$, $\|g_{(\nu)}\|_\infty \leq \hat{g}$, for all ν. Again, since $\log_2(\cdot)$ is strictly increasing we arrive at (3.10). □

2.4 Simulations

2.4.1 Sample Evolutions

The aim of this simulation is to show sample infeasibility and suboptimality evolutions generated by Algorithm 2 for multiple fixed-point precisions p, and compare them with the floating-point, double-precision case.

Simulations are performed in MATLAB R2012b equipped with the Fixed-Point Toolbox v.3.6 on a Mid-2012 Macbook Pro Retina running OSX 10.8.2.

We iterate Algorithm 2 for a fixed number of steps ν on the dual of randomly generated QP problems, with 10 optimization variables and 20 constraints. Figure 2.1 shows the experimental convergence of the primal infeasibility $\|[g(\bar{z}_{(\nu)})]_+\|_\infty$ and the primal suboptimality $|V(\bar{z}_{(\nu)}) - V^\star|$ for the averaged iterates. In both figures, computation in double precision is compared with fixed-point precision for $p = \{2, 4, 6\}$. Simulation for the double-precision evolution was performed according to the Dual Gradient Projection algorithm (2.2).

Curve shapes are consistent with the theoretical bounds given by Theorem 2.2.3 and Theorem 2.2.4. In addition, the algorithm presents a remarkable robustness to round-off errors; 4-bits and 6-bits curves already show a convergence comparable with the double-precision case. This fact is of particular interest for embedded implementations, since computational burden and power

consumption are heavily dependent on the number of bits used to represent numbers [59].

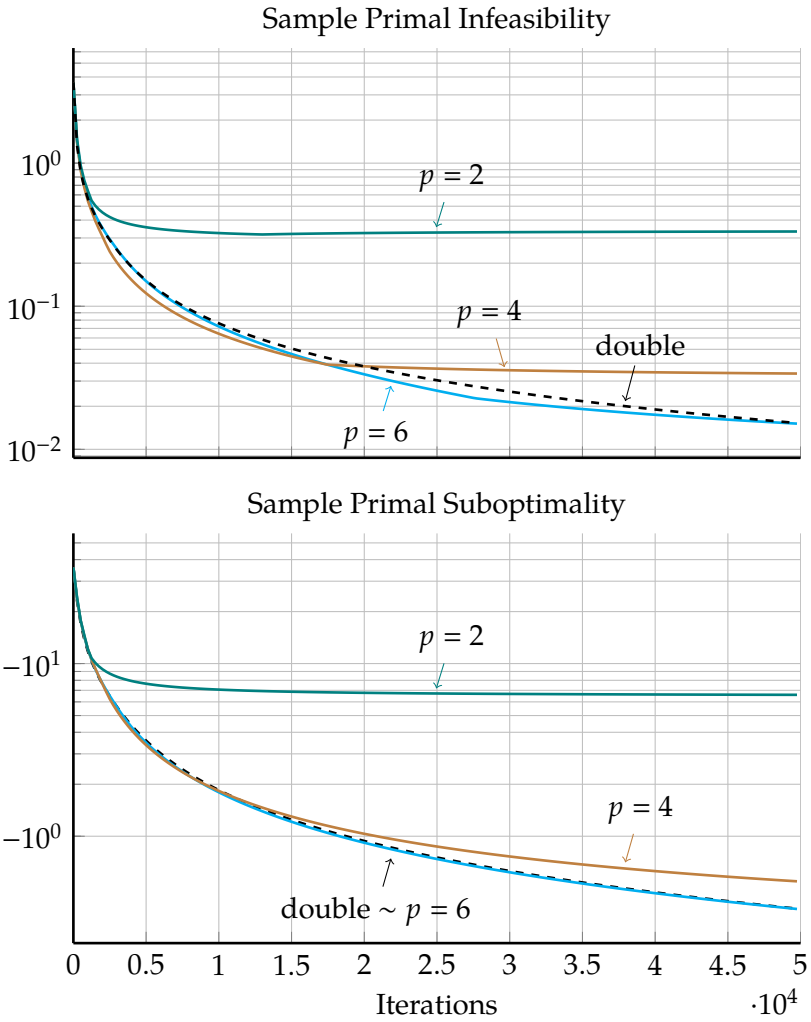

Figure 2.1: Sample primal infeasibility (*top* plot) and suboptimality (*bottom* plot) evolutions, obtained by iterating Algorithm 2 on a random QP problem for varying number of fractional bits $p = (2, 4, 6)$. Results are compared with the 64-bit floating-point double precision (dashed lines).

2.4.2 Infeasibility and Suboptimality Bounds

The purpose of the second simulation is to test the tightness for the primal infeasibility and suboptimality bounds given by Theorem 2.2.3 and Theorem 2.2.4, respectively.

The analysis is performed on a worst-case scenario, running Inexact Dual Gradient projection iterations (2.31) with $\|\xi_{(\nu)}\| = \epsilon_\xi$ to solve 100 randomly generated QP problems, with 10 optimization variables and 20 constraints.

The goal is to compare error bound terms δ_α and δ_α^g with the practical asymptotic values of the primal infeasibility and suboptimality as $\nu \to \infty$. Different trials are ordered for increasing values of D, term proportional to the constraint set diameter.

Simulation results, depicted in Figure 2.2, show an acceptable tightness for bounds, as they exceed the practical values by a factor between 3.33 and 8.32 for infeasibility, and between 3.05 and 5.74 for suboptimality. In addition, the linear (for infeasibility) and quadratic (for suboptimality) theoretical dependencies on D are reflected in the experiment results.

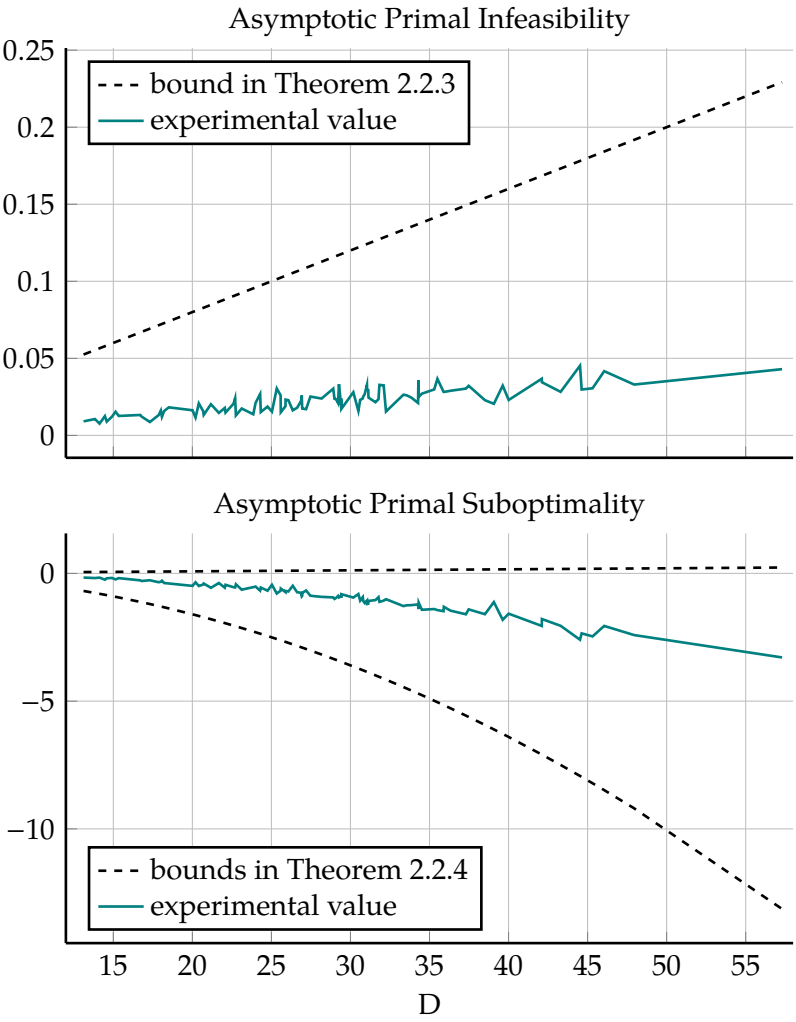

Figure 2.2: Asymptotic primal infeasibility (*top* plot) and suboptimality (*bottom* plot) compared to theoretical bounds, ordered for increasing D. Experimental values (*solid* lines) are obtained running Algorithm 2 on random QPs. Theoretical bounds (*dashed* lines) are given by Theorem 2.2.3 and Theorem 2.2.4.

2.4.3 Target Infeasibility

Figure 2.3 shows simulation results on the practical number of iterations needed to reach a target primal infeasibility for a sample, random QP problem with 5 variables and 10 constraints.

A comparison is made between the double-precision Gradient Projection algorithm with iterations as in (2.2), and Algorithm 2 running supported by fixed-point arithmetic with 2-bit precision. Results are in accordance with the theoretical results of Theorem 2.2.3 since, for finite precision, the number of iterations grows to infinity when target infeasibility reaches a critical value, different from zero.

Figure 2.4 shows how the theoretical guaranteed primal infeasibility is related to the number of fractional bits, based on the guidelines in Corollary 2.3.1. Note that inequality constraints for the primal QP have been normalized, such that all elements of b are equal to one. Results show how such guidelines are not over-conservative and are applicable in practical scenarios (e.g., 13-bit precision for a guaranteed infeasibility below 10%, which is enough for many applications).

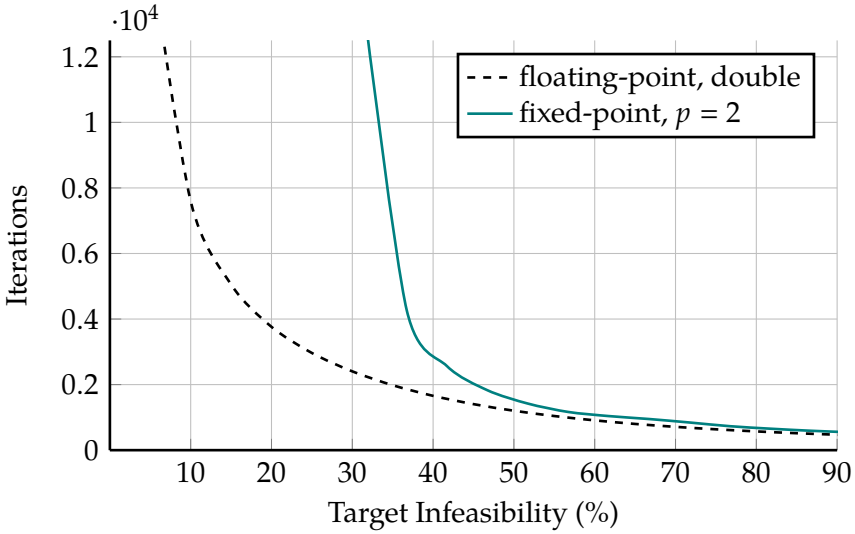

Figure 2.3: Iterations to target infeasibility. Experimental iteration counts obtained by running Algorithm 2 to reach different levels of target infeasibility, supported by double precision floating-point arithmetic (*dashed* line) and fixed-point arithmetic with 2 bits for the fractional part (*solid* line).

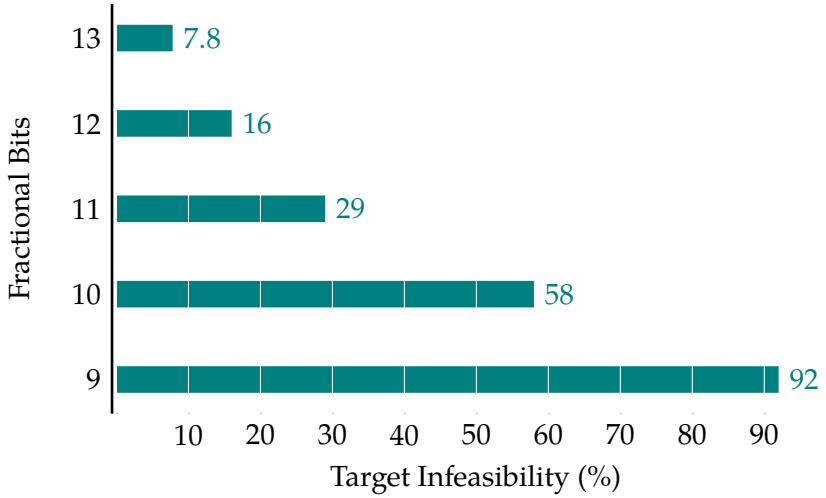

Figure 2.4: Fractional bits for target infeasibility. Entries are computed according to the theoretical guidelines on the required fractional bits to achieve a guaranteed target infeasibility given by Corollary 2.3.1.

2.4.4 Bounds on Iteration Count

This simulation is performed to test the tightness of the theoretical bound on the number of iterations given by Theorem 2.2.6.

We let Algorithm 2 run on MATLAB R2012b and Fixed-Point Toolbox v.3.6 on a Mid-2012 Macbook Pro Retina running OSX 10.8.2 to solve various random QP problems (sizes are equal to 4 and 8 for the primal and the dual, respectively). In Figure 2.5 the practical number of iterations needed to reach varying target infeasibility is compared the theoretical bound; different colors and markers are chosen for different QP problems.

Results show that theoretical bounds are about one order of magnitude larger than actual iterations. In addition to this, two interesting properties emerge from the plot:

1. within one single QP plot, the two curves of practical and theoretical values have similar shapes;

2. if the practical number of iterations needed to solve a first QP is larger than the number to solve a second QP, then also the first QP theoretical iteration bound is larger then second QP bound.

Those facts increase the confidence in the accuracy of bound formulation.

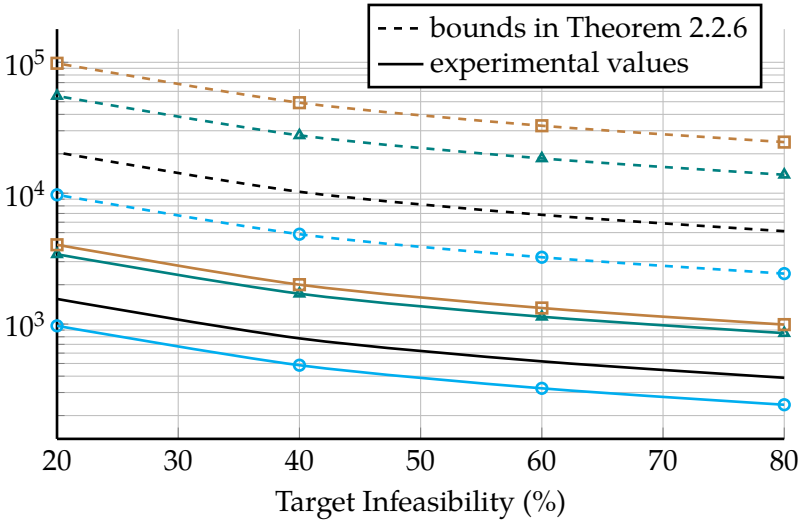

Figure 2.5: Number of iterations for target infeasibility in four different random problems (identified by different colors and markings). Theoretical bounds (*dashed* lines) are given by Theorem 2.2.6. Experimental values (*solid* lines) are obtained by iterating Algorithm 2.

2.4.5 Masses Serially Connected Example

The purpose of this example is to test the fixed-point Dual Gradient Projection Algorithm 1 as a QP solver for a Model Predictive Control design.

The reference physical system is composed by a series of M elements, each of mass m, connected by springs with constant k and dampers with constant c. The first and the last element are connected to fixed walls, and actuators are placed between each pair of masses to exert tensions. The state-space model is derived by a set of first-principle ODEs, where the system states are the displacements and velocities of the masses and the inputs are the tensions exerted by the actuators. This is a modification of the example proposed in [131].

By letting x^i be the displacement of the i-th mass, and u^i the tension exerted by the actuator placed right of the i-th element, the system can be modeled by the following set of ODEs:

$$
\begin{cases}
m\ddot{x}^i &= -kx^i - k\left(x^i - x^{i+1}\right) + \\
&\quad -c\dot{x}^i + c\left(\dot{x}^i - \dot{x}^{i+1}\right) + u^i \qquad (i = 1) \\
m\ddot{x}^i &= k\left(x^{i-1} - x^i\right) - k\left(x^i - x^{i+1}\right) + c\left(\dot{x}^{i-1} - \dot{x}^i\right) + \\
&\quad -c\left(\dot{x}^i - \dot{x}^{i+1}\right) + u^i - u^{i-1} \qquad (i \in \mathbb{N}_{[2,M-1]}) \\
m\ddot{x}^i &= k\left(x^{i-1} - x^i\right) - kx^i - \\
&\quad +c\dot{x}^i + c\left(\dot{x}^{i-1} - \dot{x}^i\right) - u^{i-1} \qquad (i = M)
\end{cases}
\tag{2.63}
$$

Simulations have been performed in MATLAB R2012b on a

Mid-2012 Macbook Pro Retina running OSX 10.8.2. The QP problem is built forcing the system states to be in $[-4, 4]$ and inputs in $[-1, 1]$, and setting the stage cost equal to $l(x, u) = \frac{1}{2}\,(x'W_x x + u'W_u u)$ with W_x and W_u as identity matrices. The prediction horizon N equals 10, and the sampling time $0.5\,s$.

Figure 2.6 shows the evolution of position and velocity for the second mass out of a total of 3 masses. The reference dashed lines (double precision) are obtained closing the loop with a Model Predictive Controller supported by IBM ILOG CPLEX v.12.4 as solver of the QP optimization problem. For the remaining plots, the controller is instead supported by fixed-point Dual Gradient Projection algorithm implemented with Fixed-Point Toolbox v.3.6; two simulations are performed varying precision to 2 and 6 bits.

Results show a remarkable robustness of the closed-loop evolutions with respect to fixed-point precision. Position and velocity trajectories of the 6-bit simulation are almost indistinguishable with the double precision simulation, while for the 2-bit case a small divergence shows up. This behavior is consistent with what shown in the sample infeasibility and suboptimality evolutions of Figure 2.1.

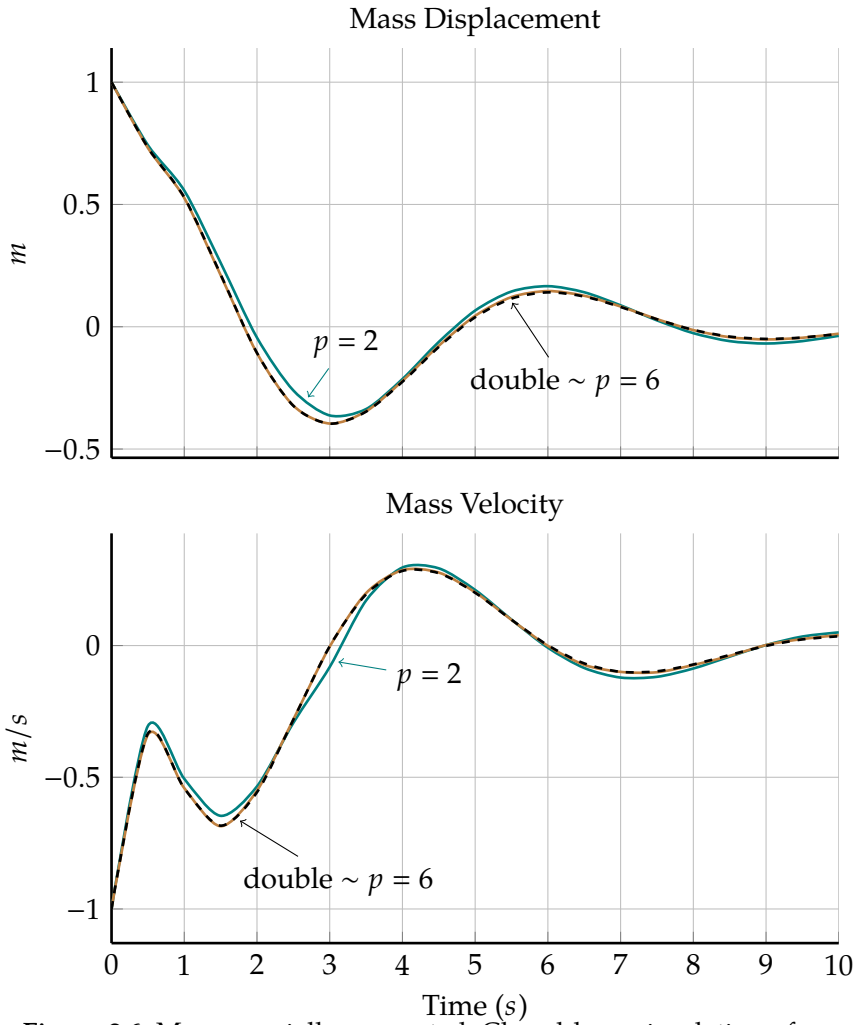

Figure 2.6: Masses serially connected. Closed-loop simulation of a Model Predictive Controller with QP solver based on Algorithm 2 attached to a set of three masses serially connected by springs and dampers. The plot depicts displacement (*top*) and velocity (*bottom*) of the middle mass.

Chapter 3

Proximal Newton Methods in Finite Precision Arithmetic

This chapter details aspects in the fixed-point implementation of Proximal Newton Methods. After a brief introduction on the problem setup in Section 3.1, Section 3.2 details the proximal Newton procedure, and Section 3.3 investigates the impact of low-precision computations, due to fixed-point arithmetic, on the algorithm execution. In Section 3.4 two algorithm optimizations are discussed, namely preconditioning and division-free computations. Finally, Section 3.5 shows the results of simulations exploring computational complexity, solution accuracy, and an aircraft control application.

3.1 Problem Setup

Consider the following input-constrained, discrete-time, linear time-invariant system

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t \\ \underline{u} &\leq u_t \leq \overline{u}, \ \forall t \in \mathbb{N}, \end{aligned} \tag{3.1}$$

where $x_t \in \mathbb{R}^{n_x}$ is the state vector, $u_t \in \mathbb{R}^{n_u}$ the input vector, and A, B are real-valued matrices of appropriate dimensions.

From (3.1) it is possible to formulate a receding-horizon optimal control problem over N prediction steps in the following condensed form

$$\begin{aligned} \min_{z} \quad & \frac{1}{2} z'Qz + c'z \\ \text{subject to} \quad & g(z) \leq 0, \end{aligned} \tag{3.2}$$

where $z \in \mathbb{R}^n$ denotes the sequence of inputs over the control horizon ($n = N_c n_u$), $c \in \mathbb{R}^n$ is computed from the state measurement, $Q \in \mathbb{R}^{n \times n}$ is a symmetric positive definite (SPD) matrix, $F_x \in \mathbb{R}^{n \times n_x}$ and and $g(z) : \mathbb{R}^n \to \mathbb{R}^m$ is an affine mapping.

The procedure to compute (3.2) starting from a Model Predictive Control formulation applied to (3.1) is detailed in Section 1.2.3. Peculiarity of the present formulation is that constraints are posed only in the form of box constraints on the control input. Hence, the mapping $g(z)$ takes the simplified form

$$g(z) = Gz - w$$

where

$$G = \begin{bmatrix} \mathbf{I}^n \\ -\mathbf{I}^n \end{bmatrix}, \quad w = \begin{bmatrix} \bar{U} \\ -\underline{U} \end{bmatrix}$$

3.2 Proximal Newton Algorithm

A computationally efficient Proximal Newton method for convex, possibly non-smooth, composite optimization problems has been recently introduced by [119].

The proposed approach is attractive for embedded applications since it retains the low number of iterations typical of Newton-based methods, and concurrently lowers per-iteration complexity requiring the solution of a linear system on a reduced-order problem.

Algorithm 3 is the Proximal Newton method applied to solve the quadratic programming problem (3.2). It is based on the idea that problem (3.2) is equivalent to minimizing the real-valued, continuously differentiable convex function

$$F_\gamma(z) = V(z) - \frac{\gamma}{2}\|\nabla V(z)\|^2 + \frac{1}{2\gamma}\|z - \gamma\nabla V(z) - [z - \gamma\nabla V(z)]_{\underline{z}}^{\bar{z}}\|^2, \tag{3.3}$$

where V denotes the cost function of problem (3.2), provided that the parameter γ is smaller than $1/\lambda_{\max}(Q)$.

The procedure requires as inputs the Hessian matrix Q, the linear term c, and the box constraints $(\underline{z}, \bar{z})$ on the optimization vector. Additional tuning parameters are the value of γ and the line search parameter $\sigma \in (0, \frac{1}{2})$, e.g. $\sigma = 10^{-4}$. It should be remarked here that the performance of the algorithm is insensitive to the choice of these two parameters.

Under exact arithmetic, the algorithm guarantees convergence to the unique optimal solution $z^\star$ in a finite number of iterations.

Algorithm 3 Proximal Newton algorithm for box-constrained QP

Input: $Q, c, \underline{z}, \overline{z}, \sigma \in (0, \frac{1}{2}), \gamma < 1/\lambda_{\max}(Q), z_0 \in \mathbb{R}^n$

for $\nu = 1, 2, \ldots$ **do**

`compute Newton direction`

$\beta \leftarrow \left\{ i | \underline{z}^i < z^i_{(\nu)} - \left(Q^i z_{(\nu)} + c^i \right) < \overline{z}^i \right\}$

$d^i \leftarrow - \left(z^i - \left[z^i_{(\nu)} - \gamma \left(Q^i z_{(\nu)} + c^i \right) \right]^{\overline{z}^i}_{\underline{z}^i} \right), i \notin \beta$

$d^\beta \leftarrow -(Q^{\beta\beta})^{-1} \left(Q^\beta z^\beta_{(\nu)} + c^\beta + Q^{\beta\neg\beta} d^{\neg\beta} \right)$

`perform line search`

$\alpha \leftarrow 1$

while $F_\gamma \left(z_{(\nu)} + \alpha d \right) - F_\gamma \left(z_{(\nu)} \right) > \sigma \alpha \nabla F_\gamma \left(z_{(\nu)} \right)' d$ **do**

$\alpha \leftarrow \frac{\alpha}{2}$

end while

$z_{(\nu+1)} \leftarrow z_{(\nu)} + \alpha d, k \leftarrow k + 1$

end for

Output: z^*

In practice the algorithm can be stopped as soon as

$$\left\| z_{(\nu)} - \left[z_{(\nu)} - \gamma \left(Q z_{(\nu)} + c \right) \right]^{\overline{z}}_{\underline{z}} \right\| \leq \gamma \sqrt{2 \mu \epsilon} \,,$$

where μ is (lower bound on) the smallest eigenvalue of the positive definite matrix Q, for some given error tolerance $\epsilon > 0$. This condition guarantees that

$$V(\hat{z}_{(\nu)}, x) - V^\star \leq \epsilon$$

for

$$\hat{z}_{(\nu)} = \left[z_{(\nu)} - \gamma \left(Q z_{(\nu)} + c\right)\right]_{\underline{z}}^{\overline{z}},$$

where $V^\star$ is the optimal value of problem (3.2).

3.3 Fixed-Point Proximal Newton Algorithm

3.3.1 Round-off Error Analysis

We now analyze the effects of round-off errors (cf. Section 1.3.2) occurring when executing Algorithm 3 supported by a fixed-point number representation with p bits for the fractional part.

Assume that input data is represented exactly, i.e. $\texttt{fi}(\eta) = \eta$ for $\eta = \left\{Q, c, \underline{z}, \bar{z}, \epsilon, \sigma\right\}$.

The goal is to bound the round-off error accumulated on the optimization vector z during the execution of one algorithm iteration.

We proceed by bounding the error on the Newton direction $\texttt{fi}(d) - d$ when executing steps 2-4 in Algorithm 3. The values of d are updated either according to step 3, by computing a matrix-vector product and an Euclidean projection, or according to step 4, where a matrix-vector product and the solution of a linear system is required. Computing the projection does not cause additional errors; on the other hand (as detailed shortly) this happens for the linear system. Therefore, we analyze the error on d by considering the worst-case scenario where $\beta = \{1, 2, \ldots, N_c n_u\}$ and d is updated by solving a linear system whose dimension is equal to the number of variables of Problem (3.2).

From standard linear algebra results (see, e.g., [60, 132]), given a perturbed linear system in the form

$$Q\left(x + \tilde{x}\right) = b + \tilde{b},$$

it holds that

$$\begin{aligned} \|\tilde{x}\| &\leq \kappa(Q)\frac{\|\tilde{b}\|}{\|b\|}\|x\| \\ &\leq \kappa(Q)\|\tilde{b}\|\|Q^{-1}\|, \end{aligned} \tag{3.4}$$

where $\kappa(Q) = \|Q\|\|Q^{-1}\|$ is the condition number of matrix Q.

In our case,

$$\begin{aligned} b &= Qz + q, \\ \tilde{x} &= d - \texttt{fi}\,(d)\,. \end{aligned} \tag{3.5}$$

Substituting (3.5) into (3.4) and taking into account the bound (1.17) on round-off error propagation in matrix-vector multiplications, we obtain

$$\|d - \texttt{fi}\,(d)\,\| \leq \kappa(Q)\|Q^{-1}\|2^{-(p+1)}\left(N_c n_u\right)^{3/2}. \tag{3.6}$$

The line search operation does not cause round-off errors to accumulate on z, since it only involves halving the stepsize α.

Finally, in line (9) of Algorithm 3, z acquires the perturbation on d, bounded by (3.6), plus a final round-off due to multiplication by α. Therefore, the bound on round-off errors accumulated on the optimization vector in one iteration becomes

$$\|z - \texttt{fi}\,(z)\,\| \leq \left(1 + \kappa(Q)\|Q^{-1}\|\left(N_c n_u\right)^{3/2}\right)2^{-(p+1)}. \tag{3.7}$$

3.3.2 Avoiding Overflow Errors

Overflow errors occur when trying to store a number outside of the representable range $[-2^r, 2^r - 1]$ (cf. Section 1.3.1). To avoid them, r must be chosen large enough such that every computed value

during the execution of the algorithm lies within the admissible range. Provided that r is chosen large enough to represent all static problem data, we now give lower bounds for it that guarantee the representability of variable data as well, namely z and d vectors.

Let ε_d and ε_z be the right-hand sides of inequalities (3.6) and (3.7), respectively. Then,

$$\begin{aligned} \|\texttt{fi}\,(z)\,\|_\infty &= \|\texttt{fi}\,(z) - z + z\|_\infty \\ &\leq \|z\|_\infty + \|\texttt{fi}\,(z) - z\|_\infty \\ &\leq \max\left\{\|\underline{z}\|_\infty, \|\overline{z}\|_\infty\right\} + \varepsilon_z \\ &\triangleq \hat{z}. \end{aligned} \tag{3.8}$$

Moreover,

$$\begin{aligned} \|\texttt{fi}\,(d)\,\|_\infty &= \|\texttt{fi}\,(d) - d + d\|_\infty \\ &\leq \|d\|_\infty + \|\texttt{fi}\,(d) - d\|_\infty \\ &\leq \|Q^{-1}\|_\infty \left(\|Q\|_\infty \hat{z} + \|q\|_\infty\right) + \varepsilon_d \\ &\triangleq \hat{d}. \end{aligned} \tag{3.9}$$

Therefore, the execution of Algorithm 3 supported by fixed-point number representation with r bits for the integer part does not cause overflow or underflow errors if

$$r \geq \log_2\left(\max\left\{\hat{z}, \hat{d}\right\} + 1\right) + 1, \tag{3.10}$$

where $\hat{z}$ and $\hat{d}$ are defined in (3.8) and (3.9), respectively.

3.4 Optimization of the Algorithm

In this section we analyze key issues regarding algorithm implementation in fixed-point arithmetic and propose procedures to deal with them and optimize computation efficiency.

3.4.1 Preconditioning

The accuracy of the result, when computing the solution of a perturbed linear system, is sensible to the *conditioning* of the problem, as shown in (3.4).

This sensitivity is directly reflected in the accuracy of the overall solution of the QP given by Algorithm 3, where an ill-conditioned Hessian cause a significant degeneration of its performance. However, the impact of this phenomenon can be reduced by *preconditioning* the problem, i.e., finding a change of coordinates

$$\bar{z} = \bar{P}^{-1} z$$

such that the condition number of the Hessian of the transformed problem, $\kappa\left(\bar{P}Q\bar{P}\right)$, is smaller than the condition number of the Hessian of the original problem (3.2), $\kappa(Q)$.

Ideally $\bar{P}$ should solve the problem

$$\begin{aligned} \min \quad & \kappa\left(PQP\right) \\ \text{subject to} \quad & P \in \mathbb{R}^{n\times n} \text{ is PSD.} \end{aligned} \tag{3.11}$$

In general, the preconditioner resulting from (3.11) will not preserve the box structure of the feasible set $\mathcal{Z}$. This can be avoided by forcing P diagonal.

Solving problem (3.11), as shown in [133, Sec 2.2.3], is equivalent to solving the following generalized eigenvalue problem (GEVP)

$$\begin{aligned} \min \quad & \xi \\ \text{subject to} \quad & S > 0, \\ & S \text{ diagonal}, \\ & \xi \geq 0, \\ & S \leq Q \leq \xi S, \end{aligned} \tag{3.12}$$

and picking $P = S^{-1/2}$.

In order to avoid the inclusion of a bilinear matrix inequality constraint, problem (3.12) can be reformulated (for an alternative approach, see [134]) by means of the Schur complement as follows

$$\begin{aligned} \max \quad & \delta \\ \text{subject to} \quad & S > 0, \\ & S \text{ diagonal}, \\ & \delta \geq 0, \\ & \begin{bmatrix} S & \delta I \\ \delta I & Q^{-1} \end{bmatrix} \geq 0, \end{aligned} \tag{3.13}$$

Figure 3.1 highlights the effects of preconditioning on 1000 random QP problems of variable sizes in the range $[20, 200]$ and variable Hessian condition numbers. The top histogram plots the variation of the condition number expressed as $\kappa(PQP))/\kappa(Q)$. The bottom histogram plots the variation of the solution errors expressed as $\|\zeta^* - z^C\|/\|z^* - z^C\|$, where ζ^* is the solution of the preconditioned problem and z^C is the solution given by the solver

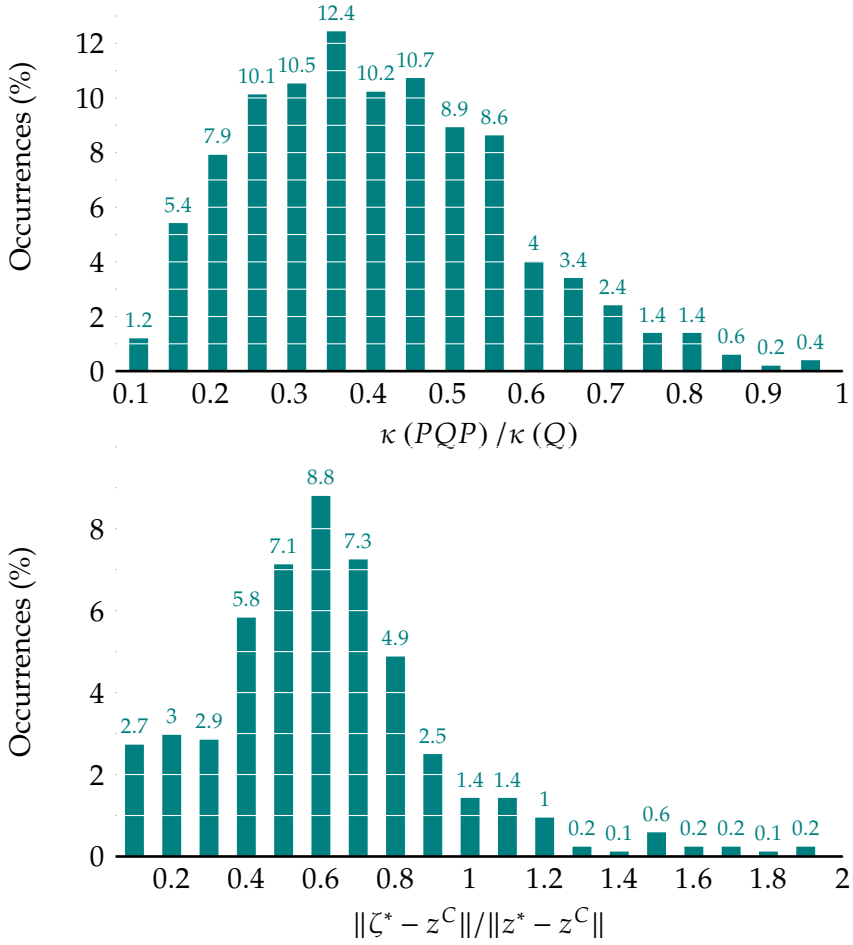

Figure 3.1: Impact of preconditioning on the Hessian condition number (*top* plot) and solution error (*bottom* plot) in terms of distributions of the conditioned to non-conditioned ratios over 1000 random QP problems.

of *CPLEX*.

Results show that by preconditioning one obtains an average 56.4 % reduction of condition number of the Hessian and 34.7 % of the solution error. Note that preconditioning can be computed offline. However, preconditioning is not beneficial for all the problems: in 10.6 % of the cases the solution error is increased. This happens since a modification of the Hessian may cause a change in the constraint activation behavior during algorithm execution, and therefore the size of the linear system that has to be solved at each iteration.

3.4.2 Division-free Computations

An efficient way to solve the linear system in step 4 of Algorithm 3 is by means of a Cholesky factorization followed by forward and backward substitution. However, these procedures require the computation of the reciprocal and the square root of the diagonal entries of the matrix. Performing a division on most embedded devices requires more cycles than performing additions and multiplications; therefore, the presence of divisions can cause a degeneration of overall performances.

A possible approach for division-free computations is the following.

1. Scale the QP problem such that the Hessian has all entries in the range $[-1, 1]$. This causes that all its diagonal entries fall in the range $(0, 1]$, since it is SDP.

2. Store into the computing device assigned to execute the algorithm a pre-computed look-up table containing $1/\sqrt{\xi}$, with ξ covering all the fixed-point values in the range $(0, 1]$ for the selected precision.

3. To evaluate the inverse of a desired value simply access the look-up table with the value itself as index, and square the result.

This solution constitutes a trade-off knob between computation speed and memory occupancy, increasing the latter by $w \cdot 2^p/8$ bytes.

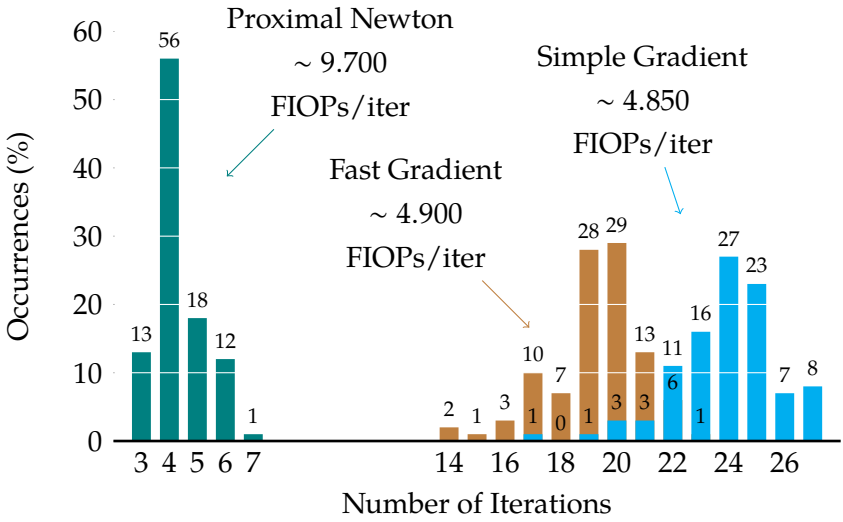

Figure 3.2: Proximal Newton methods compared to Gradient Methods (number of iterations). Histogram of the distribution for the required number of iterations to solve 100 random QPs.

3.5 Simulations

3.5.1 Computational Complexity

We first compare the computational complexity for the implementation in fixed-point arithmetic of Algorithm 3 against gradient-based methods (see [54, 56]). The reason of this choice is due to the interest arising recently in embedded implementations of first-order algorithms, which have been proven to perform well on low-precision arithmetic, as detailed in Section 1.4 and Chapter 2.

Table 3.1: Estimation of the exponential coefficient for single iteration (ρ_i) and overall algorithm (ρ_A) complexities with respect to problem size (with 95% confidence bounds).

Method	$\rho_{\mathbf{i}}$	$\rho_{\mathbf{A}}$
Proximal Newton	2.32 ± 0.08	2.544 ± 0.12
Fast Gradient	2.01 ± 0.01	2.19 ± 0.09
Simple Gradient	2.01 ± 0.01	2.21 ± 0.12

Figure 3.2 shows the histogram distribution of the number of iterations required to reach a target solution quality. Computations are performed in fixed-point arithmetic with word length $w = 32$ bits and fraction length $p = 16$ bits. Results are based on 100 random QPs of size $n = 50$ and show that iteration count lies in the range $[3, 7]$ for proximal Newton, $[14, 23]$ for fast gradient, and $[17, 27]$ for simple gradient methods. However, the estimated average fixed-point operations (multiplications and additions) performed per-iteration is roughly double for the Newton method compared to gradient-based ones.

Table 3.1 shows how per-iteration and overall computation complexities scale with number of variables n. The goal is to estimate the exponent ρ when fitting the actual fixed-point operations as a function of the problem size, according to the relation $a \cdot n^{\rho}$.

The theoretical per-iteration complexity bound for the gradient methods is $O(n^2)$, due to the computation of a matrix-vector product, and this is confirmed by the simulations. Newton-based methods are instead bounded by $O(n^3)$ due to the solution of a

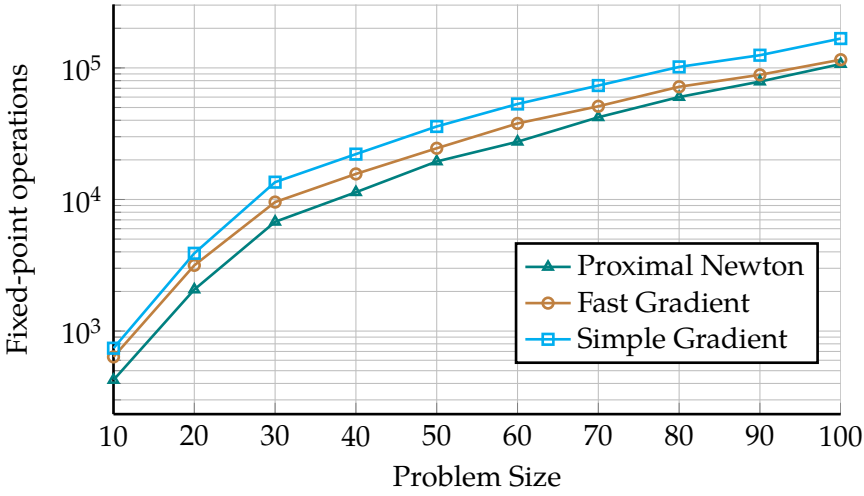

Figure 3.3: Proximal Newton methods compared to Gradient Methods (number of fixed-point operations) to solve 200 random QPs of increasing size.

linear system; however, the effective complexity growth is estimated as $n^{2.32}$. This happens because the proposed algorithm requires only the solution of a linear system of reduced order.

Finally, Figure 3.3 shows a comparison of the overall fixed-point operations executed for matrix computations, estimated over 200 random QPs of different sizes ranging from $n = 10$ to $n = 100$ variables. Results show that the proposed implementation is indeed computationally efficient; however, the benefits compared to gradient methods decrease for larger problems, and eventually vanish due to the higher exponential dependency on n (cf. Table

3.1).

3.5.2 Solution Accuracy

This simulation shows how solution accuracy varies with the number of fractional bits chosen in the fixed-point representation, comparing Algorithm 3 (Proximal Newton) with fast and simple gradient methods.

By "solution accuracy" we mean the relative discrepancy ϵ_z with the solution z^C obtained from the state-of-the-art solver of *CPLEX* running on double-precision arithmetic, that is

$$\epsilon_z = \frac{\|z^* - z^C\|}{\|z^C\|}. \tag{3.14}$$

Figure 3.4 depicts average solution errors, computed as in (3.14), over 100 QPs of size 10 and 100 QPs of size 120, varying the number of fractional bits from $p = 4$ to $p = 16$. In compliance with the bound in (3.7) on the round-off error accumulation, we observe an exponential decrease with respect to p. Results show a remarkable robustness of the proposed implementation when running in finite-precision arithmetic. Nevertheless, it is more susceptible to variations on problem size, as reflected by the term $(Nn_u)^{3/2}$ in (3.7).

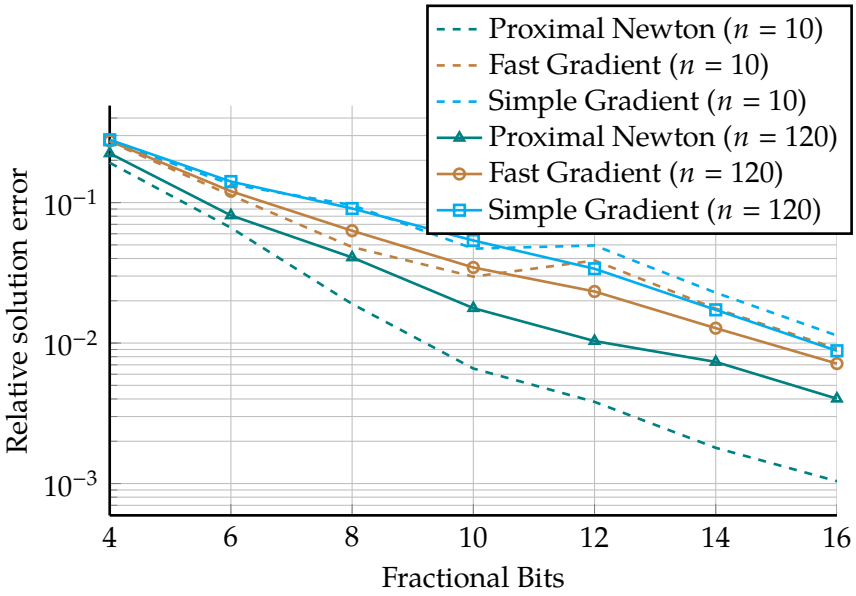

Figure 3.4: Proximal Newton methods compared to Gradient Methods (solution accuracy) to solve two random QPs of different sizes $n = (10, 120)$, varying the number of fractional bits.

3.5.3 Control of a F16 Aircraft Example

We verify the closed-loop behavior of a simulated physical system connected to a predictive controller that relies on an implementation in fixed-point arithmetic of Algorithm 3 to solve on-line the QP problem. The goal is to regulate roll and pitch angles of an AFTI-F16 aircraft.

The aircraft dynamics is described by a linearized, continuous-time, state-space model (cf. [135]) in the form

$$\begin{aligned} \overbrace{\begin{bmatrix} \dot{v}(t) \\ \dot{\alpha}(t) \\ \ddot{\theta}(t) \\ \dot{\theta}(t) \end{bmatrix}}^{\dot{x}(t)} &= A \overbrace{\begin{bmatrix} v(t) \\ \alpha(t) \\ \dot{\theta}(t) \\ \theta(t) \end{bmatrix}}^{x(t)} + B \overbrace{\begin{bmatrix} u_e(t) \\ u_f(t) \end{bmatrix}}^{u(t)} \\ \overbrace{\begin{bmatrix} \alpha(t) \\ \theta(t) \end{bmatrix}}^{y(t)} &= Cx(t), \end{aligned} \tag{3.15}$$

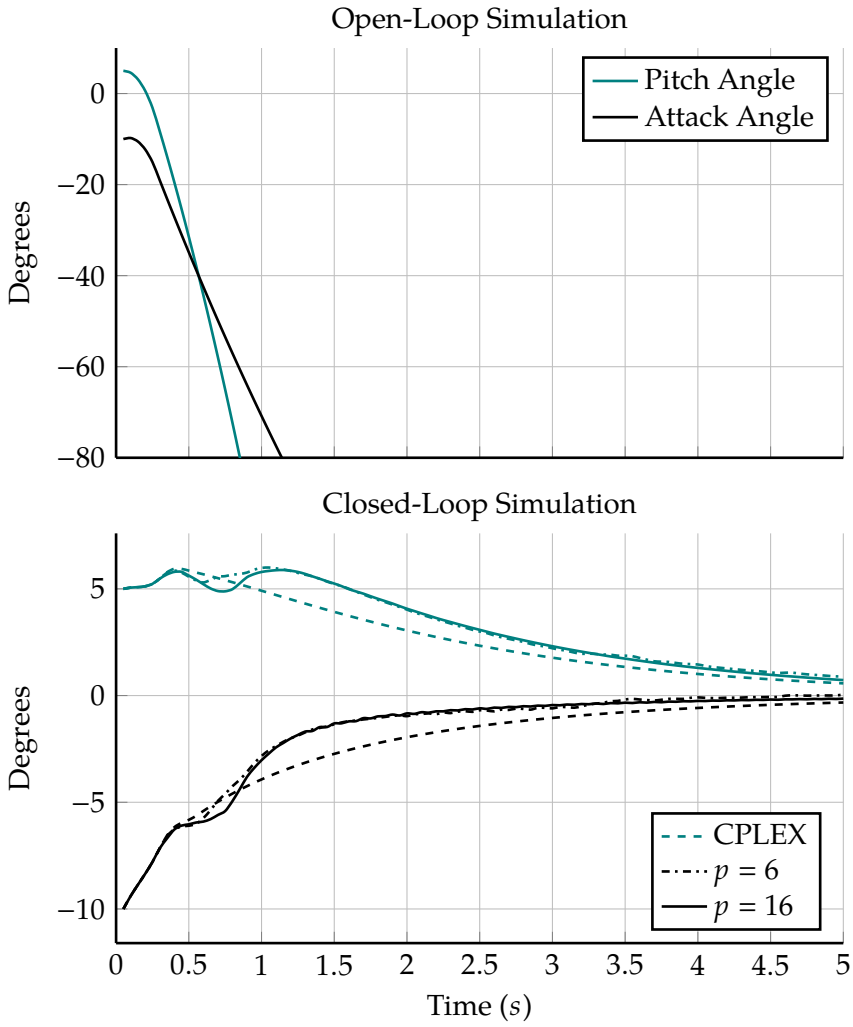

Figure 3.5: Open-loop (*top* plot) and closed-loop (*bottom* plot) trajectories of pitch and roll angles in F16 aircraft simulation. Closed-loop trajectories are compared for floating-point solver of CPLEX and Proximal Newton Algorithm in fixed-point arithmetic with $p = (6, 16)$.

with

$$A = \begin{bmatrix} -1.51 \times 10^{-2} & -6.056 \times 10^{1} & 0 & -3.217 \times 10^{1} \\ - \times 10^{-4} & -1.341 & 9.929 \times 10^{-1} & 0 \\ -1.8 \times 10^{-4} & 4.325 \times 10^{1} & -8.693 \times 10^{-1} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

$$B = \begin{bmatrix} -2.516 & -1.313 \times 10^{1} \\ -1.689 \times 10^{-1} & -2.514 \times 10^{-1} \\ -1.725 \times 10^{1} & -1.576 \\ 0 & 0 \end{bmatrix},$$

$$C = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

where v (ft/s) is the forward velocity, α (degrees) is the angle of attack, θ (degrees) is the pitch angle, and u_e, u_f (degrees) are the elevator and flaperon angles, respectively. Both inputs are constrained in the $[-25°, +25°]$ range.

As shown by the top plot in Figure 3.5 the open-loop response of the system described in (3.15) is unstable. The open-loop poles are $(-7.6636, -0.0075 \pm 0.0556j, 5.453)$.

The control objective is to regulate pitch and attack angles to zero. A Model Predictive is Controller is designed using as prediction model (3.15) discretized with sampling time $Ts = 50\ ms$. Prediction horizon is $N = 10$, control horizon is $N_c = 3$. Weight matrices are set to $W_y = \mathbf{I}^2$ for measured outputs and $W_u = 0.1 \cdot \mathbf{I}^2$ for control inputs. No terminal weight W_N is imposed.

The bottom plot in Figure 3.5 shows the closed-loop trajectories

of attack and pitch angles starting from $x_0 = \begin{bmatrix} 0 & -10 & 2 & 5 \end{bmatrix}'$. Two implementations with fixed-point arithmetic of Algorithm 3, with 16 and 6 bits for the fractional part, are compared with the trajectories obtained acquiring control inputs from the state-of-the-art solver of *CPLEX*, running in double-precision arithmetic.

Results show that a predictive controller supported by the proposed implementation with fixed-point arithmetic is able to stabilize the system. Note that the controller sampling time should be fast enough such that oscillations in system inputs due to the quantization of the QP solution fall outside the process bandwidth.

Chapter 4

Experimental Tests

Experimental tests on various hardware are reported in this chapter. Section 4.1 shows implementations on low-power, general purpose processors in the ARM Cortex family, for both the Gradient Projection and the Proximal Newton methods. Then, Section 4.2 details the implementation of Gradient Projection method on FPGA.

4.1 Embedded Optimization on ARM Cortex

4.1.1 The ARM Cortex-M3 Processing Unit

The *ARM* architecture is a family of processing units widely popular in embedded systems due to the low power consumption with respect to the computational capabilities. Its application scope ranges from microcontrollers to portable devices. It is estimated that ARM-based processors are at the core of 75% of embedded systems.

ARM devices belong to the category of *reduced instruction set computers* (RISC), which favors simple and linear architectures. A key distinction with *complex instruction set computers* (such as Intel x86 devices) is that the first allow access to the memory only with basic *load* and *store* instructions that transfer data to and from the processor registers.

The *Cortex M3* is a 32-bit RISC processor of the ARM family specifically developed for highly deterministic real-time applications, such as in automotive control systems. The specific model employed for the experimental tests is the *Atmel SAM3X8E*, which operates at a maximum speed of 84 MHz and comes with 512 KB of flash memory and 100 KB of RAM.

4.1.2 Gradient Projection Methods on ARM Cortex

The purpose of this experimental test is to assess the performance of a QP solver based on the Dual Gradient Projection algorithm detailed in Chapter 2 when implemented on a low-power, low-cost microcontroller based on the 32-bit Atmel SAM3X8E ARM Cortex-M3 processing unit. The test emphasizes in appraising the benefits coming from performing computation in fixed-point arithmetic.

The microcontroller was assigned to solve random QP problems of increasing size, ranging from 10 to 60 primal variables and 20 to 120 primal constraints. The algorithm was stopped upon reaching a suboptimal solution bounded by 10% primal infeasibility.

Table 4.1 shows the results when a fixed-point number representation is adopted, with 8 bits for the decimal part and 7 bits for the integer part. For each problem size we report convergence time, average time per iteration (TPI) and size of the binary code; the latter plays an important role in embedded applications, where usually a limited amount of memory is available.

In order to evaluate the performance enhancements coming from fixed-point computations, we repeated all the hardware simulations after switching to floating-point number representation. Results are reported in Table 4.1 together with their fixed-point counterpart, showing how the floating-point implementation is about 4 times slower than the fixed-point one, and up to twice as bigger in code size.

Figure 4.1 depicts the relationship between problem size, ex-

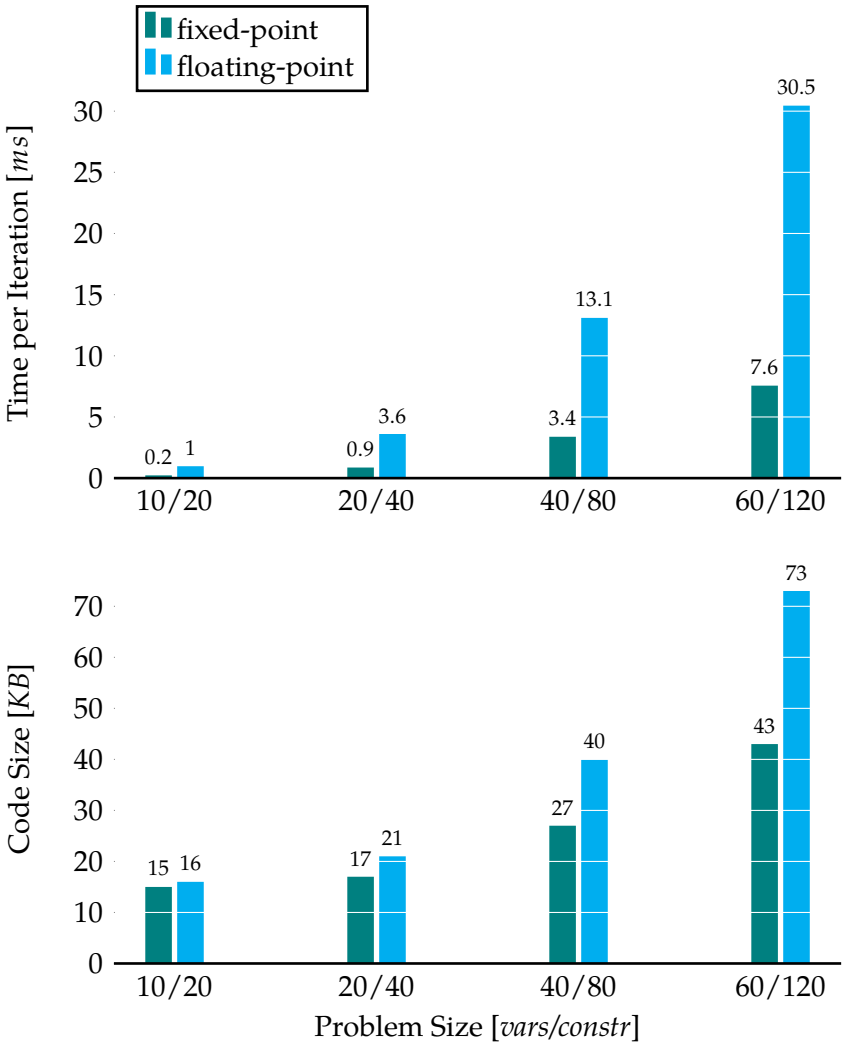

Figure 4.1: Dual Gradient Projection Algorithm on ARM Cortex-M3. Comparison between fixed-point and floating-point implementations in terms of computation speed (*top* plot) and memory occupancy (*bottom* plot).

Table 4.1: Dual Gradient Projection Algorithm on ARM Cortex-M3.
Comparison between fixed-point and floating-point implementations.

	Fixed-Point			Floating-Point		
Size [*vars/constr*]	**Time** [ms]	**Time/Iter** [μs]	**Size** [KB]	**Time** [ms]	**Time/Iter** [μs]	**Size** [KB]
10/20	22.9	226	15	88.6 (+287%)	974 (+331%)	16 (+6.67%)
20/40	52.9	867	17	220.1 (+316%)	3608 (+316%)	21 (+23.5%)
40/80	544.9	3382	27	2240 (+311%)	13099 (+287%)	40 (+48.1%)
60/120	1519.8	7561	43	5816 (+283%)	30450 (+303%)	73 (+69.8%)

pressed as number of variables and constraints, and the time per iterations (top plot), and code size (bottom plot). Results are compared between the implementations supported by fixed-point and floating-point arithmetic. It is important to notice how the gain in performance obtained by switching from floating-point to fixed-point arithmetic increases as the problem becomes larger in size, especially in terms of code size..

This implementation highlights some of the key advantages of the fixed-point format: the computational burden and the memory footprint are lowered, especially on devices lacking hardware support for floating-point operations. However, it has to be noted that the flexibility of the floating-point representation is lost, causing reduced precision and range; the choice of the optimal format is therefore dependent on the specific application and computing capabilities. Especially in the case of chipsets equipped with a floating-point unit (FPU), the benefits from switching to fixed-point arithmetic may be substantially reduced.

4.1.3 Proximal Newton Methods on ARM Cortex

This section details the experimental test results when an implementation of the Proximal Newton Method (cf. Algorithm 3 and Chapter 3) is deployed on the low-power, low-cost, ARM-based Cortex-M3 general-purpose processing unit, similarly to what shown for the Gradient Projection Method in Section 4.1.2.

The device was assigned to solve a set of random QPs of increasing size, ranging from 10 primal variables and 20 constraints, up

to 80 primal variables and 160 constraints. Note that they were not the same QPs of Section 4.1.2. The algorithm was coded both in floating-point arithmetic (word length of 32 bits) and fixed-point arithmetic (word length of 16 bits, of which 8 bits for the fractional part).

Table 4.2 shows the average experimental results: for each problem size, we report overall computation time for the fixed-point arithmetic and floating-point arithmetic versions; similarly, we report code size for the compiled binary. Between brackets in the floating-point columns we also include the variation with respect to the fixed-point counterpart.

Table 4.2 shows that switching from floating- to fixed-point arithmetic causes the computation time and code size to become up to 4 and 2 times smaller, respectively. Advantages become more evident as the number of variables increases; for problems with $n \geq 70$ the floating point version is not able to converge at all, due to lack of memory. Those results are also visualized graphically in Figure 4.2.

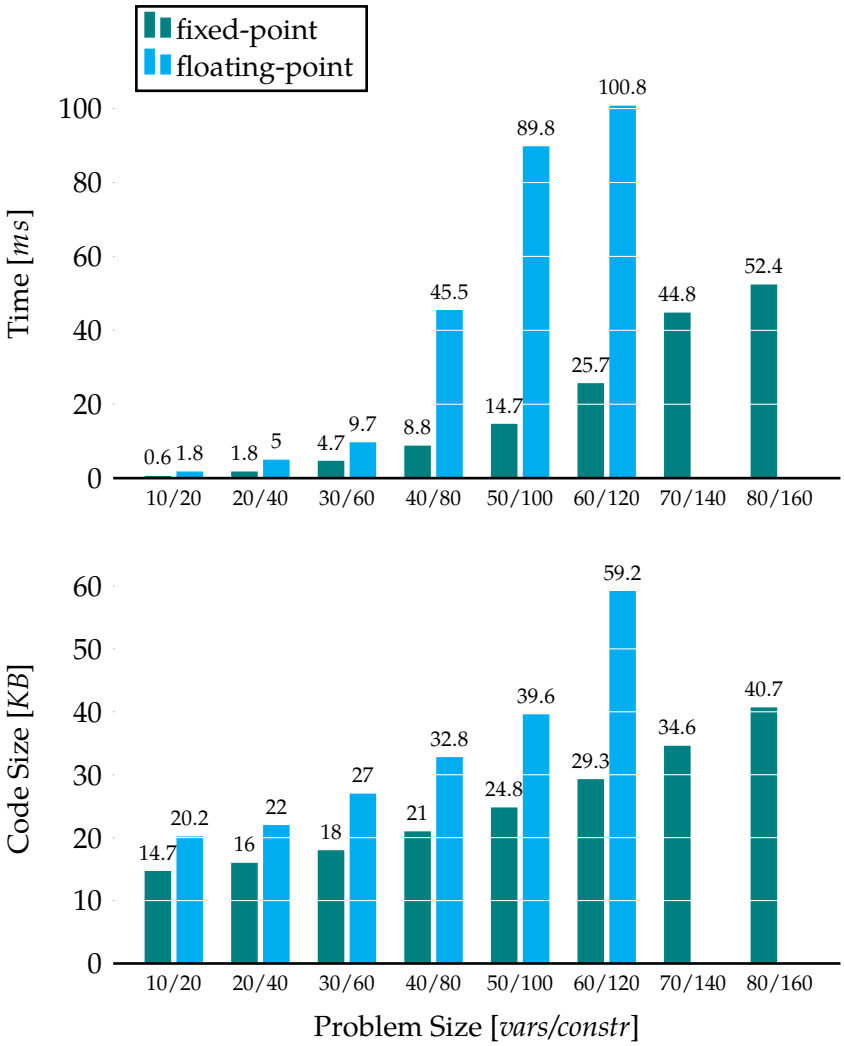

Figure 4.2: Proximal Newton Algorithm on ARM Cortex-M3. Comparison between fixed-point and floating-point implementations in terms of computation speed (*top* plot) and memory occupancy (*bottom* plot).

Table 4.2: Proximal Newton Algorithm on ARM Cortex-M3.
Comparison between fixed-point and floating-point implementations.

	Fixed-Point		Floating-Point	
Size [*vars/constr*]	**Time** [*ms*]	**Size** [*KB*]	**Time** [*ms*]	**Size** [*KB*]
10/20	0.6	14.7	1.8 (+200%)	20.2 (+37.4%)
20/40	1.8	16	5 (+178%)	22 (+37.5%)
30/60	4.7	18	9.7 (+106%)	27 (+50%)
40/80	8.8	21	45.5 (+417%)	32.8 (+56.2%)
50/100	14.7	24.8	89.8 (+511%)	39.6 (+59.7%)
60/120	25.7	29.3	100.8 (+292%)	59.2 (+102%)
70/140	44.8	34.6	*n/a*	*n/a*
80/160	52.4	40.7	*n/a*	*n/a*

4.2 Embedded Optimization on FPGA

4.2.1 Introduction to FPGA Devices

FPGAs (*Field Programmable Gate Arrays*) are chipsets where connections between multiple logic blocks can be programmed by the user "on the field" to perform the desired computations. Functionalities of FPGAs range from basic glue-logic to advanced, high-speed data acquisition and processing, as in high-energy and nuclear physics experiments.

Usually, developing and producing application-specific integrated circuits (ASICs) requires a first phase of circuit synthesis on appropriate CAD software, and then a physical printing on silicon wafers in high-technology factories; the whole process may require an initial investment in the order of million of dollars. Moreover, it is not possible to correct possible bugs and errors discovered after the design phase. On the contrary, FPGA devices can be implemented with a very short developing cycle. They can be reprogrammed as needed, and present almost no starting costs.

An FPGA device (see Figure 4.3) is a chip composed mainly by arrays of logic blocks and routing channels, with every single logic block generally constituted of a 4-input Lookup Table (LUT), a Flip-Flop, and one output; finally, a set of input/output blocks complete the schematic. These are the basic components of every integrated circuit; the only things missing are the interconnections between them. Here comes the key advantage of the FPGA technology: those interconnections can be programmed (see Figure 4.4) as required

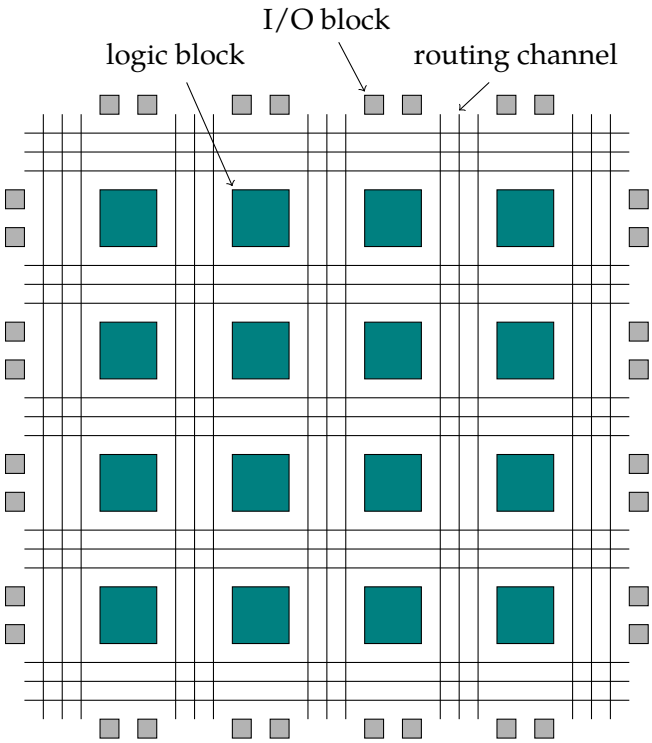

Figure 4.3: FPGA structure, where arrays of logic blocks (*green*) are connected to Input/Output blocks (*grey*) and routing channels.

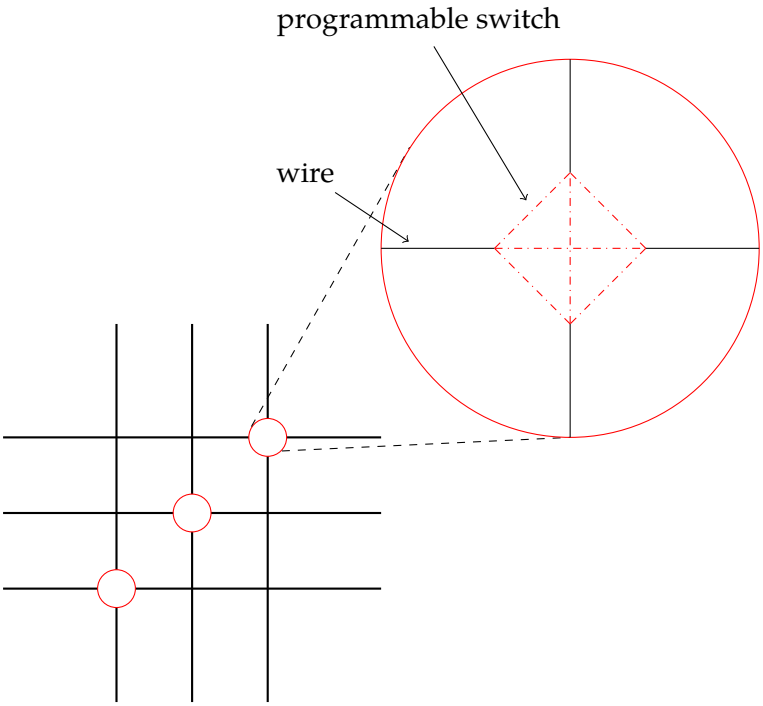

Figure 4.4: Detail of FPGA programmable switches, where wire connections can be manipulated to build custom circuits.

by simply feeding a serial bit stream to the device after a reset.

Modern FPGAs have a number of logic cells that ranges from hundreds of thousands to millions, with latencies of a few nanoseconds and data throughputs in the order of Gb/s, opening for the implementation of high-performance, complex data processing solutions; a good overview of the modern scientific applications is covered in [136].

Predictive Control on FPGAs Reconfigurable computing devices such as FPGAs has become subject of a great deal of research, in particular starting from the late 1990s when affordable, high-performance (for the standards of the time) solutions appeared on the market. This was driven by the key feature of efficiently perform computations at hardware-level, while retaining much of the flexibility of a software solution.

An interesting survey exploring the hardware and software aspects of reconfigurable computing machines and the issues involved in runtime during program execution can be found in [137]; similar topics and issues are covered in [138].

Starting from the late 2000s, an increasing research interest started to show up for FPGA-based Model Predictive Control applied to high-bandwidth applications; this was driven by the availability of new-generation FPGAs with computing powers sufficient to evaluate the solution for the constrained optimization problem within strict real=time constraints.

In [65, 139], the authors describe a rapid prototyping environment for MPC on FPGA, exploring the possibilities of parallel computations. In [140, 141], a mixed software-hardware formulation is proposed for the embedded controller, where the bulk of the MPC matrix computations are performed in hardware and the rest in a general-purpose microprocessor. A comparison between active-set and interior-point solvers is carried out in [142]. Finally, other notable applications of Model Predictive Controllers on FPGAs include [66–68, 143].

4.2.2 Fixed-Point Dual Gradient Projection on FPGA

In this section we report details on the implementation of the Fixed-Point Dual Gradient Projection method (cf. Algorithm 2 and Chapter 2) on a Field Programmable Gate Array, along with a performance evaluation when solving quadratic programs.

The FPGA circuit design was performed according to the graphical approach proposed by *Xilinx System Generator for DSP*, part of the *Xilinx ISE Design Suite* v14.7. With these tools, the single circuit blocks as multipliers, accumulators, and memories can be placed and connected to each other in the *Simulink* environment. The compiler will then automatically generate the corresponding *VHDL* or *Verilog* code for the FPGA platform of choice.

In the proposed implementation the tests were performed targeting a *Xilinx Kintex 7-xc7k480t* device. This is part of the $28nm$ Kintex generation, and comes with $478K$ logic cells and 1920 DSP slices. We choose to target this device for its fair balance between low cost, low power consumption, and appropriate performance.

Figure 4.5 shows the top-level view of the QP solver for a sample problem with 2 primal variables and 4 dual variables. The problem size has been chosen small to guarantee the readability of the circuit scheme.

The two Matrix-Vector Multiplication (MVM) units perform algorithm steps 5-6. The output of the first MVM are the primal variables (*black*). The accumulator units (*cyan* blocks) multiply the gradient, obtained as output of the second MVM, by the inverse

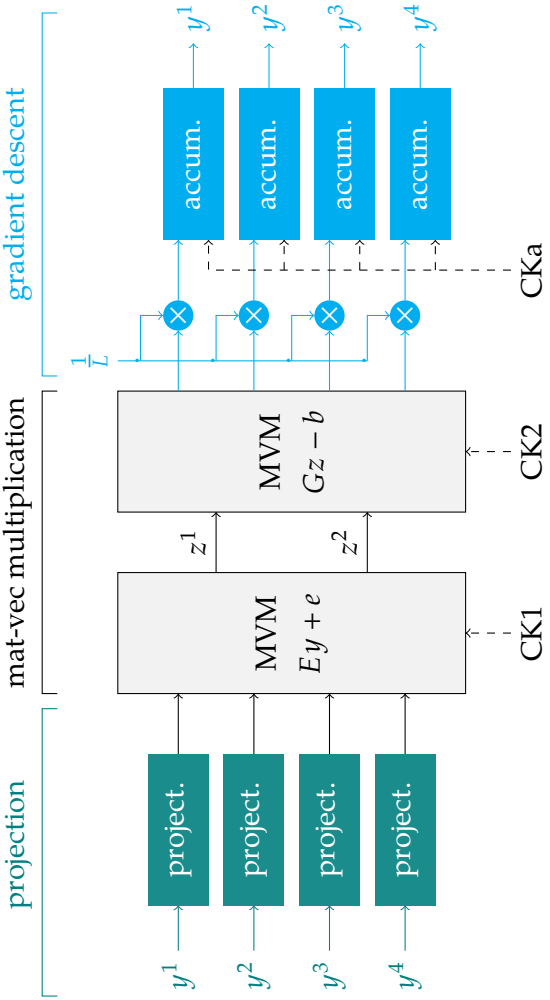

Figure 4.5: Dual Gradient Projection on FPGA. Top-level overview.

of the Lipschitz constant and accumulate the result, obtaining the dual variables vector prior to the projection step. Finally, the array of projection units n the left (*green* blocks) perform the projection, completing step 7 of the algorithm. The behavior of clock signals (*dashed* lines) will be detailed shortly.

Figure 4.6 shows the inside of a Matrix-Vector Multiplication unit. To maximize device compatibility, this block is designed up to the single multipliers/adders/accumulators units, instead than using higher-level DSP blocks. This approach requires to individually place blocks for each variable; to automate this process we developed scripts to build MVM units with arbitrarily large number of variables. For the sake of clarity, in Figure 4.6 is depicted a small MVM unit that computes $y = Ax + b$, where $A \in \mathbb{R}^{2\times4}$, $x \in \mathbb{R}^4$, and $y, b \in \mathbb{R}^2$. Computations are performed in row-wise parallel fashion.

The path of the computed variables is depicted in green, and develops as follows:

1. a switch, governed by the control logic, selects consecutively the input vector x entries;

2. the current x is split into multiple parallel paths, and each of them is multiplied by the corresponding value of the A matrix rows;

3. the result is then accumulated obtaining the inner products between the input vector and the matrix rows, and added to

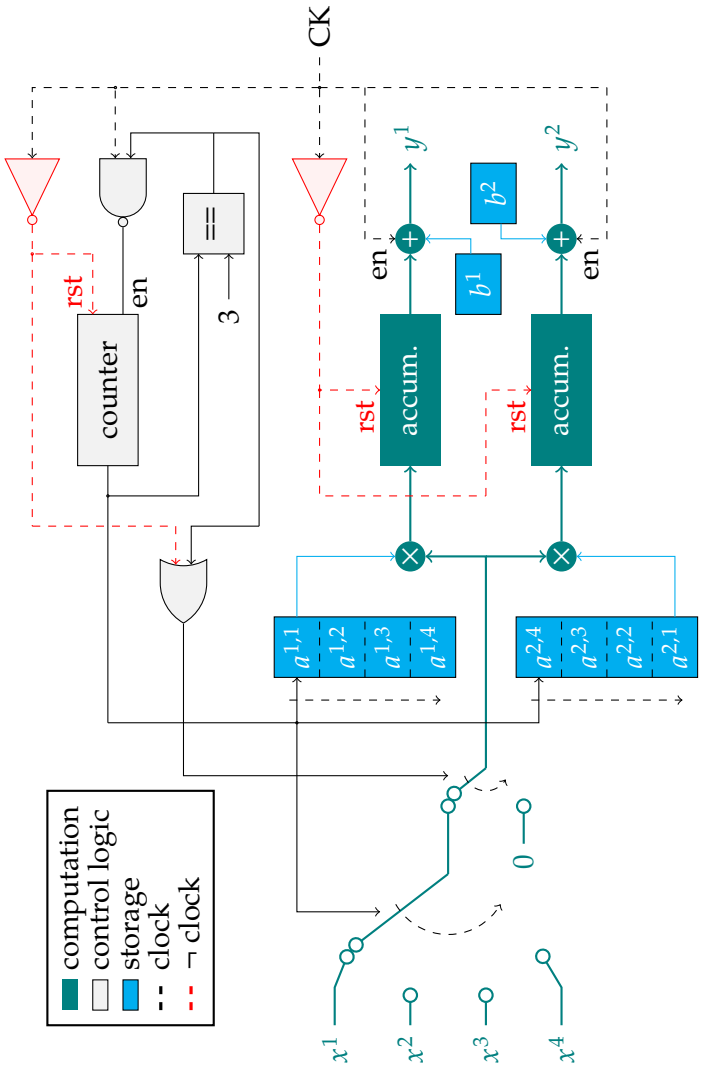

Figure 4.6: Dual Gradient Projection on FPGA. Detail of the Matrix-Vector Multiplication unit.

the proper entry of the b vector.

The rows of the A matrix (*cyan* blocks) are stored in distributed RAM blocks, meaning that can be placed by the compiler anywhere on the chipset. This is a trade-off that minimizes latency at the cost of increased chip occupancy.

The control logic is depicted in black. The key element is a counter that directly pilots the input selector switch. Whenever the counter reaches the input size, a second switch is triggered and the multiplier units start to receive the 0 signal thus stopping the accumulation on the output. Moreover, the output of the `nand` block becomes `FALSE`, disabling the counter itself.

The MVM clock signal is depicted as a black dashed path, and its negate in red. While `TRUE`, it keeps the adders working. Then, as soon as it turns `FALSE`:

1. the counter is reset to 0 and disabled;
2. a switch is triggered so that a 0-signal is fed to the accumulator units;
3. the accumulators reset.

As a result, the MVM black-box behavior works as follows. While the MVM clock is `FALSE`, the unit outputs 0. As soon as a `FALSE`→`TRUE` event is detected, the unit starts reading input variables and computing partial results on the output signals. After $n_x + 5$ master FPGA

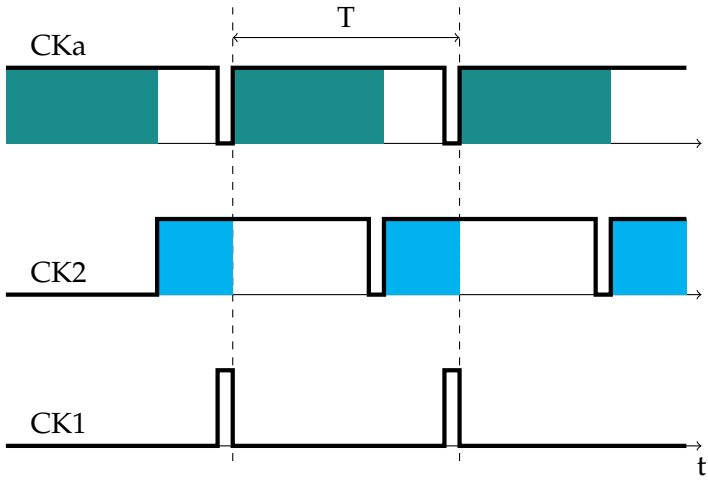

Figure 4.7: Clock signals for the MVM units (CK1, CK2) and for the accumulator unit (CKa). Shaded sectors mean that computations are in progress.

clock ticks the matrix-vector products are ready, and the outputs are kept stable with the final result as long as the MVM clock remains `TRUE`. Notice that the clocks governing the MVM units control logic are not the same of the FPGA master clock.

Figure 4.7 shows the evolution of the two MVM clock signals CK1, CK2 and the accumulator unit clock signal CKa. A single algorithm iteration is completed in a period T of length equal to $(n_z + n_y + 11)$ master FPGA clock cycles, and evolves as follows:

1. a `FALSE`→`TRUE` event is triggered on the first MVM unit, which starts its computations using the y signals of the previous

iteration;

2. after $(n_y + 5)$ master clock cycles (*green* area) the computation is ready, and a `FALSE`→`TRUE` event is triggered on the second MVM unit (in the meanwhile, the first MVM is kept enabled to feed the correct solution to the downstream units);

3. after $(n_z + 5)$ master clock cycles (*green* area) all the matrix-vector computations are executed, and a single `TRUE` clock tick is fed to the accumulator unit, completing the algorithm iteration.

Table 4.3 reports the results of timing and power analysis performed for a *Xilinx Kintex 7-xc7k480t* chipset. The tests are performed starting from randomly-generated QP problems of increasing size. Table columns report: (1) the number of primal and dual variables for the QP, (2) the maximum path latency, (3) the maximum master clock frequency, (4) the time needed to complete a single algorithm iteration, (5) the percentage of occupied slices, and (6) the power consumption.

Due to parallelization and pipelining, the maximum path latency is not affected by the problem size, and the computation time grows only linearly with size, in spite of the quadratic complexity of the matrix-vector operations.

Table 4.3: Fixed-Point Dual Gradient Projection Algorithm on FPGA.

Size [*vars/constr*]	**Latency** [ns]	**Clock** [MHz]	**TPI** [μs]	**Slice Occupation**	**Power** [W]
4/8	5.773 ±0.035	173	0.13	1.81%	0.648
8/16	5.773 ±0.035	173	0.2	3.59%	1.102
16/32	5.773 ±0.035	173	0.34	4.77%	8.571

Chapter 5

Aerospace Applications

Spacecraft attitude control with reaction wheels actuators is detailed in this chapter. The spacecraft nonlinear model is described in Section 5.1, followed by the control objective in Section 5.2. Then, the linearized control model is defined (Section 5.3). The MPC formulation is presented in Section 5.4, and its computational complexity discussed in Section 5.5. Closed-loop simulation results are reported in Section 5.6. Finally, the problems of reaction wheels desaturation with gravity gradients and Earth magnetic field are analyzed in Section 5.7 and Section 5.8, respectively.

5.1 Spacecraft Nonlinear Model

The rotational kinematics and dynamics equations of a spacecraft, considering a principal body frame fixed to its center of mass, are given by

$$\begin{bmatrix} \dot{\phi}(t) \\ \dot{\theta}(t) \\ \dot{\psi}(t) \end{bmatrix} = \frac{1}{c(\theta)} \begin{bmatrix} c(\theta) & s(\phi)s(\theta) & c(\phi)s(\theta) \\ 0 & c(\phi)c(\theta) & -s(\phi)c(\theta) \\ 0 & s(\phi) & c(\phi) \end{bmatrix} \begin{bmatrix} \omega_1(t) \\ \omega_2(t) \\ \omega_3(t) \end{bmatrix} \tag{5.1}$$

and

$$\begin{aligned} J_1\dot{\omega}_1 &= (J_2 - J_3)\omega_2\omega_3 + M_1 \simeq M_1, \\ J_2\dot{\omega}_2 &= (J_3 - J_1)\omega_1\omega_3 + M_2 \simeq M_2, \\ J_3\dot{\omega}_3 &= (J_1 - J_2)\omega_1\omega_2 + M_3 \simeq M_3, \end{aligned} \tag{5.2}$$

where $c(\cdot) \triangleq \cos(\cdot)$ and $s(\cdot) \triangleq \sin(\cdot)$; $\phi(t)$, $\theta(t)$, $\psi(t)$ (rad) are the spacecraft roll, pitch and yaw angles, respectively; for $i = (1, 2, 3)$, $\omega_i(t)$ (rad/s) are the angular velocities, J_i (kgm^2) are the principal moments of inertia, and M_i (Nm) are the spacecraft moments.

We suppose that the spacecraft is equipped with 3 reaction wheels along each of its body frame axes. We consider those wheels as perfect discs with moments of inertia $\tilde{J}_i$, $i = 1, 2, 3$, generating torques about the respective principal axes. The equations linking the spacecraft moments to the reaction wheels dynamics are defined as follows:

$$\begin{aligned} M_1 &= -\tilde{J}_1 (\dot{\omega}_1 + \ddot{\alpha}_1 + \dot{\alpha}_3\omega_2 - \dot{\alpha}_2\omega_3) \simeq -\tilde{J}_1(\dot{\omega}_1 + \ddot{\alpha}_1), \\ M_2 &= -\tilde{J}_2 (\dot{\omega}_2 + \ddot{\alpha}_2 + \dot{\alpha}_1\omega_3 - \dot{\alpha}_3\omega_1) \simeq -\tilde{J}_2(\dot{\omega}_2 + \ddot{\alpha}_2), \\ M_3 &= -\tilde{J}_3 (\dot{\omega}_3 + \ddot{\alpha}_3 + \dot{\alpha}_2\omega_1 - \dot{\alpha}_1\omega_2) \simeq -\tilde{J}_3(\dot{\omega}_3 + \ddot{\alpha}_3), \end{aligned} \tag{5.3}$$

where $\dot{\alpha}_i$ (rad/s) are the wheels rotational speeds, and $\ddot{\alpha}_i(rad/s^2)$ are the wheels accelerations.

Putting together equations (5.1)-(5.3) we can formulate a ninth-order state-space nonlinear model with state

$$x = \begin{bmatrix} \phi & \theta & \psi & \omega_i & \dot{\alpha}_i \end{bmatrix}',$$

for $i = 1, 2, 3$, that evolves according the following set of ODEs (dependencies on time are omitted):

$$\dot{\phi} = \frac{1}{c(\theta)} \left(c(\theta)\omega_1 + s(\phi)s(\theta)\omega_2 + c(\phi)s(\theta)\omega_3 \right) \tag{5.4a}$$

$$\dot{\theta} = \frac{1}{c(\theta)} \left(c(\phi)c(\theta)\omega_2 - s(\phi)s(\theta)\omega_3 \right) \tag{5.4b}$$

$$\dot{\psi} = \frac{1}{c(\theta)} \left(s(\phi)\omega_2 + c(\phi)\omega_3 \right) \tag{5.4c}$$

$$\dot{\omega}_1 = \frac{1}{J_1 + \tilde{J}_1} \left((J_2 - J_3)\omega_2\omega_3 - \tilde{J}_1(\dot{\alpha}_3\omega_2 - \dot{\alpha}_2\omega_3) - u_1 \right) \tag{5.4d}$$

$$\dot{\omega}_2 = \frac{1}{J_2 + \tilde{J}_2} \left((J_3 - J_1)\omega_1\omega_3 - \tilde{J}_2(\dot{\alpha}_1\omega_3 - \dot{\alpha}_3\omega_1) - u_2 \right) \tag{5.4e}$$

$$\dot{\omega}_3 = \frac{1}{J_3 + \tilde{J}_3} \left((J_1 - J_2)\omega_1\omega_2 - \tilde{J}_3(\dot{\alpha}_2\omega_1 - \dot{\alpha}_1\omega_2) - u_3 \right) \tag{5.4f}$$

$$\ddot{\alpha}_i = \frac{1}{\tilde{J}_i} u_i, \ i = 1, 2, 3, \tag{5.4g}$$

where u_1, u_2, u_3 are the torques exerted on the wheels by electric motors.

5.2 Control Objective

The control objective is to track a reference spacecraft orientation

$$\begin{bmatrix} \phi(t) \\ \theta(t) \\ \psi(t) \end{bmatrix} \rightarrow \begin{bmatrix} \phi^r(t) \\ \theta^r(t) \\ \psi^r(t) \end{bmatrix} = r(t) \tag{5.5}$$

subject to polytopic constraints in the form

$$\underline{z} \leq z_c \leq \overline{z},\ z_c = C_c \begin{bmatrix} \phi(t) \\ \theta(t) \\ \psi(t) \\ \dot{\alpha}_i \end{bmatrix} + D_c u(t), \tag{5.6}$$

for $i = 1, 2, 3$. In other words, the control framework must be able to handle constraints on the spacecraft orientation, the reaction wheel speeds, and the control inputs.

To this end, we need to formulate a Model Predictive Control setup that:

- is able to solve problem (5.5)-(5.6) taking into account both spacecraft and reaction wheels dynamics;
- minimizes the computational impact and memory requirements;
- can be deployed on fixed-point microcontrollers or as an embedded MPC-on-a-chip device.

This is achieved by formulating a reduced-order control model, halving the overall problem size while retaining a description of the

significant spacecraft and wheels dynamics, with a modified MPC formulation based on virtual optimization variables and references that guarantees offset-free tracking while lowering the prediction horizon requirements, further reducing the QP problem size.

The solution of the QP problem is assigned to the Fixed-Point Dual Gradient Algorithm described in Chapter 2, tailored for execution on embedded devices, that exploits fixed-point arithmetics to reduce computational load and memory footprint.

5.3 Control Model

The nonlinear model of Section 5.1 is too complex as a prediction model for an embedded MPC controller. We need to formulate a linear, reduced-order model, that however is still capable of capturing the significant dynamics of the system.

Since the angular momentum is conserved and neglecting all but linear terms in (5.2)-(5.3), we set the linear model

$$\dot{\omega}_i = -\frac{\tilde{J}_i}{J_i}\ddot{\alpha}_i, \ i = 1, 2, 3, \tag{5.7}$$

hence

$$\ddot{\alpha}_i = \frac{J_i}{J_i\tilde{J}_i - \tilde{J}_i^2} u_i \triangleq \mathbf{f}\left(J_i, \tilde{J}_i\right) u_i, \ 1 = 1, 2, 3. \tag{5.8}$$

We are now ready to define the reduced-order model for MPC design in LTI state-space form, linearized for small angles, as follows

$$\overbrace{\begin{bmatrix}\dot{\phi}(t)\\ \dot{\theta}(t)\\ \dot{\psi}(t)\\ \ddot{\alpha}_i(t)\\ \ddot{\alpha}_2(t)\\ \ddot{\alpha}_3(t)\end{bmatrix}}^{\dot{x}(t)} = A_C \overbrace{\begin{bmatrix}\phi(t)\\ \theta(t)\\ \psi(t)\\ \dot{\alpha}_1(t)\\ \dot{a}_2(t)\\ \dot{a}_3(t)\end{bmatrix}}^{x(t)} + B_C \overbrace{\begin{bmatrix}u_1(t)\\ u_2(t)\\ u_3(t)\end{bmatrix}}^{u(t)},$$

(5.9)

$$A_C \triangleq \left[\begin{array}{c|ccc} \mathbf{0}^{3\times3} & -\frac{\tilde{J}_1}{J_1} & 0 & 0 \\ & 0 & -\frac{\tilde{J}_2}{J_2} & 0 \\ & 0 & 0 & -\frac{\tilde{J}_3}{J_3} \\ \hline \mathbf{0}^{3\times3} & \multicolumn{3}{c}{\mathbf{0}^{3\times3}} \end{array}\right],$$

$$B_C \triangleq \left[\begin{array}{ccc} \multicolumn{3}{c}{\mathbf{0}^{3\times3}} \\ \hline \mathbf{f}\left(J_1, \tilde{J}_1\right) & 0 & 0 \\ 0 & \mathbf{f}\left(J_2, \tilde{J}_2\right) & 0 \\ 0 & 0 & \mathbf{f}\left(J_3, \tilde{J}_3\right) \end{array}\right].$$

5.4 MPC Formulation

While designing an MPC scheme, the choice of an appropriate prediction horizon is a critical step to ensure proper controller performance. The prediction horizon should ideally cover the time needed to perform the required maneuvers (e.g., on a reference step change, the controller should be able to "see" in the future when the orientation has approached the new reference). Therefore, one should know *a priori* bounds on the reference variations in order to select a prediction horizon large enough to account for them.

Moreover, we highlight two more issues:

1. the spacecraft maneuvers are usually "slow" compared with the controller sampling rates, leading to requirements for large prediction horizon and consequently a large size of the Quadratic Program associated with the MPC controller;

2. a prediction horizon tailored to account for the upper bound on the reference variations may cause the smaller maneuvers to be excessively slow (in case longer horizons lead to less aggressive control actions).

To address these issues, we propose a modified MPC setup with additional virtual optimization variables and references. The approach is similar to the one in [144], and can be described as follows.

We define a new optimization vector, where the variations on the control input (a standard choice for MPC tracking problems) are extended with a new vector of length equal to the number of system states, becoming $\begin{bmatrix}\Delta u_k' & \tilde{x}_k'\end{bmatrix}'$, where $\Delta u_k \in \mathbb{R}^3$ and $\tilde{x}_k \in \mathbb{R}^6$, $k = 0, \ldots, N-1$, and set the new cost function

$$\begin{aligned} V(x, \tilde{r}, \Delta u, \tilde{x}) = &\|x_N - \tilde{r}\|^2_{W_N} + \\ &\sum_{k=0}^{N-1} \|(x_k - \tilde{x}_k) - \tilde{r}\|^2_{W_r} + \|\tilde{x}_k\|^2_{W_x} + \|\Delta u_k\|^2_{W_u}, \end{aligned} \tag{5.10}$$

where N is the prediction horizon, $\tilde{r}$ is the reference signal for spacecraft attitude and wheels speeds, $\tilde{x} = \begin{bmatrix}\tilde{x}_0' & \ldots & \tilde{x}_{N-1}'\end{bmatrix}'$, $\Delta u = \begin{bmatrix}\Delta u_0' & \ldots & \Delta u_{N-1}'\end{bmatrix}'$.

The optimal control problem to be solved at each sampling step becomes

$$\begin{aligned} V^*(x_t, \tilde{r}_t, u_t) &= \min_{\Delta u, \tilde{x}} V(x_t, \tilde{r}_t, \Delta u, \tilde{x}) \\ \text{subject to} \quad x_{k+1} &= Ax_k + Bu_k, \ k = 0, \ldots, N-1 \\ u_k &= u_{k-1} + \Delta u_k, \ k = 0, \ldots, N_c - 1 \\ u_k &= u_{k-1}, \ k = N_c, \ldots, N-1 \\ u_{-1} &= u_t, \ x_0 = x_t \\ (x_k, u_k) &\in \mathcal{Z}, \ k = 0, \ldots, N-1, \end{aligned} \tag{5.11}$$

where N_c is the control horizon and $\mathcal{Z}$ is the polytope defined in (5.6).

Looking at the cost function (5.10), notice that there is no direct

penalty for the state being far from the reference, as in standard tracking MPC. Instead, the optimizer has the freedom to trade-off between the discrepancy state/reference and the magnitude of the auxiliary variables $\tilde{x}$, for each prediction step. This trade-off is also influenced by properly tuning the weight matrices W_r, W_x, W_N.

The practical impact on the control performance when adopting the modified MPC setup (5.10)-(5.11) is that the prediction horizon can be chosen independently of the duration of the spacecraft maneuvers, and that the simulated closed-loop system shows stable behavior for significantly smaller prediction horizons. Similar effects have been observed in the orbital maneuvering study [145].

Figure 5.1 shows the domain of initial conditions for which the controller is able to drive the state trajectories to a target set (*black*) of radius 0.05 rad, for a spacecraft simulated with the full nonlinear model (5.4). The domains of attraction are computed by performing a grid search on the state space and running closed loop simulations with two controllers.

The *green* set is obtained when the controller solves the modified problem proposed in (5.11), the red set when is instead solved a standard MPC tracking problem (i.e., removing the auxiliary optimization variables $\tilde{x}$, see also Section 1.1.1), while leaving all the other parameters unchanged.

The control models are discretized with sampling time $Ts = 0.5s$. The full parameter setup adopted for the closed-loop simulation is reported in Table 5.1.

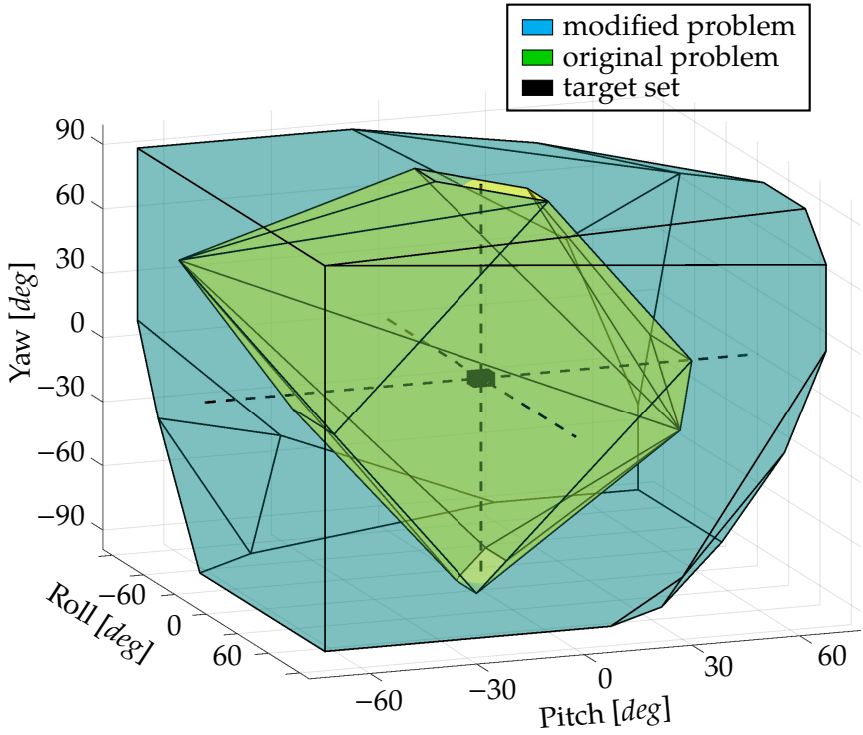

Figure 5.1: Domain of attraction representing initial conditions from where an MPC controller based on the modified problem (5.10)-(5.11) (*green* set) and a standard MPC controller (*green* set) are able to asymptotically converge to the target set (*black*).

Table 5.1: Parameters in the spacecraft closed-loop simulations.

Parameter	**Value**
Moments of inertia	
J_1	$3000\,kgm^2$
J_2	$1500\,kgm^2$
J_3	$2000\,kgm^2$
$\tilde{J}_i$	$50\,kgm^2$
Controller	
Sampling time	$0.5s$
Prediction horizon	10
Control horizon	2
Constraints	
Control moments	$[-1,+1]\,Nm$
Wheel speeds	$[-10,10]\,rad/s$
Cost function weights	
W_r	$\left[\begin{array}{c\|c} 100\cdot\mathbf{I}^3 & \mathbf{0}^{3\times3} \\ \hline \mathbf{0}^{3\times3} & 0.1\cdot\mathbf{I}^3 \end{array}\right]$
W_x	$50\cdot\mathbf{I}^6$
W_u	$0.01\cdot\mathbf{I}^3$
W_N	LQR

Simulation results show how, even with a prediction horizon $N = 10$ (that gives the controller a prediction window of 5 seconds only), we obtain a domain of attraction spanning up to $[-\pi/2, \pi/2]$ radians for the roll and yaw angles, and up to $[-\pi/3, \pi/3]$ radians for the pitch angle. This is a remarkable result, considering that the controller operates with a prediction model linearized for small angles.

The control development is not complete yet, as it does not guarantee offset-free tracking in presence of model uncertainties, and is not able to reject any constant external disturbance, such as caused by gravity gradients or solar radiation pressure.

There are several ways to achieve integral action in a controller. For a brief review of the principal ones, refer to Section 1.5. For this application we adopted the reference-governor like approach, where the reference fed to the controller becomes $\tilde{r}(t) = \begin{bmatrix} r(t) & \mathbf{0}^{3\times 1} \end{bmatrix}'$, where

$$r(t) = \begin{bmatrix} \phi_{des} \\ \theta_{des} \\ \psi_{des} \end{bmatrix} - \int_0^t \left(\begin{bmatrix} \phi(\tau) \\ \theta(\tau) \\ \psi(\tau) \end{bmatrix} - \begin{bmatrix} \phi_{des} \\ \theta_{des} \\ \psi_{des} \end{bmatrix} \right) \mathrm{d}\tau. \tag{5.12}$$

With this approach we achieve offset-free tracking. The computation of the integral action (5.12) can be easily performed by an external integrator without affecting the complexity of the controller.

The closed-loop simulation depicted in Figure 5.2 highlights the effects of the integral action. The controller is set to track a reference step change. Simulation parameters are reported in

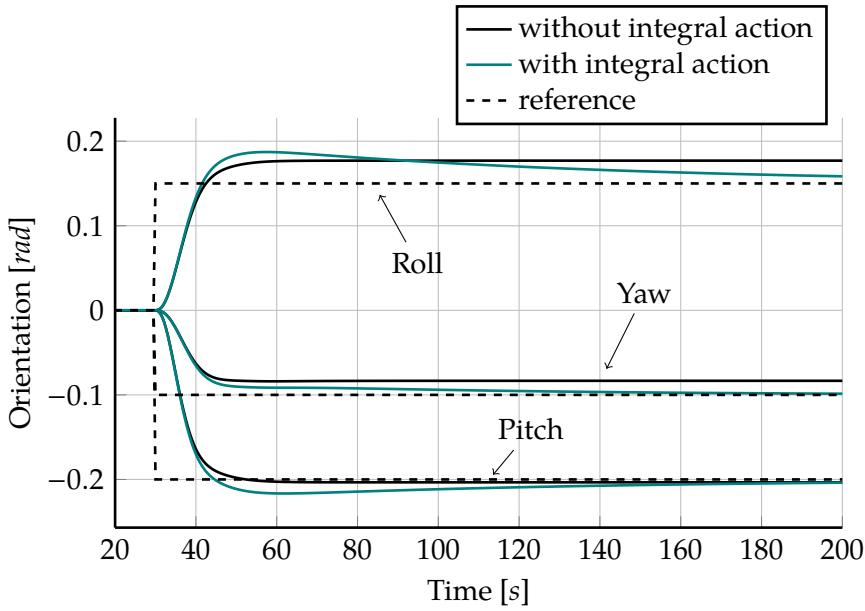

Figure 5.2: Offset-free control. Comparison between a controller equipped with the integral action on the reference (*green* lines), and a controller without integral action (*black* lines), when tracking reference roll, pitch and yaw signals (*dashed* lines).

Table 5.1. Without integral action (*black* lines) we observe a steady-state tracking error on the roll and yaw angles, that is caused by the discrepancy between the simplified, linearized model used in the prediction and the full nonlinear model used to simulate the spacecraft response. On the other hand, by adding the integral action (*green* lines) the error is eventually compensated, and steady-state offset-free tracking is achieved.

5.5 Computational Complexity

We now analyze the computational complexity of the controller built on the MPC formulation introduced in Section 5.4, with the QP solver based on the Fixed-Point Dual Gradient Projection detailed in Chapter 2. The fixed-point number representation is chosen with 32-bit word length, of which 16 bits for the fractional part, 15 bits for the integer part, and one sign bit.

The QP algorithm has been implemented in library-free ANSI-C for a streamlined deployment on hardware platforms. The resulting code is composed by two functions only; an `init` function to be called once which initializes the problem data, and a `step` function to be called at each sampling step which accepts spacecraft orientation measurements and references as input, and outputs the control signals for the reaction wheels torques.

Table 5.2 shows the complexity for three different controller configurations: the first with input constraints only (upper and lower bounds on all the control signals); the second adding constraint on the input variations; the last including also constraints on both spacecraft orientation (inclusion zone constraints) and wheels speed.

The *Variables* column reports the number of primal and dual variables of the resulting QP problem. The *Size* column shows the memory requirements to store the problem data and to execute the code functions. Finally, the *Oper./Iter* column reports the number of fixed-point operations (multiplications and additions) required to complete an algorithm iteration.

Table 5.2: Spacecraft attitude controller complexity.

Constraints	**Variables**		**Size** [KB]		**Oper./Iter**
	Primal	*Dual*	*Data*	*Code*	
u	18	12	8.7	4.2	460
$u + \Delta u$	18	24	8.7	4.5	900
$u + \Delta u + x$	18	144	37.5	7.7	5300

Results show how, thanks to the modified MPC formulation and the choice of the fixed-point QP solver, we are able to maintain the QP problem small and solve it efficiently with minimal memory footprint and computational burden. Moreover, given specific hardware it is possible to estimate precisely the time required to perform a single iteration (since the algorithm performs linear-algebra computations on matrices of pre-determined sizes). Then, given the sampling time, one can determine the number of iterations that can be computed within the sampling period. Finally, using the results in [78] one can formulate the MPC problem with stability and recursive feasibility guarantees.

5.6 Simulations

The following simulations are aimed to investigate the closed-loop behavior when a controller based on the proposed MPC setup is connected to a spacecraft simulated with the nonlinear model of Section 5.1.

5.6.1 Sinusoidal References Tracking

In this simulation the controller is required to track sinusoidal references, with varying amplitudes and frequencies, for the attitude of a spacecraft simulated according to the nonlinear model of Section 5.1. The MPC parameters are the reported in Table 5.1.

The maneuver of tracking sinusoidal references, despite not being of particular interest in a real-world scenario, becomes useful to test whether the controller is properly designed, with a reduced-order prediction model which is able to capture the plant significant dynamics.

Figure 5.3 shows the result of the closed-loop simulation with trajectories of the control inputs and wheels rotational speeds (*top* plot), and of the spacecraft orientation compared to reference trajectories (*bottom* plot).

The closed-loop behavior is consistent with the references, despite the fact that the controller relies on a reduced system model for predictions. It has to be noted that, in the current MPC formulation, the controller is not aware of future references. If such information is available, it can be incorporated into the prediction model further

improving the controller performance.

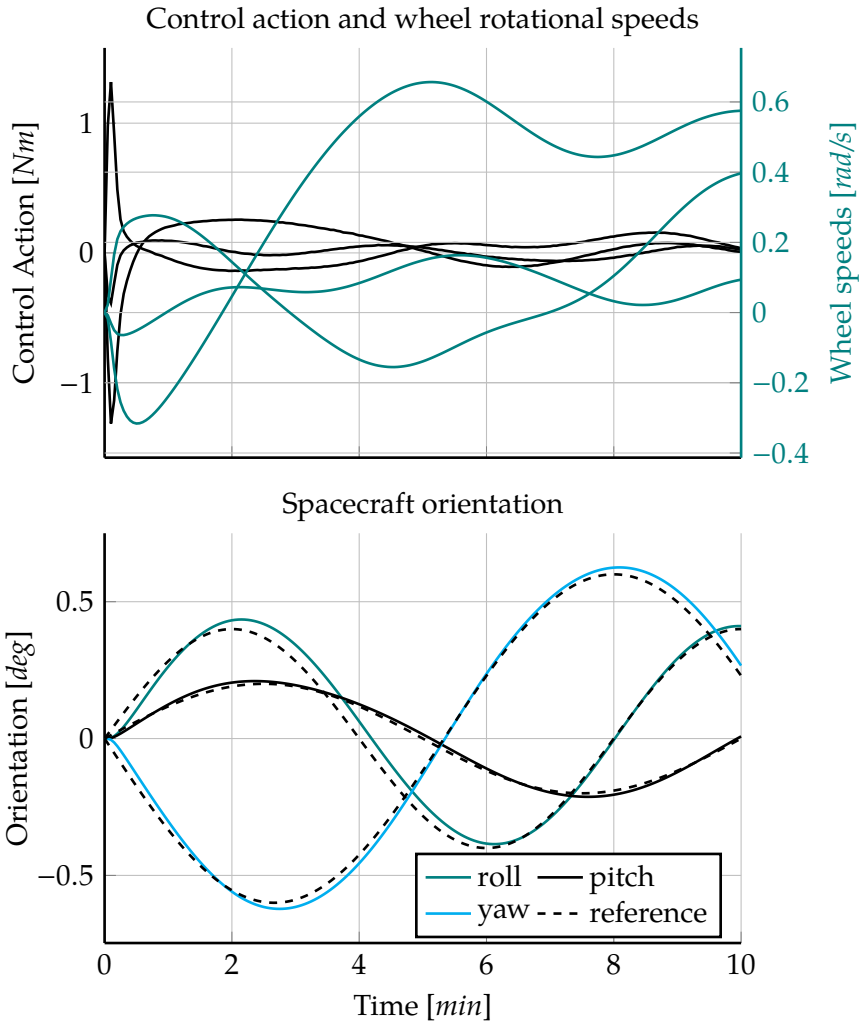

Figure 5.3: Closed-loop simulation of sinusoidal reference tracking. *Top* plot: trajectories of control inputs and reaction wheel speeds. *Bottom* plot: spacecraft orientation and reference signals.

5.6.2 Rest-to-Rest Orientation Maneuver

The purpose of this simulation is to show the closed-loop behavior when performing a rest-to-rest orientation, a common maneuver in real-world scenarios. Simulations are repeated for varying actuator constraints.

The controller is set to track a reference step change, where

$$\begin{bmatrix} \phi^r(t) \\ \theta^r(t) \\ \psi^r(t) \end{bmatrix} : \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0.08 \\ -0.03 \\ -0.1 \end{bmatrix} [rad].$$

All the parameters are as in Table 5.1, with the only difference that the maneuver is repeated for loose input constraints, $u(t) \in [-3, +3]Nm$, and for tight input constraints, $u(t) \in [-0.2, +0.2]Nm$.

Simulation results are depicted in Figure 5.4. The controller is able to react to the different input constraints and complete the rest-to-rest orientation maneuver, even in the case of strongly constrained actuators.

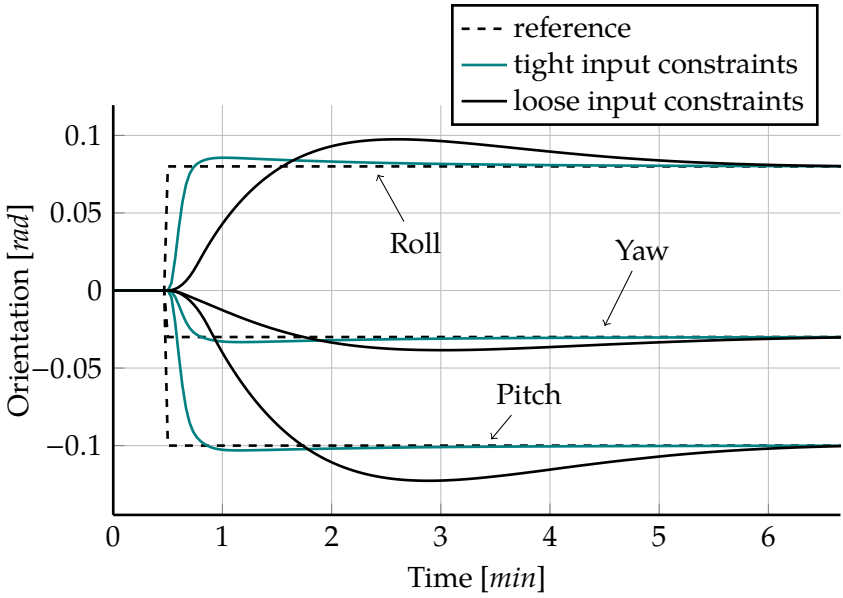

Figure 5.4: Rest-to-rest orientation maneuver with tight input constraints (*green* curves) and loose input constraints (*black* curves).

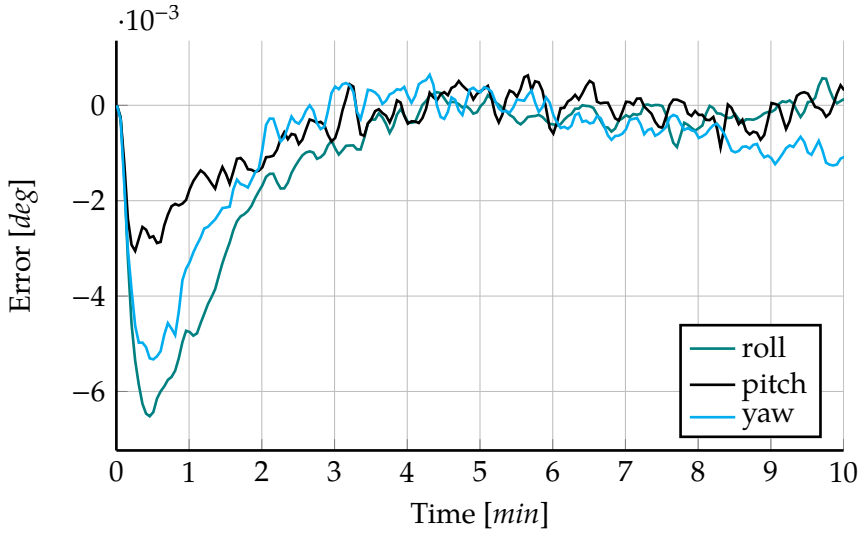

Figure 5.5: Fixed-point and 64-bit floating-point comparison. Discrepancy in spacecraft attitude trajectories during the rest-to-rest maneuver when using different controllers based on the two arithmetics.

5.6.3 Fixed-Point Accuracy

The goal of this simulation is to evaluate the robustness of the proposed QP solver with respect to finite-precision number representations.

We ran the closed-loop simulation of Section 5.6.2 (rest-to-rest orientation maneuver) first with 32-bit fixed-point arithmetic, of which 16 bits for the fractional part, then switching to 64-bit, double precision floating point arithmetic. The plots in Figure 5.5 show the discrepancy in the resulting spacecraft orientation.

The orientation discrepancy due to fixed-point computations reaches its peak at the beginning of the maneuver where actuator effort is larger (possibly saturated) and therefore the algorithm generally requires more iterations to compute the solution. Nevertheless, the magnitude of the discrepancy remains small, in the order of 10^{-3} degrees for the whole maneuver.

5.7 Reaction Wheels Desaturation by Gravity Gradients

5.7.1 Background

The *gravity gradients* are torques caused by the Earth gravitational field effects on the spacecraft body; since the gravity force decreases proportionally to the square of the distance, the spacecraft sections closer to the Earth receive a slightly larger pull with respect to sections farther away.

By including the gravity gradients effects into the spacecraft and reaction wheel kinematic and dynamic equations, one is able to derive a *completely controllable* state-space model; this means that is possible, in principle, to steer both the spacecraft attitude and the reaction wheel speeds to specified set-points.

Thanks to the adoption of an MPC-based controller, desaturation of the wheels is achieved by optimally exploiting the torques induced by the gravity gradients, while maintaining the spacecraft attitude constrained in a specified set.

5.7.2 Nonlinear Model

The spacecraft rotational kinematics equations that include gravity gradient effects, assuming that the body fixed frame is the principal

frame with the origin at the center of mass, become

$$\begin{bmatrix}\dot{\phi}(t)\\ \dot{\theta}(t)\\ \dot{\psi}(t)\end{bmatrix} = \frac{1}{c(\theta)}\begin{bmatrix}c(\theta) & s(\phi)s(\theta) & c(\phi)s(\theta)\\ 0 & c(\phi)c(\theta) & -s(\phi)c(\theta)\\ 0 & s(\phi) & c(\phi)\end{bmatrix}\begin{bmatrix}\omega_1(t)\\ \omega_2(t)\\ \omega_3(t)\end{bmatrix} + \\ +n\begin{bmatrix}c(\theta)s(\psi) + c^{-1}(\theta)\left(\sigma_2 s(\phi)s(\theta) - \sigma_1 c(\phi)s(\theta)\right)\\ \sigma_2 c(\phi) + \sigma_1 s(\phi)\\ c^{-1}(\theta)\left(\sigma_2 s(\phi) - \sigma_1 c(\phi)\right)\end{bmatrix}, \tag{5.13}$$

where

$$\sigma_1 \triangleq c(\psi)s(\phi) - c(\phi)s(\psi)s(\theta),$$
$$\sigma_2 \triangleq c(\phi)c(\psi) + s(\phi)s(\psi)s(\theta),$$
$$n \triangleq \sqrt{\frac{\mu}{R_0^3}},$$

and μ is the gravitational constant, R_0 is the nominal orbital radius.

The spacecraft rotational dynamics equations become

$$\begin{aligned} J_1\dot{\omega}_1 &= (J_2 - J_3)\left(\omega_2\omega_3 - 3n^2 s(\phi)c(\phi)c^2(\theta)\right) - \tilde{J}_1\left(\ddot{\alpha}_1 + \dot{\omega}_1\right),\\ J_2\dot{\omega}_2 &= (J_3 - J_1)\left(\omega_1\omega_3 + 3n^2 c(\phi)c(\theta)s(\theta)\right) - \tilde{J}_2\left(\ddot{\alpha}_2 + \dot{\omega}_2\right),\\ J_3\dot{\omega}_3 &= (J_1 - J_2)\left(\omega_1\omega_2 + 3n^2 s(\phi)s(\theta)c(\theta)\right) - \tilde{J}_3\left(\ddot{\alpha}_3 + \dot{\omega}_3\right). \end{aligned} \tag{5.14}$$

Finally, the reaction wheels rotational dynamics equations are

$$\begin{aligned} \ddot{\alpha}_1 &= n\dot{\alpha}_3 + u_1,\\ \ddot{\alpha}_2 &= u_2,\\ \ddot{\alpha}_3 &= -n\dot{\alpha}_1 + u_3, \end{aligned} \tag{5.15}$$

where u_i, $i = (1, 2, 3)$, are the rotational accelerations induced on the wheels by the electric motors.

5.7.3 Control Model

Due to its nonlinear nature, the model described in Section 5.7.2 is not a good prediction model for an embedded MPC implementation. We therefore obtain a simpler state-space control model by linearizing (5.13)-(5.14) around the nominal conditions

$$\bar{\phi} = 0,\ \bar{\theta} = 0,\ \bar{\psi} = 0,\ \bar{\omega}_1 = 0,\ \bar{\omega}_2 = -n,\ \bar{\omega}_3 = 0, \tag{5.16}$$

The resulting linearized model model is

$$\dot{x}(t) = A_C x(t) + B_C \begin{bmatrix} u_1(t) & u_2(t) & u_3(t) \end{bmatrix}',$$

$$x = \begin{bmatrix} \phi(t) & \theta(t) & \psi(t) & \omega_i(t) & \dot{\alpha}_i(t) \end{bmatrix}',\ i = (1,2,3),$$

$$A_C \triangleq \left[\begin{array}{ccc:ccc:ccc} 0 & 0 & n & 1 & 0 & 0 & \multicolumn{3}{c}{\multirow{5}{*}{$\mathbf{0}^{6\times 3}$}} \\ 0 & 0 & 0 & 0 & 1 & 0 & & & \\ -n & 0 & 0 & 0 & 0 & 1 & & & \\ c_1 & 0 & 0 & 0 & 0 & c_3 & & & \\ 0 & c_2 & 0 & 0 & 0 & 0 & & & \\ 0 & 0 & 0 & c_4 & 0 & 0 & & & \\ \hdashline \multicolumn{3}{c:}{\multirow{2}{*}{$\mathbf{0}^{3\times 3}$}} & \multicolumn{3}{c:}{\multirow{2}{*}{$\mathbf{0}^{3\times 3}$}} & 0 & 0 & n \\ & & & & & & 0 & 0 & 0 \\ & & & & & & -n & 0 & 0 \end{array}\right], \tag{5.17}$$

$$B_C \triangleq \left[\begin{array}{ccc} \multicolumn{3}{c}{\mathbf{0}^{3\times 3}} \\ \hdashline -\sigma_1 & 0 & 0 \\ 0 & -\sigma_2 & 0 \\ 0 & 0 & -\sigma_3 \\ \hdashline \multicolumn{3}{c}{\mathbf{I}^3} \end{array}\right],$$

Table 5.3: Spacecraft (J_i) and wheels ($\tilde{J}$) moments of inertia used in the reaction wheels desaturation simulations.

J_1	J_2	J_3	$\tilde{J}$
1400	1700	1000	50

where

$$
\begin{aligned}
\sigma_i &\triangleq \tilde{J}_i \left(J_i + \tilde{J}_i\right)^{-1}, \\
c_1 &\triangleq -3\sigma_1 n^2 (J_2 - J_3), \\
c_2 &\triangleq 3\sigma_2 n^2 (J_3 - J_1), \\
c_3 &\triangleq -\sigma_1 n (J_2 - J_3), \\
c_4 &\triangleq \sigma_3 n (J_1 - J_2).
\end{aligned}
\tag{5.18}
$$

The controllability matrix of the pair (A_C, B_C) is full rank; this property does not hold if the gravity gradient effects are neglected.

5.7.4 Simulation Results

The wheel desaturation process exploiting gravity gradients is shown in Figure 5.6. The simulation covers a period of 22 hours. The controller is set to regulate spacecraft attitude and reaction wheel speeds, operating at a sampling time of 0.5 s and using a discretized version of (5.17) as prediction model over a 10 steps horizon. The control horizon is 2 steps. The resulting QP has 6 decision variables and 72 constraints.

The system in initialized with the spacecraft body frame aligned with the *LVLH* (Local Vertical/Local Horizontal) frame, and all the

3 reaction wheels spinning at 10 rad/s. Roll, pitch and yaw angles are constrained in the set $[-0.5,\ 0.5]$ rad. The values of the chosen spacecraft and wheels moments of inertia are given in Table 5.3.

Simulation results show how the controller drives the spacecraft to perform oscillations along the roll and yaw angles, while maintaining an offset in the pitch angle. As a result, the wheels are desaturated while maintaining the spacecraft attitude within the specified constraints. The whole process takes about 8 hours to halve wheel speeds, and 16 hours to bring them close to zero rad/s. We note that similar results are obtained for different initial speeds, and it is not necessary to bring the wheel speed all the way to zero during practical desaturation maneuvers.

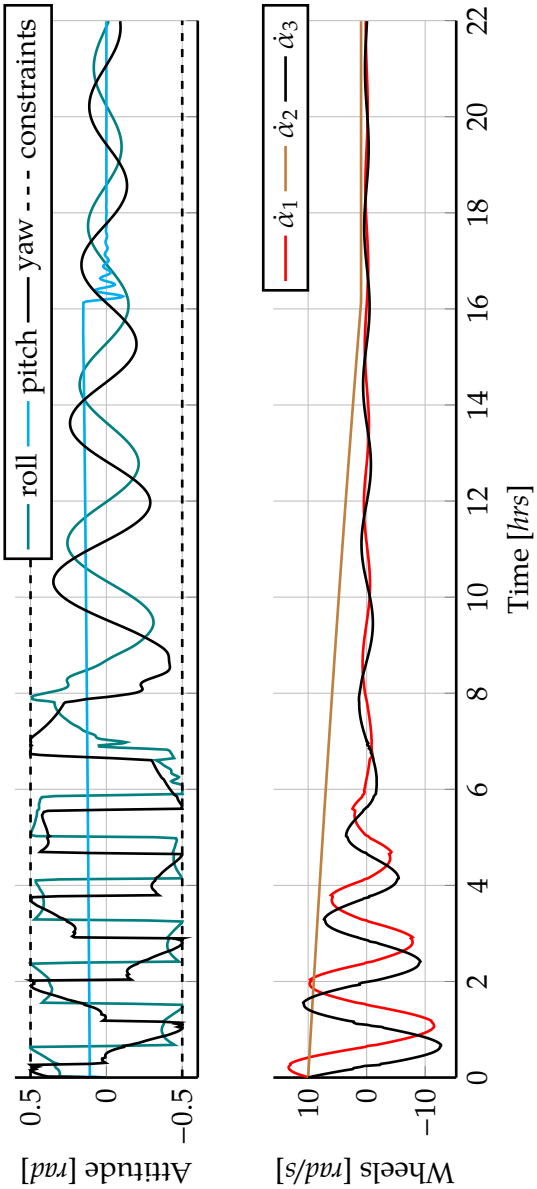

Figure 5.6: Closed-loop simulation with MPC for reaction wheels desaturation using the gravity gradients, starting from 10 rad/s. Top: spacecraft roll, pitch and yaw angles. Bottom: reaction wheels speed.

5.7.5 Comparison with LQR

It is interesting to compare the behavior and performance of MPC with respect to a standard *LQR* controller. The latter does not allow imposing constraints on the spacecraft attitude through the desaturation process. Instead, one is forced to tune properly weights in the cost function for the spacecraft orientation and wheel speeds. A fast desaturation is obtained by increasing the wheel weights; however, this subjects the spacecraft to large oscillations (see dashed lines in Figure 5.7). This behavior is not desirable, as the controller is based on a model linearized for small angles (and the model mismatch may become intolerable). The other option is to increase the weights on spacecraft attitude (see solid lines in Figure 5.7): now its oscillations are smaller, but the wheels desaturation performance is significantly degraded.

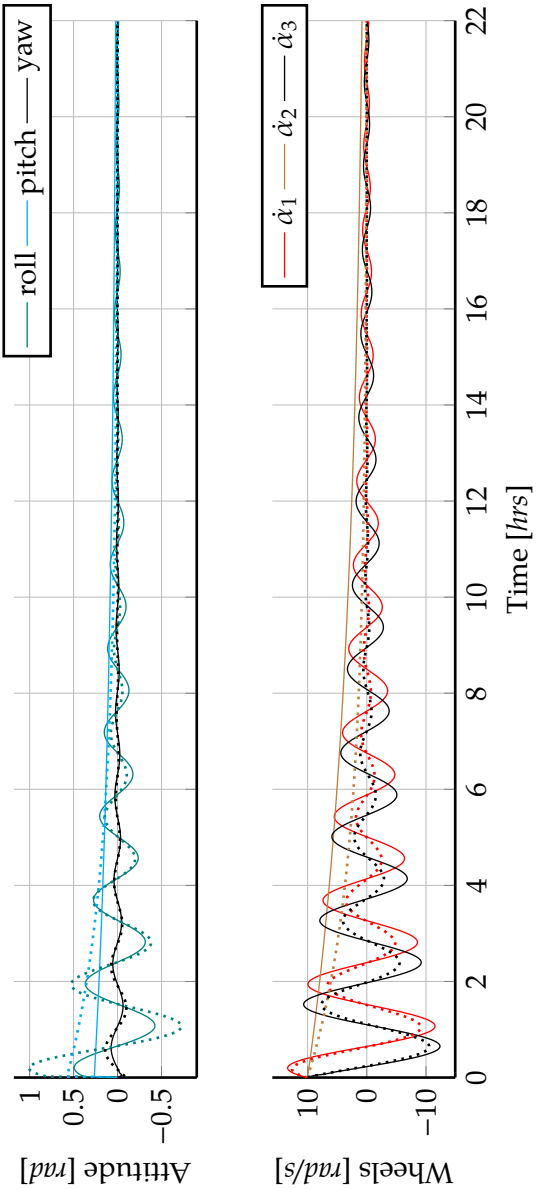

Figure 5.7: Closed-loop simulation with LQR for reaction wheels desaturation using the gravity gradients. Two controllers, one tuned for fast desaturation (*dotted* lines), one for slow desaturation (*solid* lines). *Top*: spacecraft roll, pitch and yaw angles. *Bottom*: reaction wheels speed.

5.8 Reaction Wheels Desaturation by Magnetic Moments

5.8.1 Background

The desaturation by magnetic moments is possible when the spacecraft is equipped with magnetic actuators (usually three, aligned with the body frame axes). These devices are composed by a magnetic core and a coil; when current flows through the latter, a magnetic dipole is generated which interacts with the Earth magnetic field, resulting in a control torque on the spacecraft.

The magnetic moments are stronger than the moments due to gravity gradients, allowing for faster wheel desaturation. However, they require additional equipment on the spacecraft and they are based on the interaction with the Earth magnetic field, which varies along the orbit. Moreover, the spacecraft/wheels system is not completely controllable, since it is not possible to generate magnetic torques along the Earth magnetic field direction. However, since this direction varies as the spacecraft moves along its orbit, an MPC controller can still achieve wheel desaturation.

5.8.2 Nonlinear Model

The spacecraft rotational kinematics and the wheel dynamics are as in (5.13) and (5.15) with $n = 0$.

The spacecraft dynamics equations are

$$\begin{aligned} J_1\dot{\omega}_1 &= (J_2 - J_3)\,\omega_2\omega_3 - \tilde{J}_1\,(\ddot{\alpha}_1 + \dot{\omega}_1) + M_1^B, \\ J_2\dot{\omega}_2 &= (J_3 - J_1)\,\omega_1\omega_3 - \tilde{J}_2\,(\ddot{\alpha}_2 + \dot{\omega}_2) + M_2^B, \\ J_3\dot{\omega}_3 &= (J_1 - J_2)\,\omega_1\omega_2 - \tilde{J}_3\,(\ddot{\alpha}_3 + \dot{\omega}_3) + M_3^B, \end{aligned} \tag{5.19}$$

where M_i^B denote the torques generated by the magnetic actuators. Those can be computed as

$$M \triangleq \begin{bmatrix} M_1^B & M_2^B & M_3^B \end{bmatrix}' = m \times B, \tag{5.20}$$

$$m \triangleq \begin{bmatrix} N_x A_x i_x \\ N_y A_y i_y \\ N_z A_z i_z \end{bmatrix} = \begin{bmatrix} u_1^B \\ u_2^B \\ u_3^B \end{bmatrix}, \tag{5.21}$$

where B is the Earth magnetic field vector, expressed in the spacecraft body fixed frame; $N_{(\cdot)}$, $A_{(\cdot)}$, $i_{(\cdot)}$ are, respectively, the number of coil turns, their areas, and the currents flowing through them, for each of the three magnetic actuators mounted along the x, y and z axes of the spacecraft body frame.

5.8.3 Control Model

The control model is obtained by linearizing the nonlinear model of Section 5.8.2 around the origin. The state vector is the same as in (5.17), while the input vector takes the form

$$u = \begin{bmatrix} u_1 & u_2 & u_3 & u_1^B & u_2^B & u_3^B \end{bmatrix}'. \tag{5.22}$$

The resulting model is

$$\begin{aligned}
\dot{x}(t) &= A_C x(t) + B_C(t) u(t), \\
A_C &\triangleq \left[\begin{array}{c:c:c} \mathbf{0}^{3\times3} & \mathbf{I}^3 & \mathbf{0}^{3\times3} \\ \hdashline \mathbf{0}^{6\times3} & \mathbf{0}^{3\times3} & \mathbf{0}^{3\times3} \end{array}\right], \\
B_C(t) &\triangleq \left[\begin{array}{ccc:ccc} & \mathbf{0}^{3\times3} & & & \mathbf{0}^{3\times3} & \\ \hdashline -\sigma_1 & 0 & 0 & 0 & -\eta_1 & \eta_2 \\ 0 & -\sigma_2 & 0 & \eta_1 & 0 & -\eta_3 \\ 0 & 0 & -\sigma_3 & -\eta_2 & \eta_3 & 0 \\ \hline & \mathbf{I}^3 & & & \mathbf{0}^{3\times3} & \end{array}\right], \\
\sigma_i &\triangleq \tilde{J}_i \left(J_i + \tilde{J}_i\right)^{-1}, \\
\eta_1 &\triangleq B^{long}\theta - B^{lat}\phi - B^v, \\
\eta_2 &\triangleq B^{lat}\psi - B^v\theta - B^{long}, \\
\eta_3 &\triangleq B^v\phi - B^{long}\psi - B^{lat},
\end{aligned} \tag{5.23}$$

where B^{long}, B^{lat}, B^v are, respectively, the longitudinal, latitudinal and vertical components of the Earth magnetic field.

Model (5.23) is linear time-varying (LTV); therefore, an additional computational effort is required to form the quadratic programming problem at each sampling step.

Table 5.4: Test orbit parameters.

Parameter	Value
Type	Low-Earth Orbit
Altitude (min.)	419 km
Altitude (max.)	427 km
Eccentricity	5.66×10^{-4}
Inclination	67 deg
Period	1.55 hrs

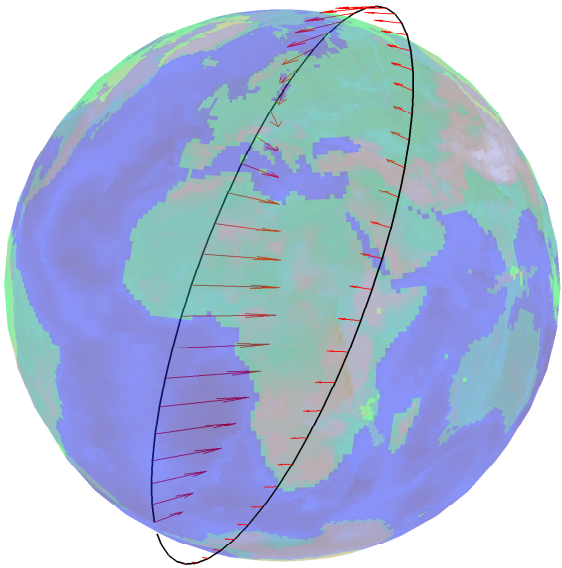

Figure 5.8: Test orbit (*black* line) and Earth magnetic vector field (*red* arrows) used for desaturation by magnetic model.

5.8.4 Simulation Results

Figure 5.8 shows the test orbit used to simulate magnetic desaturation. Orbit parameters are detailed in Table 5.4. The QP solved to compute the control action has 12 decision variables and 84 constraints. The system is initialized with reaction wheels spinning at 100 rad/s. The goal is to lower their speed below 30 rad/s.

The Earth magnetic field is generated with data from the World Magnetic Model [146].

Figure 5.9 shows the results of the simulation obtained with MPC based on the LTV model (5.23). Within 2.5 hours (less than two orbit revolutions) the desaturation process is completed, with all the reaction wheel speeds below the target of 30 rad/s. Then, in 1.5 additional hours, the spacecraft attitude is driven to the rest position, with the controller waiting for favorable Earth magnetic field directions to steer the roll, pitch and yaw angles. Note that, for the whole process, the attitude has been constrained within a small box of side 0.1 rad.

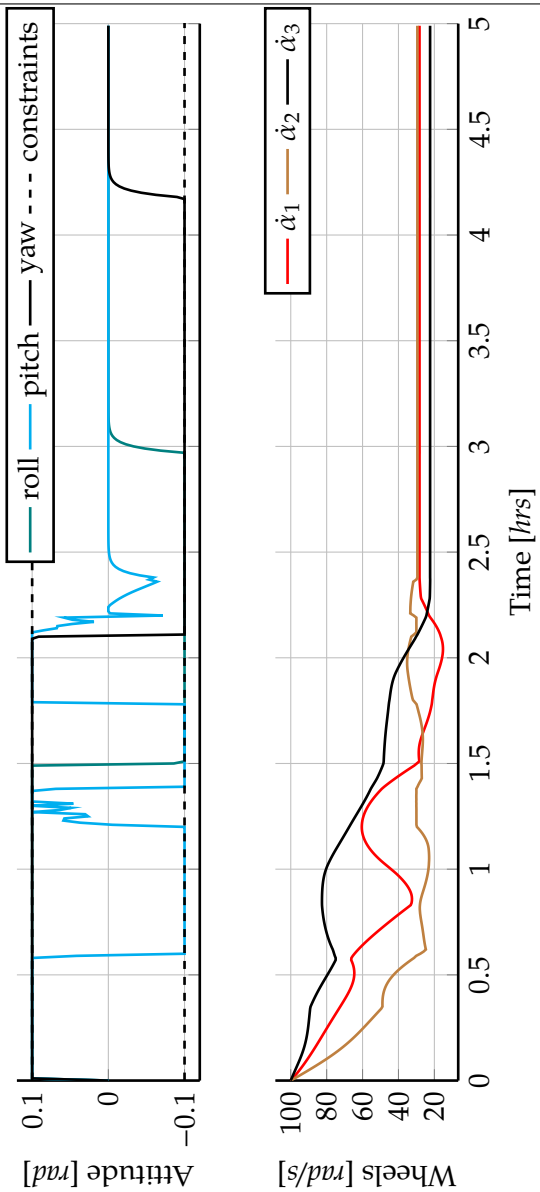

Figure 5.9: Closed-loop simulation with LTV-MPC for reaction wheels desaturation using the Earth magnetic field. The goal is to lower wheel speed below 30 rad/s, starting from 100 rad/s. *Top*: spacecraft roll, pitch and yaw angles. *Bottom*: reaction wheels speed.

Chapter 6

Conclusions

This thesis addressed the problem of extending the feasibility of Model Predictive Control approaches to applications where fast system dynamics is paired with scarce computational resources, such as in automotive and aerospace industries.

The thesis contributions covered both the theoretical aspects, with analysis of algorithms specifically tailored for embedded optimization, as well as the implementation aspects, with experimental tests on hardware platforms and a case study on spacecraft control. Their impact is summarized in the following section.

6.1 Summary

In Chapter 2 a Dual Gradient Projection algorithm tailored for implementations in finite precision arithmetic was proposed. Despite being a dual method, detailed convergence rates for the primal cost and feasibility that take into account round-off errors coming from fixed-point arithmetic were provided. Moreover, concrete and theoretically-proven guidelines for selecting the minimum number of fractional and integer bits that guarantee favorable convergence properties were given. Theoretical results were then validated in simulations.

In Chapter 3 an implementation with fixed-point arithmetic of a Newton-based QP solver for embedded MPC applications was investigated. The propagation of the round-off error coming from the quantization of the number representation space was analyzed, and design guidelines to avoid overflow errors were derived. For ill-conditioned problems, an optimal scaling method to hinder them was proposed. Finally, simulation were conducted to assess algorithm performance.

In Chapter 4 experimental tests on hardware platform were shown. Both the Dual Gradient Projection method and the Proximal Newton method presented in the previous chapters were implemented on low-power, low-cost embedded platforms based on the ARM Cortex-M3 general purpose processing unit. Test results showed that despite the scarce computational capabilities, QP solution times could be computed in the milliseconds range,

with a significant positive impact in performance coming from the adoption of fixed-point arithmetic. Moreover, an FPGA implementation of the DGP method was detailed, and was shown to achieve solution computation times in the order of microseconds.

In Chapter 5 a fixed-point Model Predictive Control framework for spacecraft attitude tracking with reaction wheels actuators was presented. Optimizations to the controller structure to reduce computational load were detailed, including a modified cost function, an external integral action on the reference to guarantee offset-free tracking, and a fixed-point QP solver. As a result, an efficient and lightweight ANSI-C implementation was obtained, with minimal memory and computational requirements, suitable for deployment on low-power embedded devices. Moreover, special MPC formulations that ensure reaction wheel desaturation capabilities, without relying on thrusters actuation, were introduced. They were obtained by either exploiting the gravity gradient effect, i.e., the torque generated by an uneven distribution of the gravity force, or the Earth magnetic field. Both of them were shown to be viable solutions in simulations, with the constraint handling property of MPC being a key factor in enhancing controller performance.

6.2 Future Work

Model Predictive Control for fast embedded implementations is an open and vibrant research topic, and many of the ideas and contributions proposed in this thesis can be refined and extended. Here follow some considerations on possible future work for each of the topics covered in the thesis chapters.

The convergence analysis of the fixed-point Dual Gradient Projection method in Chapter 2 can be extended to the accelerated version of the algorithm. Modifications to the procedure can be introduced to achieve an optimal trade-off between convergence rate and round-off error accumulation.

In Chapter 3, the considerations on the round-off error accumulation in the single iteration of the Proximal Newton algorithm can be embedded in the convergence proof to give theoretical convergence results in the presence of fixed-point arithmetic. Moreover, the performance and error accumulation of other methods for the solution of the linear system can be investigated.

The hardware implementations on ARM Cortex and FPGA devices detailed in Chapter 4 can be exploited to conduct a real process control experiment. Following the results on spacecraft attitude control shown in Chapter 5, one can implement the proposed MPC formulation on air bearing testbeds, which are commonly used to enable a variety of ground testing for spacecraft development.

Bibliography

[1] J. M. Maciejowski, *Predictive control: with constraints*. Pearson education, 2002.

[2] J. B. Rawlings and D. Q. Mayne, *Model predictive control: Theory and design*. Nob Hill Pub., 2009.

[3] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer, 2013.

[4] J. B. Rawlings, "Tutorial overview of model predictive control," *Control Systems, IEEE*, vol. 20, no. 3, pp. 38–52, 2000.

[5] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[6] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control engineering practice*, vol. 11, no. 7, pp. 733–764, 2003.

[7] A. Bemporad, "Model predictive control design: New trends and tools," in *Proc. IEEE Conference on Decision and Control*, 2006, pp. 6678–6683.

[8] R. Bellman, "Dynamic programming," *Princeton University Press.*, 1957.

[9] E. B. Lee and L. Markus, "Foundations of optimal control theory," DTIC Document, Tech. Rep., 1967.

[10] R. R. Bitmead, M. Gevers, and V. Wertz, *Adaptive optimal control: The thinking man's GPC*. Prentice-Hall, 1990.

[11] J. B. Rawlings and K. R. Muske, "The stability of constrained receding horizon control," *Automatic Control, IEEE Transactions on*, vol. 38, no. 10, pp. 1512–1516, 1993.

[12] M. Alamir and G. Bornard, "Stability of a truncated infinite constrained receding horizon scheme: the general discrete nonlinear case," *Automatica*, vol. 31, no. 9, pp. 1353–1356, 1995.

[13] F. Allgöwer, R. Findeisen, and E. Christian, *Nonlinear Model Predictive Control*, 2000.

[14] S. J. Qin and T. A. Badgwell, "An overview of nonlinear model predictive control applications," in *Nonlinear model predictive control*. Springer, 2000, pp. 369–392.

[15] R. Findeisen, L. Imsland, F. Allgöwer, and B. A. Foss, "State and output feedback nonlinear model predictive control:

An overview," *European journal of control*, vol. 9, no. 2, pp. 190–206, 2003.

[16] L. Grüne and J. Pannek, *Nonlinear model predictive control*. Springer, 2011.

[17] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.

[18] M. Lazar, W. Heemels, S. Weiland, and A. Bemporad, "Stabilizing model predictive control of hybrid systems," *Automatic Control, IEEE Transactions on*, vol. 51, no. 11, pp. 1813–1818, 2006.

[19] A. Bemporad, F. Borrelli, M. Morari *et al.*, "Model predictive control based on linear programming˜ the explicit solution," *IEEE Transactions on Automatic Control*, vol. 47, no. 12, pp. 1974–1985, 2002.

[20] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[21] J. Richalet, A. Rault, J. Testud, and J. Papon, "Model predictive heuristic control: Applications to industrial processes," *Automatica*, vol. 14, no. 5, pp. 413–428, 1978.

[22] J. Allwright, "On min-max model-based predictive control," in *Proc. Oxford Symposium on Advances in Model Based Predictive Control*, 1993, pp. 6678–6683.

[23] H. Genceli and M. Nikolaou, "Robust stability analysis of constrained l1-norm model predictive control," *AIChE Journal*, vol. 39, no. 12, pp. 1954–1965, 1993.

[24] Z. Q. Zheng and M. Morari, "Robust stability of constrained model predictive control," in *American Control Conference, 1993*. IEEE, 1993, pp. 379–383.

[25] G. De Nicolao, L. Magni, and R. Scattolini, "Robust predictive control of systems with uncertain impulse response," *Automatica*, vol. 32, no. 10, pp. 1475–1479, 1996.

[26] A. Bemporad, L. Puglia, and T. Gabbriellini, "A stochastic model predictive control approach to dynamic option hedging with transaction costs," in *Proc. American Control Conference*. IEEE, 2011, pp. 3862–3867.

[27] M. Bichi, G. Ripaccioli, S. Di Cairano, D. Bernardini, A. Bemporad, and I. V. Kolmanovsky, "Stochastic model predictive control with driver behavior learning for improved powertrain control," in *Proc. IEEE Conference on Decision and Control*. IEEE, 2010, pp. 6077–6082.

[28] D. Bernardini and A. Bemporad, "Scenario-based model predictive control of stochastic constrained linear systems," in *Proc. IEEE Conference on Decision and Control*. IEEE, 2009, pp. 6333–6338.

[29] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2009.

[30] D. P. Bertsekas, "Nonlinear programming," 1999.

[31] D. P. Bertsekas, A. Nedić, and A. E. Ozdaglar, *Convex analysis and optimization*. Athena Scientific, 2003.

[32] D. P. Bertsekas, *Convex optimization theory*. Athena Scientific, 2009.

[33] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval research logistics quarterly*, vol. 3, no. 1-2, pp. 95–110, 1956.

[34] H. Markowitz, "The optimization of a quadratic function subject to linear constraints," *Naval research logistics Quarterly*, vol. 3, no. 1-2, pp. 111–133, 1956.

[35] C. Hildreth, "A quadratic programming procedure," *Naval research logistics quarterly*, vol. 4, no. 1, pp. 79–85, 1957.

[36] D. Goldfarb and A. Idnani, "A numerically stable dual method for solving strictly convex quadratic programs," *Mathematical programming*, vol. 27, no. 1, pp. 1–33, 1983.

[37] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright, "Procedures for optimization problems with a mixture of bounds and general linear constraints," *ACM Transactions on Mathematical Software*, vol. 10, no. 3, pp. 282–298, 1984.

[38] P. E. Gill, N. I. Gould, W. Murray, M. A. Saunders, and M. H. Wright, "A weighted gram-schmidt method for con-

vex quadratic programming," *Mathematical Programming*, vol. 30, no. 2, pp. 176–195, 1984.

[39] C. Schmid and L. T. Biegler, "Quadratic programming methods for reduced hessian sqp," *Computers & chemical engineering*, vol. 18, no. 9, pp. 817–832, 1994.

[40] R. A. Bartlett and L. T. Biegler, "Qpschur: A dual, active-set, schur-complement method for large-scale and structured convex quadratic programming," *Optimization and Engineering*, vol. 7, no. 1, pp. 5–32, 2006.

[41] 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*, vol. 18, no. 8, pp. 816–830, 2008.

[42] A. V. Fiacco and G. P. McCormick, *Nonlinear programming: sequential unconstrained minimization techniques*. Siam, 1990, vol. 4.

[43] J. E. Dennis Jr and R. B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equations*. Siam, 1996, vol. 16.

[44] J. M. Ortega and W. C. Rheinboldt, *Iterative solution of nonlinear equations in several variables*. Siam, 2000, vol. 30.

[45] W. Murray, "Analytical expressions for the eigenvalues and eigenvectors of the hessian matrices of barrier and penalty

functions," *Journal of Optimization Theory and Applications*, vol. 7, no. 3, pp. 189–196, 1971.

[46] F. A. Lootsma, "Hessian matrices of penalty functions for solving constrained optimization problems," *Philips Research Reports*, vol. 24, pp. 322–331, 1969.

[47] N. Karmarkar, "A new polynomial-time algorithm for linear programming," in *Proc. ACM symposium on Theory of computing*. ACM, 1984, pp. 302–311.

[48] P. E. Gill, W. Murray, M. A. Saunders, J. A. Tomlin, and M. H. Wright, "On projected newton barrier methods for linear programming and an equivalence to karmarkar's projective method," *Mathematical programming*, vol. 36, no. 2, pp. 183–209, 1986.

[49] Y. Nesterov, A. Nemirovskii, and Y. Ye, *Interior-point polynomial algorithms in convex programming*. SIAM, 1994, vol. 13.

[50] A. Forsgren, P. E. Gill, and M. H. Wright, "Interior methods for nonlinear optimization," *SIAM review*, vol. 44, no. 4, pp. 525–597, 2002.

[51] J. Rosen, "Nonlinear programming. the gradient projection method," *Bull. Amer. Math. Soc*, vol. 63, pp. 25–26, 1957.

[52] J. B. Rosen, "The gradient projection method for nonlinear programming. part i. linear constraints," *Journal of the Society for Industrial & Applied Mathematics*, vol. 8, no. 1, pp. 181–217, 1960.

[53] J. Rosen, "The gradient projection method for nonlinear programming. part ii. nonlinear constraints," *Journal of the Society for Industrial & Applied Mathematics*, vol. 9, no. 4, pp. 514–532, 1961.

[54] Y. Nesterov, "A method of solving a convex programming problem with convergence rate o(1/k2)," in *Soviet Mathematics Doklady*, vol. 27, no. 2, 1983, pp. 372–376.

[55] ——, "On an approach to the construction of optimal methods of minimization of smooth convex functions," *Ekonomika i Mateaticheskie Metody*, vol. 24, pp. 509–517, 1988.

[56] ——, *Introductory lectures on convex optimization: A basic course*. Springer, 2004, vol. 87.

[57] ——, "Smooth minimization of non-smooth functions," *Mathematical programming*, vol. 103, no. 1, pp. 127–152, 2005.

[58] J. L. Hennessy and D. A. Patterson, *Computer architecture: a quantitative approach*. Elsevier, 2012.

[59] E. C. Kerrigan, J. L. Jerez, S. Longo, and G. A. Constantinides, "Number representation in predictive control," in *Proc. IFAC Conference on Nonlinear Model Predictive Control*, 2012, pp. 60–67.

[60] J. H. Wilkinson, *Rounding errors in algebraic processes*. Courier Dover Publications, 1994.

[61] M. Urabe, "Roundoff error distribution in fixed-point multiplication and a remark about the rounding rule," *SIAM Journal on Numerical Analysis*, vol. 5, no. 2, pp. 202–210, 1968.

[62] S. Kim and W. Sung, "Fixed-point error analysis and word length optimization of 8× 8 idct architectures," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 8, no. 8, pp. 935–940, 1998.

[63] M. Ogawa *et al.*, "Overflow and roundoff error analysis via model checking," in *Proc. IEEE Conference on Software Engineering and Formal Methods*, 2009, pp. 105–114.

[64] I. Pultarova, "Error propagation formula of multi-level iterative aggregation-disaggregation methods for non-symmetric problems," *Electron. J. Linear Algebra*, vol. 25, pp. 9–21, 2012.

[65] K.-V. Ling, B. F. Wu, and J. Maciejowski, "Embedded model predictive control (mpc) using a fpga," in *Proc. 17th IFAC World Congress*, 2008, pp. 15 250–15 255.

[66] G. Knagge, A. Wills, A. Mills, and B. Ninness, "Asic and fpga implementation strategies for model predictive control," in *Proc. IEEE European Control Conference*, 2009.

[67] E. N. Hartley, J. L. Jerez, A. Suardi, J. M. Maciejowski, E. C. Kerrigan, and G. A. Constantinides, "Predictive control

of a boeing 747 aircraft using an fpga," in *Proc. IFAC Confereince on Nonlinear Model Predictive Control*, 2012.

[68] ——, "Predictive control using an fpga with application to aircraft control," *Control Systems Technology, IEEE Transaction on*, vol. 22, no. 3, pp. 1006–1017, 2013.

[69] J. L. Jerez, G. A. Constantinides, and E. C. Kerrigan, "Towards a fixed point qp solver for predictive control." in *Proc. IEEE Conference on Decision and Control*, 2012, pp. 675–680.

[70] S. Richter, C. N. Jones, and M. Morari, "Real-time input-constrained mpc using fast gradient methods," in *Proc. IEEE Conference on Decision and Control*. IEEE, 2009, pp. 7387–7393.

[71] A. Bemporad and P. Patrinos, "Simple and certifiable quadratic programming algorithms for embedded linear model predictive control," in *Proc. IFAC Nonlinear Model Predictive Control Conference*, vol. 4, no. 1, 2012, pp. 14–20.

[72] V. Nedelcu and I. Necoara, "Iteration complexity of an inexact augmented lagrangian method for constrained mpc," in *Proc. IEEE Conference on Decision and Control*. IEEE, 2012, pp. 650–655.

[73] 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 *Proc. IEEE European Control Conference*. IEEE, 2013, pp. 3614–3620.

[74] 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*, vol. 77, no. 3, pp. 305–321, 2013.

[75] J. L. Jerez, P. J. Goulart, S. Richter, G. A. Constantinides, E. C. Kerrigan, and M. Morari, "Embedded online optimization for model predictive control at megahertz rates," *arXiv:1303.1090*, 2013.

[76] P. Patrinos and A. Bemporad, "An accelerated dual gradient-projection algorithm for embedded linear model predictive control," *Automatic Control, IEEE Transactions on*, vol. 59, no. 1, pp. 18–33, 2014.

[77] 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*, vol. 52, no. 5, pp. 3109–3134, 2014.

[78] M. Rubagotti, P. Patrinos, and A. Bemporad, "Stabilizing linear model predictive control under inexact numerical optimization," *Automatic Control, IEEE Transactions on*, vol. 59, no. 6, pp. 1660–1666, 2014.

[79] I. Necoara and V. Nedelcu, "Rate analysis of inexact dual first-order methods application to dual decomposition," *Automatic Control, IEEE Transactions on*, vol. 59, no. 5, pp. 1232–1243, 2014.

[80] I. Necoara, L. Ferranti, and T. Keviczky, "An adaptive constraint tightening approach to linear mpc based on approximation algorithms for optimization," *J. Optimal Control: Applications and Methods*, vol. 10, pp. 1–19, 2014.

[81] H. Kwakernaak and R. Sivan, *Linear optimal control systems*. John Wiley & Sons, New York, 1972.

[82] P. J. Campo and M. Morari, "Robust control of processes subject to saturation nonlinearities," *Computers & Chemical Engineering*, vol. 14, no. 4,Äì5, pp. 343–358, 1990.

[83] M. V. Kothare, P. J. Campo, M. Morari, and C. N. Nett, "A unified framework for the study of anti-windup designs," *Automatica*, vol. 30, no. 12, pp. 1869–1883, 1994.

[84] L. Magni, G. De Nicolao, and R. Scattolini, "Output feedback and tracking of nonlinear systems with model predictive control," *Automatica*, vol. 37, no. 10, 2001.

[85] L. Magni and R. Scattolini, "Tracking of non-square nonlinear continuous time systems with piecewise constant model predictive control," *Journal of Process Control*, vol. 17, no. 8, pp. 631–640, 2007.

[86] K. R. Muske and T. A. Badgwell, "Disturbance modeling for offset-free linear model predictive control," *Journal of Process Control*, vol. 12, no. 5, pp. 617–632, 2002.

[87] G. Pannocchia and J. B. Rawlings, "Disturbance models for offset-free model-predictive control," *AIChE Journal*, vol. 49, no. 2, pp. 426–437, 2003.

[88] G. Pannocchia, "Robust disturbance modeling for model predictive control with application to multivariable ill-conditioned processes," *Journal of Process Control*, vol. 13, no. 8, pp. 693–701, 2003.

[89] G. Pannocchia and E. C. Kerrigan, "Offset-free receding horizon control of constrained linear systems," *AIChE journal*, vol. 51, no. 12, pp. 3134–3146, 2005.

[90] G. Pannocchia and A. Bemporad, "Combined design of disturbance model and observer for offset-free model predictive control," *IEEE Transactions on Automatic Control*, vol. 52, no. 6, pp. 1048–1053, 2007.

[91] U. Maeder, F. Borrelli, and M. Morari, "Linear offset-free model predictive control," *Automatica*, vol. 45, no. 10, pp. 2214–2222, 2009.

[92] M. Morari and U. Maeder, "Nonlinear offset-free model predictive control," *Automatica*, vol. 48, no. 9, pp. 2059–2067, 2012.

[93] A. Bemporad, M. Morari, and N. Ricker, *Model Predictive Control Toolbox for MATLAB 5.0*. The Mathworks, Inc., 2014, http://www.mathworks.com/access/helpdesk/help/toolbox/mpc/.

[94] G. Pannocchia and J. B. Rawlings, "The velocity algorithm LQR: a survey," Technical Report 2001-01, TWMCC, Tech. Rep., 2001.

[95] L. Wang, "A tutorial on model predictive control: Using a linear velocity-form model," *Developments in Chemical Engineering and Mineral Processing*, vol. 12, no. 5-6, pp. 573–614, 2004.

[96] G. Betti, M. Farina, and R. Scattolini, "An MPC algorithm for offset-free tracking of constant reference signals." in *Proc. Conference on Decision and Control*, 2012, pp. 5182–5187.

[97] ——, "A robust MPC algorithm for offset-free tracking of constant reference signals," *Automatic Control, IEEE Transactions on*, vol. 58, no. 9, pp. 2394–2400, 2013.

[98] A. Bemporad, A. Casavola, and E. Mosca, "Nonlinear control of constrained linear systems via predictive reference management," *Automatic Control, IEEE Transaction on*, vol. AC-42, no. 3, pp. 340–349, 1997.

[99] E. Gilbert, I. Kolmanovsky, and K. T. Tan, "Discrete-time reference governors and the nonlinear control of systems with state and control constraints," *Robust and Nonlinear Control, International Journal of*, vol. 5, no. 5, pp. 487–504, 1995.

[100] A. Bemporad, "Reference governor for constrained nonlinear systems," *Automatic Control, IEEE Transaction on*, vol. AC-43, no. 3, pp. 415–419, 1998.

[101] M. Santillo and A. Karnik, "Model predictive controller design for throttle and wastegate control of a turbocharged engine," in *Proc. American Control Conference*, 2013, pp. 2183–2188.

[102] A. Richards and J. How, "Performance evaluation of rendezvous using model predictive control," in *Proc. AIAA Guidance, Navigation, and Control Conference and Exhibit*. American Institute of Aeronautics and Astronautics, 2003.

[103] H. Park, S. Di Cairano, and I. Kolmanovsky, "Model predictive control for spacecraft rendezvous and docking with a rotating/tumbling platform and for debris avoidance," in *Proc. American Control Conference*, 2011, pp. 1922–1927.

[104] M. Saponara, V. Barrena, A. Bemporad, E. Hartley, J. Maciejowski, A. Richards, A. Tramutola, and P. Trodden, "Model predictive control application to spacecraft rendezvous in mars sample return scenario," *Progress in Flight Dynamics, Guidance, Navigation, Control, Fault Detection, and Avionics*, vol. 6, pp. 137–158, 2013.

[105] E. N. Hartley and J. M. Maciejowski, "Graphical fpga design for a predictive controller with application to spacecraft rendezvous," in *Proc. Conference on Decision and Control*, 2013, pp. 1971–1976.

[106] Ø. Hegrenæs, J. T. Gravdahl, and P. Tøndel, "Spacecraft attitude control using explicit model predictive control," *Automatica*, vol. 41, no. 12, pp. 2107–2114, 2005.

[107] E. Silani and M. Lovera, "Magnetic spacecraft attitude control: a survey and some new results," *Control Engineering Practice*, vol. 13, no. 3, pp. 357–371, 2005.

[108] U. Kalabic, R. Gupta, S. D. Cairano, A. Bloch, and I. Kolmanovsky, "Constrained spacecraft attitude control on SO(3) using reference governors and nonlinear model predictive control," in *Proc. American Control Conference*, 2014, pp. 5586–5593.

[109] C. A. Pascucci, A. Bemporad, S. Bennani, and M. Rotunno, "Embedded mpc for space applications."

[110] Z. Ismail and R. Varatharajoo, "A study of reaction wheel configurations for a 3-axis satellite attitude control," *Advances in Space Research*, vol. 45, no. 6, pp. 750–759, 2010.

[111] J. Jin, S. Ko, and C.-K. Ryoo, "Fault tolerant control for satellites with four reaction wheels," *Control Engineering Practice*, vol. 16, no. 10, pp. 1250–1258, 2008.

[112] G. Creamer, P. DeLaHunt, S. Gates, and M. Levenson, "Attitude determination and control of clementine during lunar mapping," *Journal of guidance, control, and dynamics*, vol. 19, no. 3, pp. 505–511, 1996.

[113] X. Chen, W. H. Steyn, S. Hodgart, and Y. Hashida, "Optimal combined reaction-wheel momentum management for earth-pointing satellites," *Journal of Guidance, Control, and Dynamics*, vol. 22, no. 4, pp. 543–550, 1999.

[114] P. Camillo and F. Markley, "Orbit-averaged behavior of magnetic control laws for momentum unloading," *Journal of Guidance, Control, and Dynamics*, vol. 3, no. 6, pp. 563,Äì–568, 1980.

[115] D. Chang, "Magnetic and momentum bias attitude control design for the hete small satellite," in *Proc. AIAA/USU Conference on Small Satellites*, 1992.

[116] O. Devolder, F. Glineur, and Y. Nesterov, "First-order methods of smooth convex optimization with inexact oracle," *Mathematical Programming*, pp. 1–39, 2013.

[117] P. Patrinos, A. Guiggiani, and A. Bemporad, "Fixed-point dual gradient projection for embedded model predictive control," in *Proc. European Control Conference*, 2013, pp. 3602–3607.

[118] ——, "A Dual Gradient-Projection Algorithm for Model Predictive Control in Fixed-Point Arithmetic," *Automatica*, vol. 55, pp. 226–235, 2015.

[119] P. Patrinos and A. Bemporad, "Proximal Newton methods for convex composite optimization," in *Proc. 52st IEEE Conference on Decision and Control*, 2013, pp. 2358–2363.

[120] A. Guiggiani, P. Patrinos, and A. Bemporad, "Fixed-Point Implementation of a Proximal Newton Method for Embedded Model Predictive Control," in *Proc. IFAC World Congress*, 2014, pp. 2921–2926.

[121] M. Rubagotti, P. Patrinos, A. Guiggiani, and A. Bemporad, "Real-Time Model Predictive Control Based on Dual Gradient Projection: Theory and Fixed-Point FPGA Implementation," *International Journal of Robust and Nonlinear Control*, 2015.

[122] A. Guiggiani, I. Kolmanovsky, P. Patrinos, and A. Bemporad, "Fixed-Point Constrained Model Predictive Control of Spacecraft Attitude," 2015.

[123] ——, "Constrained Model Predictive Control of Spacecraft Attitude with Reaction Wheels Desaturation," 2015.

[124] G. Chen and M. Teboulle, "Convergence analysis of a proximal-like minimization algorithm using Bregman functions," *SIAM Journal on Optimization*, vol. 3, no. 3, pp. 538–543, 1993.

[125] P. Tseng, "On accelerated proximal gradient methods for convex-concave optimization," Department of Mathematics, University of Washington, Tech. Rep., 2008.

[126] H. Bauschke and P. Combettes, *Convex analysis and monotone operator theory in Hilbert spaces*. Springer, 2011.

[127] A. d'Aspremont, "Smooth optimization with approximate gradient," *SIAM Journal on Optimization*, vol. 19, no. 3, pp. 1171–1183, 2008.

[128] O. Devolder, "Stochastic first order methods in smooth convex optimization," *CORE Discussion Papers 2012*, vol. 9, 2012.

[129] S. Richter, M. Morari, and C. Jones, "Towards computational complexity certification for constrained MPC based on Lagrange relaxation and the fast gradient method," in *Proc. Conference on Decision and Control and European Control Conference*, 2011, pp. 5223–5229.

[130] P. Patrinos and A. Bemporad, "An accelerated dual gradient-projection algorithm for linear model predictive control," in *Proc. Conference on Decision and Control*, 2012, pp. 662–667.

[131] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *Control Systems Technology, IEEE Transactions on*, vol. 18, no. 2, pp. 267–278, 2010.

[132] A. van der Sluis, "Stability of solutions of linear algebraic systems," *Numerische Mathematik*, vol. 14, no. 3, pp. 246–251, 1970.

[133] S. Boyd, *Linear matrix inequalities in system and control theory*. Siam, 1994, vol. 15.

[134] S. Richter, C. N. Jones, and M. Morari, "Computational Complexity Certification for Real-Time MPC With Input Constraints Based on the Fast Gradient Method," *Automatic Control, IEEE Transactions on*, vol. 57, no. 6, pp. 1391–1403, 2012.

[135] P. Kapasouris, M. Athans, and G. Stein, "Design of feedback control systems for unstable plants with saturating actuators," in *Proc. IFAC Symp. on Nonlinear Control System Design*, 1990, pp. 302–307.

[136] H. F.-W. Sadrozinski and J. Wu, *Applications of field-programmable gate arrays in scientific research*. CRC Press, 2012.

[137] K. Compton and S. Hauck, "Reconfigurable computing: a survey of systems and software," *ACM Computing Surveys (csuR)*, vol. 34, no. 2, pp. 171–210, 2002.

[138] M. Baleani, F. Gennari, Y. Jiang, Y. Patel, R. Brayton, and A. Sangiovanni-Vincentelli, "System partitioning and timing analysis: Hw/sw partitioning and code generation of embedded control applications on a reconfigurable architecture platform," in *Proc. International Symposium on Hardware/software Codesign*, 2002, pp. 151–156.

[139] K.-V. Ling, S. Yue, and J. Maciejowski, "A fpga implementation of model predictive control," in *Proc. American Control Conference*, 2006, pp. 1930–1935.

[140] L. G. Bleris, P. D. Vouzis, M. G. Arnold, and M. V. Kothare, "A co-processor fpga platform for the implementation of real-time model predictive control," in *Proc. American Control Conference*, 2006, pp. 6–12.

[141] P. Vouzis, M. Kothare, L. Bleris, and M. Arnold, "A system-on-a-chip implementation for embedded real-time model predictive control," *Control Systems Technology, IEEE Transactions on*, vol. 17, no. 5, pp. 1006–1017, 2009.

[142] M. S. Lau, S. Yue, K. Ling, and J. Maciejowski, "A comparison of interior point and active set methods for fpga implementation of model predictive control," in *Proc. European Control Conference*, 2009, pp. 157–161.

[143] N. Yang, D. Li, J. Zhang, and Y. Xi, "Model predictive controller design and implementation on fpga with application to motor servo system," *Control Engineering Practice*, vol. 20, no. 11, pp. 1229–1235, 2012.

[144] D. Limon, I. Alvarado, T. Alamo, and E. F. Camacho, "MPC for tracking piecewise constant references for constrained linear systems," *Automatica*, vol. 44, no. 9, Sep. 2008.

[145] A. Weiss, I. Kolmanovsky, M. Baldwin, and R. Erwin, "Model predictive control of three dimensional spacecraft relative motion," in *Proc. American Control Conference*, 2012, pp. 173–178.

[146] S. Maus, S. Macmillan, S. McLean, B. Hamilton, A. Thomson, M. Nair, and C. Rollins, "The us/uk world magnetic model for 2010-2015," *NOAA Technical Report NESDIS/NGDC*, 2010.