eman ta zabal zazu

Universidad del País Vasco Euskal Herriko Unibertsitatea

Computational study of first-order optimization algorithms for model-based predictive control acceleration on FPGAs

Adriano Navarro Temoche

September, 2020

Abstract

Many of the most recent publications in the embedded MPC field choose to use first-order algorithms as a solution to the complexity of accelerating second-order algorithms in systems with restricted computing resources. Thus, the purpose of this project consist on evaluating the response of these first order algorithms and compare their performance with second-order algorithms to have consistent criteria to design specific accelerators optimized in the development of SoCs with application to the high performance embedded MPC.

To aim this objective, the first part of the text focuses on explaining a background needed to understand the approach of the optimization algorithms in the solution of a reformulated MPC problem arising from a state-space model. In this chapter, definitions such as Shift and Delta discretisation, System Variable and Dynamics Stacking, and Sparse and Dense formulation are explained.

Once the related background is presented, it is analysed in detail the application to the MPC of the most current and efficient first order algorithms, making a study of the proposals published in recent years. In this section, definitions such as optimality, stopping conditions and convergence will be taken into account.

After this analysis, the application of these algorithms will be adapted to different MPC formulation options where it will be evaluated their computational performance in different scenarios.

Lastly, the results obtained of the previous evaluation will be compared with the performance of the second order algorithms, particularly using a simplified Interior Point which it is optimised for an embedded MPC implementation. This comparison will allow to take appropriate design decisions in the implementation of MPC in embedded systems considering aspects such as hardware simplicity, computational performance and numerical characteristics.

Keywords: Model Predictive Control, optimization algorithms, AMA, ADMM, reduced-Hessian method, embedded MPC.

Laburpena

MPC txertatuaren arloko argitalpen berrienetako askok lehen mailako algoritmoak erabiltzen dituzte baliabide informatiko mugatuak dituzten sistemetan bigarren mailako algoritmoen azelerazioaren konplexutasunari aurre egiteko. Beraz, proiektu honen helburua lehen mailako algoritmo horien erantzuna ebaluatzea eta haien errendimendua bigarren mailako algoritmoekin alderatzen da. Horrela, eraginkortasun handiko MPC kontrolagailu txertatuak gauzatzeko SoC arkitektura berriak diseinatzeko beharrezkoak diren azeleragailuak diseinatzeko irizpideak lortu nahi dira.

Helburu hori lortzeko, lan honen lehen atalean, egoera-espazioko eredutik sortzen den birformulatutako MPC problemen ebazpenerako beharrezko diren optimizazio-algoritmoen planteamendua ulertzeko beharrezkoak diren aurrekariak azaltzen dira. Kapitulu honetan, zenbait definizio azaltzen dira, hala nola Shift eta Delta diskretizazioa, aldagaien eta sistemen dinamikaren pilaketa, eta formulazio trinko eta sakabanatuak.

Aurrekariak aurkeztu ondoren, lehen mailako algoritmo berri eta eraginkorrenak MPC-aren arloan nola aplikatu diren aztertzen da, horretarako azken urteetako argitalpen zientifikoetan agertutako proposamenak aztertuz. Atal honetan optimizazioa, bukaera-baldintzak eta konbergentzia bezalako kontzeptuak kontuan hartuko dira. Ondoren, algoritmo horien aplikazioa MPCren formulazio-aukera desberdinetara egokituko da, haien errendimendu konputazionala hainbat egoera esanguratsuetan ebaluatuz.

Azkenik, aurreko ebaluaziotik lortutako emaitzak bigarren mailako algoritmoen errendimenduarekin konparatuko dira, bereziki sistema txertatuetan erabiltzeko optimizatua izan den IP (Interior Point) algoritmo sinplifikatu bat erabilita, MPC txertatua ezartzeko optimizatuta dagoena. Konparazio horri esker, diseinu-erabaki egokiak hartu ahal izango dira sistema txertatuetan MPCa ezartzeko, ondorengo alderdi hauek kontuan hartuta: hardwarearen sinpletasuna, errendimendu konputazionala eta zenbakizko ezaugarriak.

Hitz-gakoak: Kontrol prediktiboa, optimizazio-algoritmoak, AMA, ADMM, Hesiar-murriztuaren metodoa, MPC txertatua.

Resumen

Muchas de las publicaciones más recientes en el campo del MPC embebido optan por utilizar algoritmos de primer orden como solución a la complejidad de la aceleración de los algoritmos de segundo orden en sistemas con recursos informáticos restringidos. Por lo tanto, el propósito de este proyecto consiste en evaluar la respuesta de estos algoritmos de primer orden y comparar su rendimiento con los algoritmos de segundo orden con la finalidad de tener criterios consistentes para diseñar aceleradores específicos optimizados en el desarrollo de SoCs con aplicación al MPC embebido de alto rendimiento.

Para lograr este objetivo, la primera parte del texto se centra en explicar los antecedentes necesarios para entender el enfoque de los algoritmos de optimización en la solución de un problema de MPC reformulado que surge de un modelo en espacio de estados. En este capítulo, se explican definiciones como la discretización Shift y Delta, el apilamiento de las variables y de la dinámica del sistema, y las formulaciones densa y dispersa.

Una vez presentados los antecedentes, se analiza en detalle la aplicación al MPC de los algoritmos de primer orden más actuales y eficientes, haciendo un estudio de las propuestas publicadas en los últimos años. En este apartado se tendrán en cuenta definiciones como la optimización, las condiciones de parada y la convergencia.

Tras este análisis, la aplicación de estos algoritmos se adaptará a las diferentes opciones de formulación de MPC donde se evaluará su rendimiento computacional en diferentes escenarios.

Por último, los resultados obtenidos de la evaluación anterior se compararán con el rendimiento de los algoritmos de segundo orden, en particular utilizando un IP simplificado que está optimizado para una implementación MPC embebida. Esta comparación permitirá tomar decisiones de diseño adecuadas en la implementación de MPC en sistemas embebidos considerando aspectos como la simplicidad del hardware, el rendimiento computacional y las características numéricas.

Palabras clave: Control predictivo, algoritmos de optimización, AMA, ADMM, método de Hessiana-reducida, MPC embebido.

Contents

List of Figures

List of Tables

Acronyms

ADMM Alternating direction method of multipliers.

AMA Alternating minimization algorithm.

FAMA Fast AMA.

FBS Forward-Backward Splitting.

FGM Fast Gradient Method.

FISTA Fast Iterative Shrinkage-Thresholding Algorithm.

FPGA Field-programmable Gate Array.

KKT Karush-Kuhn-Tucker.

LQR Linear-Quadratic Regulator.

LTI Linear time-invariant.

MPC Model Predictive Control.

PGM Proximal Gradient Method.

QP Quadratic problem.

Chapter 1

Introduction

1.1 Motivation

Model Predictive Control (MPC) is recognized due to its ability in handling multiple inputs and outputs with constraints. MPC has been popularly employed in processes where the system dynamics is slow, since the computational time for solving the optimisation problems is greater. In recent years, model predictive control for fast dynamic embedded systems has been in the spotlight. In these embedded systems, the computing platforms usually have limited computation resources.

Existing iterative-based optimisation algorithms for MPC need to solve a system of linear equations based on a Karush-Kuhn-Tucker (KKT) system. In most of cases, it is the main computational load. Thus, accelerating the solution of the KKT system will substantially improve the performance of many MPC optimisation algorithms.

In some recent research papers it is claimed that first-order optimization methods, in contrast to second-order methods should be used to solve MPC problems in embedded implementations due to the fact that they often require simpler arithmetics. Although, first order methods usually converge much more slowly than second order do, they can be more easily parallelized , they are better to larger problems due to the simplicity of the operations and they can be accommodated to fixed-point arithmetics [Jerez et al., 2013].

On the other hand, it is established that first-order algorithms using splitting methods are quite beneficial when applied to large-scale problems [Boyd et al., 2010] and their potential in solving small to medium scale embedded optimisation problems has been studied [Stathopoulos et al., 2016].

In this work, MPC problem is solved with two first-order optimisation algorithms: Alternating minimization algorithm (AMA) and Alternating direction method of multipliers (ADMM). To solve the MPC problem, it is needed to be reformulated into a appropriate formulation. Each reformulation results in a different problem but all of them describe the same MPC setup.

1.2 Project outline

This work starts in Chapter 2 with a brief review focuses on explaining a background needed to understand the approach of the optimization algorithms in the solution of are formulated MPC problem arising from a state-space model. Definitions such as Shift and Delta discretisation, Sparse and Dense formulation, and QP solvers are explained. This chapter is intended mainly for background and can be skipped.

In Chapter 3, first-order methods for embedded MPC are described. Alternating direction method of multipliers (ADMM), Alternating minimization algorithm (AMA) and their accelerated variants are defined.

In Chapter 4, the reduced Hessian method is presented in order to reduce the size of the matrices in a new QP. In addition, Turnback algorithm, a method to obtain banded null space from the left-side of the equality constraints matrix is described. This method is applied to the first-order methods mentioned in the previous chapter.

In Chapter 5, computational analysis is presented for all the algorithms before studied. Tests as scalability, comparison of the best first-order methods with a simplified IP and the performance of these algorithms in closed loop are carried out. With these simulations, conclusions about the stability of iterations and total times according to a selected prediction horizon are obtained.

In Chapter 6, experimental results are shown and their tests are carried out in a embedded platform. In this analysis, closed loop performance for selected algorithms is evaluated where the repeated variables with more computational burden are identified. Finally in Chapter 7, conclusions of this project are presented.

Chapter 2

Related Backgrounds

2.1 Model Predictive Control

The basic idea of MPC is shown in Figure 2.1. Considering a system with discrete-time setting which should follow a set-point, at each sampling time instant, a MPC controller determines a sequence of control inputs by minimising an objective function with constraints over a prediction horizon. This objective function depends on the model of the controlled system since in their constraints, the state-space model of the system is included as well as other restrictions required in the behaviour of the plant.

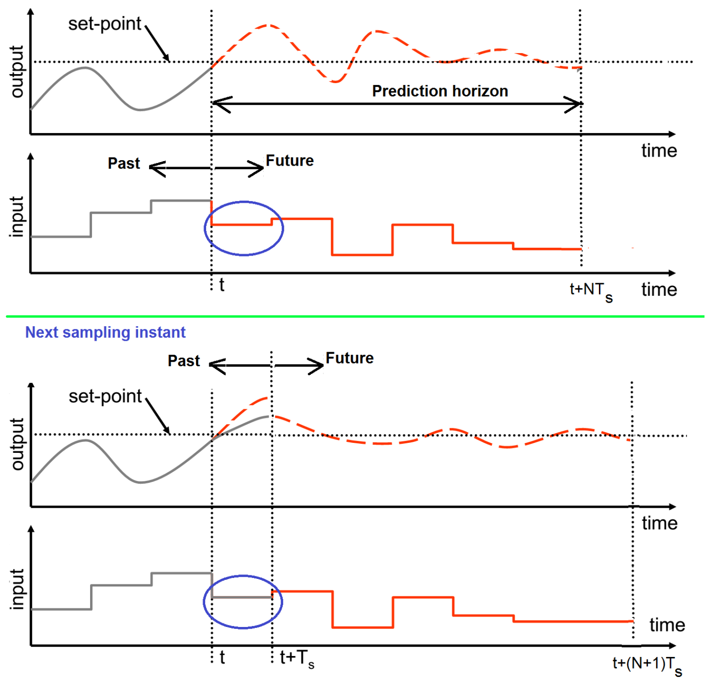


Figure 2.1: Receding Horizon Strategy [Dang, 2018]

The prediction horizon N is the number of steps that the behaviour of the system is predicted being each step the sampling time. To achieve a good prediction, a reliable state-space model is needed. In addition, considering this horizon and minimising the objective function, the ideal inputs to be applied in the following N sampling times are obtained. These set of inputs is also named control sequence.

Other concept to take into account is the control horizon which consists on the number of steps applied in the plant once calculated the control sequence. This horizon will always be less or equal to the prediction horizon and often is defined as 1 which means that only the input corresponding to the next sampling instant of the control sequence is applied to the plant.

In the Figure 2.1, the control horizon es equal to 1 and thus, the control sequence is determined again in next sampling instant. On the contrary, if the control horizon is different to 1, the control sequence is calculated again in the next instant after the application of the control horizon defined. This technique is also known as receding horizon strategy.

In this work, MPC with a linear system, quadratic stage and linear constraints is considered. Thus, its objective function is defined as

$$\begin{aligned} \min \quad & J = \frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^T Q x_k + u_k^T R u_k\right) + \frac{1}{2}x_N^T Q_N x_N \\ s.t. \quad & x_{k+1} = Ax_k + Bu_k \quad \text{for} \quad i = 0, \ldots, N-1 \\ & G_x x_k \leq g_x \ ; \ G_u u_k \leq g_u \\ & G_N x_N \leq g_N \end{aligned} \tag{2.1}$$

where N is the prediction horizon; Q and Q_N are positive semi-definite matrices; R is positive definite matrix; A and B are the matrices of the discrete-time state space model; G_x, G_N, G_u, g_x, g_N, g_u are appropriate matrices and vectors describing the constraints of the system.

Thus, the computational burden of this controller is quite larger than a classical PID controller since the solution in each sampling time is obtained taking into account the model of the plant which is solved reformulating the MPC problem and calculating variables on-line. This is not the case of classical PIDs since they can be tuned manually or using some strategy before staring the control; however, the MPC problem uses the description of the system to predict the future, the objective function that describes what we want to obtain reformulated to a mathematical optimization problem, the control law that described how the problem should be optimized and the optimal control value found with an algorithm based on the requirements of the system.

Therefore, taking into account the statements mentioned in the last paragraph, if it is considered to control a system using embedded platforms, the design complexity for MPC controllers is quite substantial unlike the PIDs. In this design, the complexity involves how to reformulate the MPC problem to convert it into a QP problem, the code size as well as the hardware resources needed to solve this problem, and also the accuracy in case of designing the controller with a word length required.

In addition, the main feature of MPC type controllers type is their natural way of taking constraints into account. On this way, the controllers can work close to that constraints which it is extra beneficial in all applications where the optimal work point lies close to a limit.

2.2 MPC Problem

The state space model of the model of the linear continuous-time system

$$\begin{aligned} \dot{x}_c(t) &= A_c x_c(t) + B_c u_c(t) \\ y_c(t) &= C_c x_c(t) \end{aligned} \tag{2.2}$$

The matrices A_c, B_c and C_c are coefficients corresponding to continuous-time problem.

2.2.1 Discrete Domain

There are two ways to discretise the system. Considering a LTI system represented by the discrete-time state space model

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k \end{aligned} \tag{2.3}$$

where x_k is the state vector at sample instant k, u_k is the input vector and y_k is the output vector.

2.2.1.1 Shift Form

Representation

$$\begin{aligned} x_{k+1} &= A_q x_k + B_q u_k \\ y_k &= C_q x_k \end{aligned} \tag{2.4}$$

The matrices are defined as

$$\begin{aligned} A_q &= e^{A_c \Delta} \\ B_q &= \frac{e^{A_c \Delta} - I}{A_c} B_c \end{aligned} \tag{2.5}$$

where A_q and B_q are the matrices for the shift discrete representation. Δ is the sampling time.

2.2.1.2 Delta Form

In control systems with high sampling rates, especially with oversampling, numerical problems can be severe. Thus, delta discretisation offers the possibility of increasing the sampling rate in the controller by improving control performance and alleviating numerical problems resulting from limitations in computing accuracy.

This performance is relevant in the embedded control with FPGAs since they allow us to work with oversampling which involves defining the sampling frequency above the recommended frequencies in a classical control and thus, the response of the control system is improved [Goodall and Donoghue, 1993].

Furthermore, using this discretisation, the margin to limit the accuracy of the calculation without affecting considerably the response of the controller can be increased. This advantage allows us to make processor designs for MPC by reducing the word length in the arithmetic of the algorithms and, therefore, the consumption of resources and area can be reduced.

Representation

$$\begin{aligned} \delta_k &= A_\delta x_k + B_\delta u_k \\ x_{k+1} &= x_k + \Delta \delta_k \end{aligned} \tag{2.6}$$

Being the matrices

$$\begin{aligned} A_\delta &= \frac{A_q - I}{\Delta} \\ B_\delta &= \frac{B_q}{\Delta} \end{aligned} \tag{2.7}$$

where A_δ and B_δ are the matrices for the shift discrete representation. Δ is the sampling time.

2.3 MPC Formulation

The control of this system is determined by MPC with objective function and constraints. At every sampling instant, given an estimate or measurement of the current state of the plant , the finite-horizon constrained LQR problem is to minimize

$$\begin{aligned} \min \quad & \frac{1}{2}\sum_{k=0}^{N-1} \left(x_k^T Q x_k + u_k^T R u_k\right) + \frac{1}{2} x_N^T Q_N x_N \\ s.t. \quad & x_{k+1} = Ax_k + Bu_k \quad \text{for} \quad i = 0, \dots, N-1 \\ & Jx_k \leq j \\ & Eu_k \leq e \end{aligned} \tag{2.8}$$

Inequality constraints express the box constraints $x_{min_k} \leq x_k \leq x_{max_k}$ and $u_{min_k} \leq x_k \leq u_{max_k}$. Also, these can be represented as $J_t x_k + E_t u_k \leq d$ where J_t, E_t and d are defined as

$$J_t = \begin{bmatrix} J \\ 0 \end{bmatrix}, E_t = \begin{bmatrix} 0 \\ E \end{bmatrix}, d = \begin{bmatrix} j \\ e \end{bmatrix} \tag{2.9}$$

The form of the inequality constraints will affect directly the system dynamics stacking of the problem (dense or sparse). On the other hand, the objective function can be expressed in a more general way,

$$\frac{1}{2}\sum_{k=0}^{N-1}(||x_k - x_{ref_x}||^2_{Q_k} + ||u_k - u_{ref_x}||^2_{R_k}) + \frac{1}{2}||x_N - x_{ref_N}||^2_{Q_N} \tag{2.10}$$

Thus, if it is assumed that x_{ref_x} and x_{ref_x} are equal to zero, first formulation (2.8) is obtained.

2.3.1 System Variable Stacking

Consist on assembling decision variables (inputs and states) over MPC prediction horizon (N) in a single variable.

Simple Stacking

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix}, u = \begin{bmatrix} u_0 \\ \vdots \\ u_{N-1} \end{bmatrix} \longrightarrow z = \begin{bmatrix} x_1 \\ \vdots \\ x_N \\ u_0 \\ \vdots \\ u_{N-1} \end{bmatrix} \tag{2.11}$$

Alternating Stacking

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix}, u = \begin{bmatrix} u_1 \\ \vdots \\ u_{N-1} \end{bmatrix} \longrightarrow z = \begin{bmatrix} u_0 \\ x_1 \\ u_1 \\ \vdots \\ u_{N-1} \\ x_N \end{bmatrix} \tag{2.12}$$

2.4 MPC Reformulation

In MPC, if the system model is linear and the objective function is quadratic, one can formulate the MPC optimisation problem as a sparse QP, keeping both the states and controls as decision variables. In contrast, MPC can also be formulated as a dense QP, keeping only the controls as decision variables. The sparse QP will have both equality and inequality constraints while the dense QP will have only inequality constraints.

2.4.1 Dense QP Formulation

According to [Maciejowski, 2002], considering a discrete-time linear time-invariant model of the plant

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k, \\ y_k &= Cx_k. \end{aligned} \tag{2.13}$$

where $x_k \in \mathbb{R}^n$ is the state vector at sample instant k, $u_k \in \mathbb{R}^l$ is the input vector and $y_k \in \mathbb{R}^m$ is the output vector.

Then, predictions can be described as follows

$$\begin{aligned} \hat{x}(k+1|k) &= Ax(k) + B\hat{u}(k|k), \\ \hat{x}(k+2|k) &= Ax(k+1) + B\hat{u}(k+1|k) \\ &= A^2x(k) + AB\hat{u}(k|k) + B\hat{u}(k+1|k), \\ &\vdots \\ \hat{x}(k+H_p|k) &= A\hat{x}(k+H_p-1|k) + B\hat{u}(k+H_p-1|k) \\ &= A^{H_p}x(k) + A^{H_p-1}B\hat{u}(k|k) + ... + B\hat{u}(k+H_p-1|k) \end{aligned}$$

where H_p is the prediction horizon and H_u is the control horizon.

There will be changes at times $k, k+1, ..., k+H_u-1$ and will remain constant after that $\hat{u}(k+i|k) = \hat{u}(k+H_u-1)$ where $H_u \leq i \leq H_p-1$. Being,

$$\Delta\hat{u}(k+i|k) = \hat{u}(k+i|k) - \hat{u}(k+i-1|k) \tag{2.14}$$

Then,

$$\begin{aligned}
\hat{u}(k|k) &= \Delta\hat{u}(k|k) + u(k-1),\\
\hat{u}(k+1|k) &= \Delta\hat{u}(k+1|k) + \Delta\hat{u}(k|k) + u(k-1),\\
&\vdots\\
\hat{u}(k+H_u-1|k) &= \Delta\hat{u}(k+H_u-1|k) + ... + \Delta\hat{u}(k|k) + u(k-1)
\end{aligned}$$

This is, in a reduced form

$$\hat{u}(k+i|k) = u(k-1) + \sum_{i=0}^{i} \Delta\hat{u}(k+i|k) \tag{2.15}$$

since $\Delta\hat{u}(k+i|k)$ changes respect to the previous value $\Delta\hat{u}(k+i-1|k)$.

So, rewriting the state predictions with variation of input vectors

$$\begin{aligned}
\hat{x}(k+1|k) &= Ax(k) + B[\Delta\hat{u}(k|k) + u(k-1)],\\
\hat{x}(k+2|k) &= A^2x(k) + (A+I)B\Delta\hat{u}(k|k) + B\Delta\hat{u}(k+1|k)\\
&+ (A+I)Bu(k-1),\\
&\vdots\\
\hat{x}(k+H_u|k) &= A^{H_u}x(k) + (A^{H_u-1} + ... + A + I)B\Delta\hat{u}(k|k) + ...\\
&+ B\Delta\hat{u}(k+H_u-1|k) + (A^{H_u-1} + ...A + I)Bu(k-1),\\
\hat{x}(k+H_u+1|k) &= A^{H_u+1}x(k) + (A^{H_u} + ... + A + I)B\Delta\hat{u}(k|k) + ...\\
&+ (A+I)B\Delta\hat{u}(k+H_u-1|k) + (A^{H_u} + ...A + I)Bu(k-1),\\
&\vdots\\
\hat{x}(k+H_p|k) &= A^{H_p}x(k) + (A^{H_p-1} + ... + A + I)B\Delta\hat{u}(k|k) + ...\\
&+ (A^{H_p-H_u} + ... + A + I)B\Delta\hat{u}(k+H_u-1|k)\\
&+ (A^{H_p-1} + ...A + I)Bu(k-1).\\
&= A^{H_p}x(k) + \left[\sum_{i=0}^{H_p-1} A^iB\right]\Delta\hat{u}(k|k) + ...\\
&+ \left[\sum_{i=0}^{H_p-H_u} A^iB\right]\Delta\hat{u}(k+H_u-1|k)\\
&+ \left[\sum_{i=0}^{H_p-1} A^iB\right]u(k-1).
\end{aligned}$$

In matrix form:

$$\begin{aligned}
Y(k) &= Cx(k)\\
Y(k) &= \Psi x(k) + \Upsilon u(k-1) + \Theta\Delta u(k)
\end{aligned} \tag{2.16}$$

where,

$$Y(k) = \begin{bmatrix} \hat{y}(k+1|k) \\ \vdots \\ \hat{y}(k+H_u|k) \\ \hat{y}(k+H_u+1|k) \\ \vdots \\ \hat{y}(k+H_p|k) \end{bmatrix} ; \quad \Delta u(k) = \begin{bmatrix} \Delta\hat{u}(k|k) \\ \vdots \\ \Delta\hat{u}(k+H_u-1|k) \end{bmatrix}$$

$$\Psi = \begin{bmatrix} CA \\ \vdots \\ A^{H_u} \\ A^{H_u+1} \\ \vdots \\ A^{H_p} \end{bmatrix} ; \quad \Upsilon = \begin{bmatrix} CB \\ \vdots \\ C\sum_{i=0}^{H_u-1} A^iB \\ C\sum_{i=0}^{H_u} A^iB \\ \vdots \\ C\sum_{i=0}^{H_p-1} A^iB \end{bmatrix}$$

$$\Theta = \begin{bmatrix} B & \cdots & 0 \\ \vdots & \ddots & \vdots \\ C\sum_{i=0}^{H_u-1} A^iB & \cdots & CB \\ C\sum_{i=0}^{H_u} A^iB & \cdots & C(A+I)B \\ \vdots & \ddots & \vdots \\ C\sum_{i=0}^{H_p-1} A^iB & \cdots & C\sum_{i=0}^{H_p-H_u} A^iB \end{bmatrix}$$

Objective Function

Since $y(k+1)$ must maintain close to the reference $r(k|k)$, a quadratic error is used to minimize it in order to consider the deviations both the positives and the negative ones.

$$\|r(k+i|k) - \hat{y}(k+i|k)\|^2 \tag{2.17}$$

where, $i = 1, 2, ..., H_p$.

In addition, seeking for minimizing $\Delta\hat{u}(k+1|k$ to avoid rough control actions

$$\|\Delta\hat{u}(k+i|k)\|^2 \quad i = 0, 1, ..., H_u - 1. \tag{2.18}$$

The objective of carrying out a control function is defined

$$\nu(k+i) = \|r(k+i|k) - \hat{y}(k+i|k)\|^2_{\alpha_i} + \|\Delta\hat{u}(k+j|k)\|^2_{\beta_j} \tag{2.19}$$

for $i = 1, 2, ..., n$ and $j = 0, 1, ..., n-1$. Being, α_i and β_j are scalar weight factors.

Also denoted as

$$\nu(k+i) = \sum_{i=1}^{H_p} \|r(k+i|k) - \hat{y}(k+i|k)\|^2_{\alpha_i} + \sum_{i=0}^{H_u-1} \|\Delta\hat{u}(k+i|k)\|^2_{\beta_i} \tag{2.20}$$

In matrix form,

$$\nu(k) = \|\Theta\Delta u(k) - E(k)\|^2_Q + \|\Delta u(k)\|^2_R \tag{2.21}$$

if $E(k) = Y_r(k) - \Psi x(k) - \Upsilon u(k-1)$.

Eq.(2.21) can be developed taking into account that $\|x\|_M^2 \equiv x^T M x$. Then,

$$x = \Theta \Delta u - E; \quad x^T = (\Theta\Delta u - E)^T = \Delta u^T \Theta^T - E^T \tag{2.22}$$

Thus, the objective function is equivalent

$$\begin{aligned}\nu(k) &= (\Delta u(k)^T\Theta^T - E(k)^T)Q(\Theta\Delta u(k) - E(k)) + \Delta u(k)^T R\Delta u(k)\\ &= \Delta u(k)^T\Theta^T Q\Theta\Delta u(k) - E(k)^T Q\Theta\Delta u(k) - \Delta u(k)^T\Theta^T QE(k)\\ &+ E(k)^T QE(k) + \Delta u(k)^T R\Delta u(k)\\ &= \Delta u(k)^T\Theta^T Q\Theta\Delta u(k) - 2\Delta u(k)^T\Theta^T QE(k) + E(k)^T QE(k) + \Delta u(k)^T R\Delta u(k)\\ &= E(k)^T QE(k) - 2\Delta u(k)^T\Theta^T QE(k) + \Delta u(k)^T(\Theta^T Q\Theta + R)\Delta u(k)\end{aligned}$$

where,

$$Q = \begin{bmatrix}\alpha(1) & \cdots & 0\\ \vdots & \vdots & \vdots\\ 0 & \alpha(i) & \vdots\\ \vdots & \ddots & \vdots\\ 0 & \cdots & \alpha(H_p)\end{bmatrix} \quad R = \begin{bmatrix}\beta(0) & \cdots & 0\\ \vdots & \vdots & \vdots\\ 0 & \beta(i) & \vdots\\ \vdots & \ddots & \vdots\\ 0 & \cdots & \beta(H_u)\end{bmatrix} \tag{2.23}$$

Since $E(k)^T QE(k) = \mathit{constant}$, objective function is equal

$$\nu(k) = \frac{1}{2}\Delta u(k)^T H\Delta u(k) + h^T\Delta u(k) \tag{2.24}$$

where,

$$H = 2(\Theta^T Q\Theta + R) \tag{2.25}$$

$$h = -2(\Theta^T QE(k)) \tag{2.26}$$

Finally, minimizing $\nu(k)$ with linear constraints in $\Delta u(k)$ is a quadratic-convex optimization problem.

Linear constraints

$$F\begin{pmatrix}\hat{u}(k|k)\\ \vdots\\ \hat{u}(k+H_u-1|k)\\ 1\end{pmatrix} \le 0; \quad L\begin{pmatrix}\hat{y}(k+1|k)\\ \vdots\\ \hat{y}(k+H_p|k)\\ 1\end{pmatrix} \le 0; \quad E\begin{pmatrix}\Delta\hat{u}(k|k)\\ \vdots\\ \Delta\hat{u}(k+H_u-1|k)\\ 1\end{pmatrix} \le 0$$

To minimize $\nu(k)$ subject to constraints imposed to $\Delta u(k)$.

- First constraint is equal to $\sum_{i=0}^{H_u-1} F_i\hat{u}(k+i|k)+f \le 0$ where $F = [F_1, F_2, ..., F_{H_u}, f]$. Since $\hat{u}(k+i-1|k) = u(k-1) + \sum_{j=0}^{i-1}\Delta\hat{u}(k+j|k)$, constraints can be written as

$$\begin{aligned}&\sum_{j=1}^{H_u} F_j\Delta\hat{u}(k|k) + \sum_{j=2}^{H_u} F_j\Delta\hat{u}(k+1|k) + ... + F_{H_u}\Delta\hat{u}(k+H_u-1|k)\\ &\qquad + \sum_{j=1}^{H_u} F_j u(k-1) + f \le 0\end{aligned} \tag{2.27}$$

If $\mathbb{F}_i = \sum_{j=i}^{H_u} F_j$ and $\mathbb{F} = [\mathbb{F}_0, \mathbb{F}_1, ..., \mathbb{F}_{H_u}]$. Then,

$$\mathbb{F}\Delta u(k) \leq -\mathbb{F}_1 u(k-1) - f \tag{2.28}$$

- Now, doing the same for the second constraint. Assuming full state measurements, $L = [\Gamma, l]$ can be defined as

  $$L \begin{pmatrix} \Psi x(k) + \Upsilon u(k-1) + \Theta \Delta u(k) \\ 1 \end{pmatrix} \leq 0 \tag{2.29}$$

  It is the same

  $$\Gamma[\Psi x(k) + \Upsilon u(k-1) + \Theta \Delta u(k)] + l \leq 0 \tag{2.30}$$

  Resulting

  $$\Gamma\Theta\Delta u(k) \leq -\Gamma[\Psi x(k) + \Upsilon u(k-1)] - l \tag{2.31}$$

- Here, the expression is formulated in function of $\Delta(u(k))$. Then, with $E = [W, -\omega]$ it only remains to put the inequality into the form

  $$W\Delta(u(k)) \leq \omega \tag{2.32}$$

Therefore, inequalities can be assembled before described into the single inequality

$$\begin{bmatrix} \mathbb{F} \\ \Gamma\Theta \\ W \end{bmatrix} \Delta u(k) \leq \begin{bmatrix} -\mathbb{F}_1 u(k-1) - f \\ -\Gamma[\Psi x(k) + \Upsilon u(k-1)] - l \\ \omega \end{bmatrix} \tag{2.33}$$

Simplifying the inequality

$$G\Delta u(k) \leq g \tag{2.34}$$

In summary, the dense QP formulation is obtained

$$\begin{aligned} \min_{\Delta u(k)} \quad & \frac{1}{2}\Delta u(k)^T H \Delta u(k) + h^T \Delta u(k) \\ s.t. \quad & G\Delta u(k) \leq g. \end{aligned} \tag{2.35}$$

2.4.2 Sparse QP formulation

At every sampling instant, given an estimate or measurement of the current state of the plant $\hat{x}$, the finite-horizon constrained Linear-Quadratic Regulator (LQR) problem is to minimize

$$\begin{aligned} \min_{u} \quad & \sum_{k=0}^{N-1} (x_k^T Q x_k + u_k^T R u_k) + x_N^T Q_N x_N \\ s.t. \quad & x_{k+1} = A x_k + B u_k \\ & G_x x_k \leq g_x \\ & G_u u_k \leq g_u \\ & G_N x_N \leq g_N \end{aligned} \tag{2.36}$$

where $k = 0, 1, \ldots, N-1$; N is equal to H_p which is the prediction horizon; Q_x, $Q_N \in \mathbb{R}^{n_x \times n_x}$ are positive semi-definite matrices; $R \in \mathbb{R}^{n_u \times n_u}$ is positive definite matrix; $G_x \in \mathbb{R}^{n_{ix} \times n_x}$, $G_N \in \mathbb{R}^{n_{it} \times n_x}$, $G_u \in \mathbb{R}^{n_{iu} \times n_u}$, $g_x \in \mathbb{R}^{n_{ix}}$, $g_N \in \mathbb{R}^{n_{it}}$, $g_u \in \mathbb{R}^{n_{iu}}$ are appropriate matrices and vectors describing the constraints of the system (n_{ix}, n_{it}, n_{iu} are number of the state constraints, terminal constraints and input constraints).

Minimization function can be described in matrix form

$$J = \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}^T \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} + x_N^T Q_N x_N \tag{2.37}$$

By using sparse formulation which keeps both the states and control as decision variables

$$z = (u_k, x_{k+1}, u_1, x_{k+2}, \dots, u_{k+N-1}, x_{k+N}) \tag{2.38}$$

With respect to the constraints

- *Equality constraint*: As it is known, in sparse formulation, system dynamics is left as equality constraints. Thus, it must be expressed respect to the new matrix of decision variables, resulting

$$Fz = f \tag{2.39}$$

where, $F \in \mathbb{R}^{n_{eq} \times n_d}$ and $f \in \mathbb{R}^{n_{eq}}$ being $n_{eq} = Nn_x$, , the number of equality constraints:

$$F = \begin{bmatrix} -B & I & 0 & \cdots & 0 \\ 0 & -A & -B & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & I \end{bmatrix} \quad f = \begin{bmatrix} Ax_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{2.40}$$

- *Inequality constraints*: A ordered matrix in function of the inequalities described in Eq.(2.36) must be created, resulting

$$Gz \leq g \tag{2.41}$$

where, $G \in \mathbb{R}^{n_{ineq} \times n_d}$ and $g \in \mathbb{R}^{n_{ineq}}$ being $n_{ineq} = (N-1)(n_{ix} + n_{iu}) + n_{it} + n_{iu}$, the number of inequality constraints:

$$G = \begin{bmatrix} G_u & 0 & \cdots & 0 & 0 \\ 0 & G_x & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & G_u & 0 \\ 0 & 0 & \cdots & 0 & G_N \end{bmatrix} \quad g = \begin{bmatrix} g_u \\ g_x \\ \vdots \\ g_u \\ g_N \end{bmatrix} \tag{2.42}$$

Thus, problem (2.36) can be written as

$$\begin{aligned} \min_z \quad & \frac{1}{2} z^T H z \\ s.t. \quad & Fz = f \\ & Gz \leq g \end{aligned} \tag{2.43}$$

where $H \in \mathbb{R}^{n_d \times n_d}$ is positive semi-definite ($n_d = N(n_x + n_u)$ is the number of decision variables). H have the following description

$$H = \begin{bmatrix} R & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & Q & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & R & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & Q & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & R & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & Q_N \end{bmatrix} \tag{2.44}$$

Finally, in comparison to the general QP formulation

$$\begin{aligned} \min_{z} \quad & \frac{1}{2} z^T H z + h^T z \\ s.t. \quad & Fz = f \\ & Gz \leq g \end{aligned} \tag{2.45}$$

the linear expression of the objective function disappears since **h:=0**.

In summary, in sparse formulation, the future states can be kept as decision variables and the system dynamics can be incorporated into the problem by enforcing equality constraints. In this case, the vector z is created with both variables and all the matrices are banded and sparse. Also, the number of non-zero elements in those matrices grows linearly with the horizon length.

2.5 QP solvers

Once formulated the QP according to the formulations previously seen, a suitable algorithm to solve this optimisation problem is needed.

Second order algorithms such as interior-point methods (IPM) and active-set (ASM) are two commonly employed solvers in the optimisation field. The main computational load of these methods is the solution of a set of linear equations at every iteration, and this can be the bottleneck for embedded systems using these methods. However, recent publications have given guidelines to overcome this bottleneck. Particularly, in [del Rio Ruiz and Basterretxea, 2019], it is described a robust methodology to design fixed-point processing modules for the hardware acceleration of linear solvers in IP algorithms. In this article, it is pointed out the need for a simplified IP algorithm for a more hardware friendly scheme without perceptible negative impact on the control performance.

In addition, in terms of embedded systems, FPGAs and hybrid platforms (SoCs) are really useful to test the performance of each enhance due to the fact that they are embedded platforms which have a highly parallel computation capability. Thus, they are often chosen to tackle the computation demanding task of solving the system of linear equations [Hartley et al., 2012] [Liu et al., 2014].

Recently, first-order QP solvers, such as gradient-based methods and splitting algorithms have been in the spotlight because of their simpler structure, low cost implementation, efficient parallelism, and fixed-point arithmetic opportunity. Due to its simplicity in concept and underlying algebraic calculations, splitting methods are gaining particularly popular for embedded MPC applications.

Fast gradient method is one of the most known first order algorithms and it is particularly attractive for application to MPC in embedded control system design due both to the relative ease of implementation and to the availability of strong performance certification guarantees. However, its use is limited to cases in which the projection operation in these algorithms is simple, e.g. in the case of box constrained inputs. This method is not convenient when state constraints is included in the problem since they change the geometry of the feasible set such that the projection subproblem is as difficult as the original problem. Therefore, fast gradient method (FGM) is often applied in cases where only input constraints are present, and splitting algorithms such as AMA and ADMM are used for cases in which both state and input constraints are present. Indeed, the sparse formulations, used by ADMM and AMA, are constructed taking into account both constraints.

For this reason, in this work, the analysis will be focused on these last two algorithms.

2.6 Splitting method

Operator splitting method are used by first-order optimization algorithms to reduce the complexity of the problem. This splitting can be developed for both dense and sparse formulation. For simplicity, **sparse formulation** will be analysed; however, this procedure can be replicated for a dense formulation deleting the equality constraints.

Considering the following optimal control problem to define the MPC problem

$$\begin{aligned} \min_{z} \quad & \frac{1}{2} z^T H z + h^T z \\ \text{s.t.} \quad & Gz \leq g \\ & Fz = f \end{aligned} \tag{2.46}$$

where, $H \in \mathbb{R}^{n \times n}$ positive definite. $F \in \mathbb{R}^{p \times n}$, $G \in \mathbb{R}^{q \times n}$ and $z \in \mathbb{R}^n$, $h \in \mathbb{R}^n$, $f \in \mathbb{R}^p$, $g \in \mathbb{R}^q$ are vectors.

As it is mentioned before, Eq.(2.46) must be modified to be applied to splitting algorithms. Thus, it is achieved inserting a slack variable v to the inequality constraints. Therefore, the transformation of the inequalities are described as follows:

$$\begin{aligned} & Gz \leq g \\ & 0 \leq g - Gz \\ & 0 \leq v = g - Gz \end{aligned} \tag{2.47}$$

Now,

$$\begin{aligned} Gz{+}v &= g \\ v &\geq 0 \end{aligned} \tag{2.48}$$

In order to express the same problem, Eq.(2.48) must be expressed as an indicator function:

$$I_v(v) = \begin{cases} 0 & if \quad v \geq 0 \\ \infty & otherwise \end{cases} \tag{2.49}$$

It will be inserted as an additional function, named *g(v)*, to be minimized in (2.46). Thus, the *sparse* formulation ends up being defined as

$$\begin{aligned} \min_{z,v} \quad & f(z) + g(v) \\ \text{s.t.} \quad & Az + Bv = c \end{aligned} \tag{2.50}$$

where,

$$f(z) = \frac{1}{2} z^T H z + h^T z \tag{2.51}$$

$$A = \begin{bmatrix} G \\ F \end{bmatrix}, \quad B = \begin{bmatrix} I \\ 0 \end{bmatrix}, \quad c = \begin{bmatrix} g \\ f \end{bmatrix} \tag{2.52}$$

On the other hand, if a **dense formulation** is needed, the equality constraints are deleted of the Eq.(2.46) and following the insertion of the slack variable, the problem is described as

$$\begin{aligned} \min_{z,v} \quad & f(z) + g(v) \\ \text{s.t.} \quad & Gz + v = g \end{aligned} \tag{2.53}$$

where,

$$f(z) = \frac{1}{2} z^T H z + h^T z \tag{2.54}$$

2.6.1 Lagrangian

First-order algorithms work with the Lagrangian of the constraints to solve the optimization problem. So, the Lagrangian for the formulation in (2.50), is denoted as

$$L(z, v, y) = f(z) + g(v) + y^T(Az + Bv - c) \tag{2.55}$$

where y is a Lagrange multiplier. Thus, its gradient in function of z:

$$\nabla L_z = \frac{\partial f(z)}{\partial z} + \frac{\partial g(v)}{\partial z} + \frac{\partial y^T(Az + Bv - c)}{\partial z} \tag{2.56}$$

The solution for the first parameter:

$$\frac{\partial f(z)}{\partial z} = \frac{1}{2}(H^T + H)z + h \tag{2.57}$$

Due to the fact that H is symmetric, it is the same that its transpose and the derivative of the first parameter is defined:

$$\frac{\partial f(z)}{\partial z} = Hz + h \tag{2.58}$$

Now, the third parameter is solved since the second one is zero:

$$\frac{\partial y^T(Az + Bv - c)}{\partial z} = A^T y \tag{2.59}$$

Therefore, the gradient of the Lagrangian results:

$$\nabla L_z = Hz + h + A^T y \tag{2.60}$$

2.6.2 Augmented Lagrangian

An extended version of the Lagrangian is used in some splitting algorithms. So, it is necessary to define both its expression and its gradient.

Augmented Lagrangian for the formulation in (2.50) is defined as

$$L_\rho(z, v, y) = f(z) + g(v) + y^T(Az + Bv - c) + \frac{\rho}{2}\|Az + Bv - c\|_2^2 \tag{2.61}$$

where ρ is the penalty parameter which in some cases can be selected to increase the convergence rate.

The gradient of Eq.(2.61) must be calculated. As it is seen in the previous subsection, the gradient for a standard Lagrangian is obtained so only it is necessary to calculate the derivative of the Euclidean norm attached to the penalty parameter. This expression will be obtained using indicial notation:

$$Q_i = (Az + Bv - c)_i = A_{ij}z_j + B_{ij}v_j - c_i \tag{2.62}$$

$$M = \|.\|_2^2 = Q_iQ_i = (A_{ij}z_j + B_{ij}v_j - c_i)(A_{iq}z_q + B_{iq}v_q - c_i) \tag{2.63}$$

$$\frac{\partial M}{\partial z_p} = A_{ij}\delta_{jp}(A_{iq}z_q + B_{iq}v_q - c_i) + (A_{ij}x_j + B_{ij}v_j - c_i)A_{iq}\delta_{qp} \tag{2.64}$$

where, δ is the Kronecker delta:

$$\frac{\partial z_j}{\partial z_p} = \delta_{jp} = \begin{cases} 1 & if \quad j = p \\ 0 & if \quad j \neq p \end{cases} \tag{2.65}$$

$$\frac{\partial z_q}{\partial z_p} = \delta_{qp} = \begin{cases} 1 & if \quad q = p \\ 0 & if \quad q \neq p \end{cases} \tag{2.66}$$

Then,

$$\begin{aligned} \frac{\partial M}{\partial z_p} &= A_{ip}A_{iq}z_q + A_{ip}B_{iq}v_q - A_{ip}c_i + A_{ij}A_{ip}z_j + B_{ij}A_{ip}v_j - A_{ip}c_i \\ &= 2A_{ip}(A_{iq}z_q + B_{iq}v_q - c_i) \\ &= 2A_{ip}Q_i \end{aligned} \tag{2.67}$$

Taking into account:

$$[R(n \times n)y(n \times 1)]_s = \sum_{t=1}^{n} R_{st}y_t \tag{2.68}$$

Thus,

$$\frac{\partial M}{\partial z_p} = 2(A^T Q)_p = (\nabla M)_p \tag{2.69}$$

Therefore, the gradient of an Euclidean norm:

$$\nabla_z M = 2A^T(Az + B\upsilon - c) \tag{2.70}$$

Finally, the gradient of the augmented Lagrangian is described as

$$\begin{aligned}\nabla_z L_p(z, \upsilon, y) &= Hz + h + A^T y + \rho A^T(Az + B\upsilon - c) \\ &= (H + \rho A^T A)z + [q + \rho A^T(B\upsilon + \frac{y}{\rho} - c)]\end{aligned} \tag{2.71}$$

Chapter 3

First-order algorithms for embedded MPC

According to [McInerney et al., 2018], FPGA implementations have recently used first-order methods such as Nesterov's Fast Gradient Method (FGM), or Alternating direction method of multipliers (ADMM). These methods utilize only first-order information in their computations, and have generally three main steps:

- Compute a search direction.
- Compute the step size and apply the search direction.
- Project onto the feasible set.

According to [Shukla et al., 2017], the key feature of splitting methods is that each of the three steps is computationally cheap and often has a closed form solution.

In this work, two well-known methods are analysed in order to know more about their structure and performance: Alternating minimization algorithm (AMA) and Alternating direction method of multipliers (ADMM).

For ease of reference, the sparse QP is recalled:

$$\begin{aligned} \min_{x,z} \quad & f(z) + g(v) \\ \text{s.t.} \quad & Az + Bv = c \end{aligned} \tag{3.1}$$

where,

$$f(z) = \frac{1}{2} z^T H z + h^T z \tag{3.2}$$

$$A = \begin{bmatrix} G \\ F \end{bmatrix}, \quad B = \begin{bmatrix} I \\ 0 \end{bmatrix}, \quad c = \begin{bmatrix} g \\ f \end{bmatrix} \tag{3.3}$$

3.1 ADMM

In agreement with [Boyd et al., 2010], there are precursors to the alternating method of multipliers. These concepts are not crucial in the development of the algorithm, however, it provides some useful background.

3.1.1 Dual Ascent

Consider the equality-constrained convex optimization problem

$$\begin{array}{ll} \min & f(z) \\ \text{s.t.} & Az = b, \end{array} \tag{3.4}$$

with variable $z \in \mathbb{R}^n$, where $A \in \mathbb{R}^{m\times n}$. As it is know, Lagrangian for this problem is

$$L(z, y) = f(z) + y^T(Az - b) \tag{3.5}$$

and the dual function is

$$g(y) = \inf_x L(z, y) = -f^*(-A^T y) - b^T y, \tag{3.6}$$

where y is the dual variable or Lagrange multiplier, and f^* is the convex conjugate of f.

The dual function was obtained taking into account as convex conjugate

$$f^*(u) = z^T u - f(z) \tag{3.7}$$

So, Lagrangian in Eq.(3.5)

$$L(z, y) = f(z) + y^T(Az - b) = f(z) + z^T A^T y - b^T y \tag{3.8}$$

And its derivative

$$\frac{\partial f}{\partial z} + A^T y = 0, \tag{3.9}$$

Also expressed as

$$\frac{\partial f}{\partial z} = -A^T y, \tag{3.10}$$

Thus, the first and second element of the Lagrangian can be written as

$$f(z) + z^T A^T y = f(z) + z^T(-\frac{\partial f}{\partial z}), \tag{3.11}$$

It can be represented using the definition of the Eq.(3.7)

$$f(z) + z^T A^T y = f(z) + z^T(-(-A^T y)) = -f^*(-A^T y). \tag{3.12}$$

Therefore, the dual function results as it is described in Eq.(3.6).

On the other hand, the dual problem will be

$$\max \quad g(y), \tag{3.13}$$

with variable $y \in \mathbb{R}^m$. Assuming that strong duality holds, the optimal values of the primal and dual problem are the same. A primal optimal point x^* from a dual optimal point y^* can be recovered as

$$z^* = \min_z L(z, y^*), \tag{3.14}$$

provided there is only one minimizer of $L(z, y^*)$.

Assuming that g is differentiable, the gradient $\nabla g(y)$ can be evaluated as follows.

$$z_{k+1} = \min L(z, y_k) \tag{3.15}$$

$$y_{k+1} = y_k + \alpha^k (Az_{k+1} - b), \tag{3.16}$$

where $\alpha^k > 0$ is a step size. This algorithm is called dual ascent since the dual function increases in each step.

In the *dual ascent method*, the dual problem using gradient ascent is solved. Note that the dual function might not be differentiable. To guarantee that f^* is differentiable, it is assumed that f is strictly convex. In some cases, this prevent us from simply solving the dual problem with gradient ascent. If the dual function is not differentiable, the *proximal point method* can be used which does not require the objective function to be differentiable. As a result of applying the proximal point method to the dual problem, *method of multipliers* is obtained.

3.1.2 Dual Decomposition

Dual ascent method has the benefit which can lead to a decentralized algorithm in some cases. For instance, if f is *separable*, meaning that

$$f(z) = \sum_{i=1}^{N} f_i(z_i), \tag{3.17}$$

where $z = (z_1, ..., z_N)$ and variables z_i are subvectors of z.

This means that the z-minimization step (3.15) splits into N separate problems that can be solved in parallel. Explicitly, the algorithm is

$$z^i_{k+1} = \min_{z_i} L_i(z^i, y_k) \tag{3.18}$$

$$y_{k+1} = y_k + \alpha^k (Az_{k+1} - b). \tag{3.19}$$

In this case, the dual ascent method is referred as *dual decomposition.*

3.1.3 Method of Multipliers

Also known as *Augmented Lagrangian method.* Here, it is necessary to consider the *augmented Lagrangian* for Eq.(3.4)

$$L_\rho(z, y) = f(z) + g(v) + y^T(Az - b) + \frac{\rho}{2}\|Az - b\|_2^2 \tag{3.20}$$

being the penalty parameter ρ> 0. The augmented Lagrangian can be viewed as the ordinary Lagrangian associated with the problem

$$\begin{aligned} \min \quad & f(z) + \frac{\rho}{2}\|Az - b\|_2^2 \\ \text{s.t.} \quad & Az = b. \end{aligned} \tag{3.21}$$

This problem is clearly equivalent to the original problem Eq.(3.4), since for any feasible x the term added to the objective is zero. The associated dual function is $g_\rho(y) = \inf L_\rho(z, y)$.

According to [Boyd et al., 2010], the benefit of including the penalty term is that g_ρ can be shown to be differentiable under mild conditions on the original problem. Thus, applying dual ascent to the modified problem yields the algorithm

$$z_{k+1} = \min_z L_\rho(z, y_k) \tag{3.22}$$

$$y_{k+1} = y_k + \rho(Az_{k+1} - b). \tag{3.23}$$

which is known as *method of multipliers* for solving Eq.(3.4). It is worth highlighting that the method of multipliers converges under far more general conditions than dual ascent, including cases when f is not strictly convex.

3.1.4 Algorithm

According to [Boyd et al., 2010], ADMM is an algorithm that is intended to blend the decomposability of dual ascent with the superior convergence properties of the method of multipliers. The algorithm solves problems in the form (2.50). The only difference from the general linear equality-constrained problem (3.4) is that variable x has been split into two parts, with the objective function separable across this splitting.

As in the method of multipliers, the augmented Lagrangian is formed and it is expressed in the Eq.(2.61) for the problem defined in Eq.(2.50). ADMM consists of the iterations

$$z_{k+1} := \min_z L_\rho(z, \upsilon_k, y_k) \tag{3.24}$$

$$\upsilon_{k+1} := \min_\upsilon L_\rho(z_{k+1}, \upsilon, y_k) \tag{3.25}$$

$$y_{k+1} = y_k + \rho(Az_{k+1} + B\upsilon_{k+1} - c), \tag{3.26}$$

where $\rho > 0$.

ADMM can be writen in a slightly different form by combining the linear and quadratic terms in the augmented Lagrangian and scaling the dual variable y.

If $r = Az + B\upsilon - c$:

$$\begin{aligned} y^T r + \frac{\rho}{2}\|r\|_2^2 &= (\frac{\rho}{2})\|r + \frac{y}{\rho}\|_2^2 - (\frac{1}{2\rho})\|y\|_2^2 \\ &= (\frac{\rho}{2})\|r + \tau\|_2^2 - (\frac{\rho}{2})\|\tau\|_2^2, \end{aligned} \tag{3.27}$$

where $\tau = \frac{y}{\rho}$ is the scaled dual variable.

Using the scaled dual variable, it can express ADMM as

$$z_{k+1} := \min_z (f(z) + \frac{\rho}{2}\|Az + B\upsilon_k - c + \tau_k\|_2^2), \tag{3.28}$$

$$\upsilon_{k+1} := \min_\upsilon (g(\upsilon) + \frac{\rho}{2}\|Az_{k+1} + B\upsilon - c + \tau_k\|_2^2), \tag{3.29}$$

$$\tau_{k+1} = \tau_k + Az_{k+1} + B\upsilon_{k+1} - c. \tag{3.30}$$

This formulation is named *scaled form* which is exactly the same that the unscaled form.

3.1.5 z-update

According to [Boyd et al., 2010], following the scaled form, if z-update step is expressed as

$$z_{k+1} = \min_z (f(z) + \frac{\rho}{2})\|Az - \nu\|_2^2) \tag{3.31}$$

where $\nu = -B\upsilon_k + c - \tau_k$. In case A=I, z-update is

$$z^{k+1} = \min_z (f(z) + \frac{\rho}{2}\|z - \nu\|_2^2) = prox_{\rho,f} \tag{3.32}$$

This in variational analysis is known as the Moreau envelope and it is connected to the theory of the proximal point algorithm.

In addition, if f is the indicator function of a closed nonempty convex set C, the z-update is

$$z_{k+1} = \min_z (f(z) + \frac{\rho}{2})\|z - \nu\|_2^2) = \Pi_C(\nu) \tag{3.33}$$

where Π_C denotes projection onto C.

It is important to know that the procedure for the minimization of z is symmetric for υ-update. In our case, the sub-problem for the z-update in Eq.(3.24) is an unconstrained QP and has the unique solution

$$z_{k+1} = -(H + \rho A^T A)^{-1}[h + \rho A^T (B\upsilon + \frac{y}{\rho} - c)] \tag{3.34}$$

for Eq.(2.71) equal to zero.

3.1.6 υ-update

The sub-problem for the υ-update in Eq.(3.25) is derived from the definition of proximal operator described in [Stathopoulos et al., 2016].

As it was mentioned, for a function f, its *proximal operator* $prox_f$ is defined as

$$\mathbf{prox}_f := \min_x (f(\upsilon) + \frac{1}{2}\|\upsilon - x\|^2), \tag{3.35}$$

Also defined, according to [Parikh and Boyd, 2013], in a scaled function λf where $\lambda> 0$ and it is called as the proximal operator of f with parameter λ.

$$\mathbf{prox}_{\lambda f}(\kappa) = \min_{x}(f(\upsilon) + \frac{1}{2\lambda}\|\upsilon - \kappa\|^2). \tag{3.36}$$

Thus,

$$\mathbf{prox}_{\frac{1}{\rho}f} = \min_{x}(f(\upsilon) + \frac{\rho}{2}\|\upsilon - x\|^2), \tag{3.37}$$

Resulting the Moreau envelope described in Eq.(3.32).

Since there is a special case which indicates that when g is the indicator function of a closed convex set $\Upsilon = \{\upsilon : \upsilon \geq 0\}$, its proximal operator $prox_{\frac{1}{\rho}g}$ reduces to projection onto Υ. According to [Dang et al., 2015], this notation for projection can be reduced for this sub-problem as

$$\Pi_{\Upsilon}(\phi) = \min_{\upsilon\in\Upsilon} \|\upsilon - \phi\|_2^2 \tag{3.38}$$

Now, doing the same operation calculated for z-update:

$$\upsilon_{k+1} = \min_{\upsilon}(g(\upsilon) + \frac{\rho}{2}\|B\upsilon - \phi\|_2^2) \tag{3.39}$$

where $\phi = -Az_{k+1} - \tau_k + c$.

Due to the fact that υ only operates in inequality constraints

$$\upsilon_{k+1} = \min_{\upsilon}(g(\upsilon) + \frac{\rho}{2}\|\upsilon - \phi\|_2^2) = \Pi_{\Upsilon}(\phi) \tag{3.40}$$

where $\phi = -Gz_{k+1} - \tau_k^g + g$.

To demonstrate that the projection corresponds to the original sub-problem

$$\begin{aligned}\Pi_{\Upsilon}(-Gz^{k+1} - \tau_k^g + g) &= \min_{\upsilon}(g(\upsilon) + \frac{\rho}{2}\|\upsilon + Gz_{k+1} + \tau_k^g - g\|_2^2)\\ &= \min_{\upsilon} L_\rho(z_{k+1}, \upsilon, y_k)\end{aligned} \tag{3.41}$$

Therefore, solution for a projection on the box type set:

$$\upsilon_{k+1} = \pi_{\Upsilon}(-Gz_{k+1} - \tau_k{}^g + g) = max\{0, -Gz_{k+1} - \tau_k{}^g + g\}, \tag{3.42}$$

where τ_k is the scaled dual variable and g are the values of c corresponding to the dimension of inequality constraints.

On the other hand, another way to obtain this result is operating as was done for z-update but now calculating the derivative of augmented Lagrangian respect to υ

$$\nabla_\upsilon L_\rho(z, \upsilon, y) = \rho B^T(Az_{k+1} + B\upsilon + \tau_k - c) = 0 \tag{3.43}$$

Since υ is only present in the inequality constraints, the expression before obtained can be reduced as

$$0 = \rho(Gz_{k+1} + \upsilon + \tau_k^g - g), \tag{3.44}$$

So, to solve υ-update it is necessary to take into account these conditions
(1) $\upsilon = -Gz_{k+1} - \tau_k^g + g$, obtained from Eq.(3.44).
(2) $\upsilon \geq 0$, obtained from indicator function *g(υ)*.

Then, in terms of transcription in a code, it is the same that calculate the maximum between 0 and the first condition ($max\{0, -Gz_{k+1} - \tau_k^g + g\}$) since the result never will be negative. On this way, conditions will always be fulfilled.

3.1.7 Optimality and Stopping conditions

According to [Boyd et al., 2010], the necessary and sufficient optimality conditions for the ADMM problem are primal feasibility

$$Az^* + B\upsilon^* - c = 0, \tag{3.45}$$

and dual feasibility,

$$0 \in \partial f(z^*) + A^T y^* \tag{3.46}$$

$$0 \in \partial g(\upsilon^*) + B^T y^* \tag{3.47}$$

Here, ∂ denotes the sub differential operator since when f and g are differentiable, the sub differentials ∂f and ∂g can be replaced by their gradients and $\in$ can be replaced by $=$.

Since υ_{k+1} minimizes $L_\rho(z_{k+1}, \upsilon, y_k)$ by definition,

$$\begin{aligned} 0 &\in \partial g(\upsilon_{k+1} + B^T y_k + \rho B^T(Az_{k+1} + B\upsilon_{k+1} - c)), \\ &= \partial g(\upsilon_{k+1} + B^T y_k + \rho B^T r_{k+1}), \\ &= \partial g(\upsilon_{k+1} + B^T y_{k+1}), \end{aligned} \tag{3.48}$$

This means that υ_{k+1} and y_{k+1} always satisfy Eq.(3.47), so attaining optimality comes down to satisfying Eq.(3.45) and Eq.(3.46). This phenomenon is analogous to the iterates of the method of multipliers always being dual feasible.

Since z_{k+1} minimizes $L_\rho(z, \upsilon_k, y_k)$ by definition,

$$\begin{aligned} 0 &\in \partial f(z_{k+1} + A^T y_k + \rho A^T(Az_{k+1} + B\upsilon_k - c)), \\ &= \partial f(z_{k+1} + A^T(y_k + \rho r_{k+1} + \rho B(\upsilon_k - \upsilon_{k+1})), \\ &= \partial f(z_{k+1} + A^T y_{k+1} + \rho A^T B(\upsilon_k - \upsilon_{k+1})), \end{aligned} \tag{3.49}$$

or equivalently,

$$\rho A^T B(\upsilon_{k+1} - \upsilon_k) \in \partial f(z_{k+1} + A^T y_{k+1}) \tag{3.50}$$

This means that the quantity

$$s_{k+1} = \rho A^T B(\upsilon_{k+1} - \upsilon_k) \tag{3.51}$$

can be viewed as a residual for the dual feasibility condition (3.46). Then, s_{k+1} as the *dual residual* at iteration *k+1* and to $r_{k+1} = Az_{k+1} + B\upsilon_{k+1} - c$ as the *primal residual* at iteration *k+1*.

In summary, the optimality conditions for the ADMM problem consist of three conditions, Eq.(3.45–3.47). The last condition (3.47) always holds for $(z_{k+1}, \upsilon_{k+1}, y_{k+1})$; the residuals for the other two, (3.45) and (3.46), are the primal and dual residuals r_{k+1} and s_{k+1}, respectively. These two residuals converge to zero as ADMM proceeds.

Finally, a reasonable termination criterion is that the primal and dual residuals must be small

$$\|r_k\|_2 \leq \varepsilon_{primal} \quad and \quad \|s_k\|_2 \leq \varepsilon_{dual}. \tag{3.52}$$

where $\varepsilon_{primal} > 0$ and $\varepsilon_{dual} > 0$ are feasibility tolerances for the primal and dual feasibility conditions, respectively.

Algorithm 1 ADMM

```
Require: $z_0, \upsilon_0, \tau_0$ fixed, $\rho > 0$
  while $\|r_{k+1}\| \geq \epsilon_{primal}$ or $\|s_{k+1}\| \geq \epsilon_{dual}$ do
    $z_{k+1} = -(H + \rho A^T A)^{-1}[h + \rho A^T(B\upsilon_k + \tau_k - c)]$
    $\upsilon_{k+1} = max\{0, -Gz_{k+1} - \tau_k{}^g + g\}$
    $\tau_{k+1} = \tau_k + Az_{k+1} + B\upsilon_{k+1} - c$
  end while
```

3.1.8 Indirect Indicator formulation

According to [Manickathu2016] the indirect splitting can be applied in problem (2.46) to facilitate the implementation of ADMM and AMA. Using a slack variable ω and the indicator functions $I_{Fz=f}(z)$, $I_{\omega\geq 0}(\omega)$, the constraints are shifted to the objective by demanding the constraint $Gz + \omega = g$. Thus, the following QP is obtained

$$\begin{aligned} \min_{z,\omega} \quad & \frac{1}{2}\|z\|_H^2 + h^T z + I_{Fz=f}(z) + I_{\omega\geq 0}(\omega) \\ s.t. \quad & Gz + \omega = g \end{aligned} \tag{3.53}$$

For this setup, the first step of ADMM requires solving following KKT equation

$$\begin{bmatrix} H + \rho G^T G & F^T \\ F & 0 \end{bmatrix} \begin{bmatrix} z_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} -(h + \rho G^T(\tau_k + \omega_k - g)) \\ f \end{bmatrix} \tag{3.54}$$

where θ_{k+1} is the Lagrangian multiplier associated with an equality-constrained minimisation in the first step of the ADMM algorithm.

With respect to the step-size, it is heuristic selected, based on [Gadhimi] and it is written as

$$\rho^* = (\sqrt{\lambda_{min,>0}(GH^{-1}G^T)\lambda_{max,>0}(GH^{-1}G^T)})^{-1} \tag{3.55}$$

3.1.9 Theoretical Step-sizes

An important factor is the choice of the step size. Several papers have written on how to choose an optimal step-size ρ^*. This step size is optimal with respect to the

Algorithm 2 Indirect ADMM

```
Require: $z_0, \omega_0, \tau_0$ fixed.
  $\rho^* = (\sqrt{\lambda_{min,>0}(GH^{-1}G^T)\lambda_{max,>0}(GH^{-1}G^T)})^{-1}$
  while $\|r_{k+1}\| \geq \epsilon_{primal}$ or $\|s_{k+1}\| \geq \epsilon_{dual}$ do
    $z_{k+1} = solve(3.54)$
    $\omega_{k+1} = max\{0, -Gz_{k+1} - \tau_k + g\}$
    $\tau_{k+1} = \tau_k + Gz_{k+1} + \omega_{k+1} - g$
  end while
```

worst-case convergence rate.

An optimal step size is usually specified for a certain formulation and demands certain requirements of the problem structure to be applied. If there is not an optimal step size because of some of the requirements are not met, an heuristic approach might yield a good performance.

Raghunathan and Di Cairano without Equality Constraint

According to [Raghunathan and Di Cairano, 2014], optimal step size is calculated for a QP with the following structure

$$\begin{aligned} \min_{y,w} \quad & \frac{1}{2}y^T Q y + q^T y \\ s.t. \quad & y = w \\ & w \in \gamma \end{aligned} \tag{3.56}$$

where Q is positive definite and symmetric and γ is a nonempty polyhedron $\gamma = [y_{min}, y_{max}]$.

Thus, the optimal step-size, as it is mentioned in [Raghunathan and Di Cairano, 2014], for this formulation is

$$\rho^* = \sqrt{\lambda_{min}(Q)\lambda_{max}(Q)} \tag{3.57}$$

and the corresponding convergence rate is

$$\zeta^* = \frac{\frac{1}{\lambda_{min}(Q)}}{\frac{1}{\lambda_{min}(Q)} + \frac{1}{\sqrt{\lambda_{min}(Q)\lambda_{max}(Q)}}} \tag{3.58}$$

where $\lambda(M)$ is the calculation of eigenvalues of the matrix M.

Raghunathan and Di Cairano with Equality Constraint

As it is mentioned in the Direct formulation for the ADMM algorithm, in the article written by [Raghunathan and Cairano, 2015], an optimal step size is obtained for convex QPs which have the following structure

$$\begin{aligned} \min_{y} \quad & \frac{1}{2}y^T Q y + q^T y \\ s.t. \quad & Ay = b \\ & y \in \gamma \end{aligned} \tag{3.59}$$

where Q is assumed to be symmetric positive semidefinite and positive definite on the null space of equality constraints, $(Z^TQZ > 0)$, being Z an orthogonal basis for the null space of the matrix A. Furthermore, $\gamma = [y_{min}, y_{max}]$ are defined as box constraints.

Thus, the optimal step size for QPs which have this form is given as

$$\rho^* = \sqrt{\lambda_{min}(Z^TQZ)\lambda_{max}(Z^TQZ)} \tag{3.60}$$

This ρ^* is optimal if the matrix A is full row rank.

Gadhimi and Teixeira

According to [Ghadimi et al., 2015], ADMM algorithm solves QPs of the form

$$\begin{aligned} \min_{y} \quad & \frac{1}{2}x^TQx + q^Tx + I_+(z) \\ s.t. \quad & Ax - c + z = 0 \end{aligned} \tag{3.61}$$

where Q is assumed to be symmetric and positive definite and A is either full row-rank or invertible.

It is shown in [Ghadimi et al., 2015] that the convergence rate ζ can be arbitrarily close to 1 when the rows of A are linearly dependent. Thus, the optimal step size ρ^* and convergence rate ζ^* are

$$\rho^* = \frac{1}{\sqrt{\lambda_{min}(AQ^{-1}A^T)\lambda_{max}(AQ^{-1}A^T)}} \tag{3.62}$$

$$\zeta^* = \frac{\lambda_{max}(AQ^{-1}A^T)}{\lambda_{max}(AQ^{-1}A^T) + \sqrt{\lambda_{min}(AQ^{-1}A^T)\lambda_{max}(AQ^{-1}A^T)}} \tag{3.63}$$

3.2 AMA

Another splitting method is the Alternating minimization algorithm (AMA). According to [Shukla et al., 2017], AMA comes down to the following three numerical operations:

- Solving a system of linear equations;
- Matrix-vector multiplications;
- Vector-vector manipulations.

As it is known, the most computational costly operation is solving a linear system of equations. It is often the case that splitting methods give rise to a KKT system of the form

$$\begin{bmatrix} K_{11} & K_{21}^T \\ K_{21} & 0 \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

Fortunately, for splitting methods, the KKT matrix does **NOT** change over iterations (except very few anomaly cases) and thus one can precompute the inverse or matrix factorization. It is important to note that matrices involved are often structured. Thus, one can exploit this fact while either solving the linear system of equations or computing matrix-vector multiplications.

According to [Stathopoulos et al., 2016], AMA derives from applying Proximal Gradient Method (PGM) to the dual problem. The method requires smoothness of *f*, a property that can be recovered if and only if the function f is strongly convex.

For this algorithm, the Eq.(3.1) will be taken into account where f is a quadratic objective function defined over a linear system and g is an indicator function on a set (e.g positive orthant).

Therefore, AMA is defined as

$$z_{k+1} = \min_{z} L(z, \upsilon_k, \tau_k) \tag{3.64}$$

$$\upsilon_{k+1} = \min_{\upsilon} L_\rho(z_{k+1}, \upsilon, \tau_k) \tag{3.65}$$

$$\tau_{k+1} = \tau_k + (Az_{k+1} + B\upsilon_{k+1} - c) \tag{3.66}$$

where $\tau = \frac{y}{\rho}$.

3.2.1 z-update

Since this update is based only in the ordinary Lagrangian, it is necessary to use its calculation applied to a QP which was developed in subsection (2.6.1). Thus, equalising to zero the gradient with respect to x of the Lagrangian, z_{k+1} is obtained:

$$\nabla L_z = Hz + h + A^T y = 0 \tag{3.67}$$

$$z_{k+1} = -H^{-1}(h + A^T y_k) \tag{3.68}$$

3.2.2 υ-update

After this point, all proceedings are exactly the same as for ADMM since AMA also uses the augmented Lagrangian for υ-update and therefore, dual variable update does not change.

3.2.3 Stopping Conditions

Here, it is necessary to accomplish the same conditions defined for ADMM which are expressed in Eq.(3.45-3.47). As it was mentioned, only the first step of the algorithm changes in AMA in respect to ADMM. Thus, *primal residual* is the same and *dual residual* is defined in function of Eq.(3.46) since it is used the ordinary Lagrangian instead of the augmented one.

$$\begin{aligned} 0 &\in \partial f(z_{k+1} + A^T y_k), \\ &= \partial f(z_{k+1} + A^T(y_{k+1}) + A^T(y_k - y_{k+1})), \end{aligned} \tag{3.69}$$

or equivalently,

$$A^T(y_{k+1} - y_k) \in \partial f(z_{k+1} + A^T y_{k+1}) \tag{3.70}$$

Then, *dual residual* is

$$s_{k+1} = A^T(y_{k+1} - y_k) \tag{3.71}$$

Algorithm 3 AMA

```
Require: $z_0, \nu_0, \tau_0$ fixed, $\rho > 0$
  while $\|r_{k+1}\| \geq \epsilon_{primal}$ or $\|s_{k+1}\| \geq \epsilon_{dual}$ do
    $z_{k+1} = -(H)^{-1}[h + \rho A^T \tau_k]$
    $\nu_{k+1} = max\{0, -Gx_{k+1} - \tau_k{}^g + g\}$
    $\tau_{k+1} = \tau_k + Az_{k+1} + B\nu_{k+1} - c$
  end while
```

In summary, according to [Stathopoulos et al., 2016], compared to AMA, ADMM only differs in the minimization of the augmented Lagrangian function in the first step. This trait has the advantage that there is not *step-size restrictions* for ADMM.

On the other hand, the augmented Lagrangian minimization complicates the first step by the addition of a (possibly dense) quadratic form. This is not the case with AMA, where first actualization remains simple.

3.2.4 Dual Problem and Convergence

Although ADMM is guaranteed to convergence for any positive scalar step-size, this is not the case for AMA. Convergence of AMA is only guaranteed if the step-size satisfies a condition involving a function, which is part of the objective of the dual problem.

Recalling the Lagrangian, developed in section 2.6.1, of the problem described in Eq.(3.1) as

$$\begin{aligned} L(z, \upsilon, y) &= f(z) + g(\upsilon) + y^T(Az + B\upsilon - c) \\ L(z, \upsilon, y) &= f(z) + g(\upsilon) + y^T Az + y^T B\upsilon - y^T c \\ L(z, \upsilon, y) &= f(z) + g(\upsilon) + z^T A^T y + \upsilon^T B^T y - y^T c \\ L(z, \upsilon, y) &= f(z) + \langle z, A^T y\rangle + g(\upsilon) + \langle \upsilon, B^T y\rangle - y^T c \end{aligned}$$

The dual objective $D(\lambda)$ is given by

$$\begin{aligned} D(y) &= \inf_{z,\upsilon} L(z, \upsilon, y) \\ D(y) &= \inf_{z,\upsilon}(f(z) + \langle z, A^T y\rangle + g(\upsilon) + \langle \upsilon, B^T y\rangle - y^T c) \\ D(y) &= \inf_{z}(f(z) + \langle z, A^T y\rangle) + \inf_{\upsilon}(g(\upsilon) + \langle \upsilon, B^T y\rangle) - y^T c \end{aligned}$$

Using the definition of the conjugate function

$$h^*(y) = \sup_x(\langle x, y\rangle - h(x)) = -\inf_x(h(x) + \langle x, y\rangle) \tag{3.72}$$

this expression can be written as

$$D(y) = -f^*(-A^T y) - g^*(-B^T y) - y^T c \tag{3.73}$$

Thus, the dual of the problem (3.1) has the form

$$\max_y \quad D(y) = \underbrace{-f^*(-A^T y)}_{-F(y)} - g^*(-B^T y) - c^T y \tag{3.74}$$

As it was mentioned, f is a strongly convex and g is a convex function. Applying AMA to the primal problem (3.1) is equivalent with applying the FBS method to problem (3.74). Convergence of FBS applied to the dual problem is guaranteed under the condition $\rho < \frac{2}{L(\nabla F(y))}$.

Assuming that f is a strongly convex function with convexity modulus σ_f and g is a convex function, not necessarily smooth, convergence of AMA is guaranteed if

$$0 < \rho < \frac{2}{L(\nabla F(y))} \tag{3.75}$$

In general condition, (3.75) is difficult to impose as f^* might not have a trivial expression. A stricter condition is

$$0 < \rho < \frac{2\sigma_f}{\lambda_{max}(A^T A)} \tag{3.76}$$

which is really helpful if σ_f is known. Nonetheless, if σ_f is equal to zero, the objective function is not strongly convex and AMA cannot be used.

3.2.5 Indirect Indicator formulation

In the previous algorithm, an indirect indicator formulation was developed considering the problem (2.46). In this subsection, the AMA method is applied to this problem. So, the iterates are the same as (3.64-3.66).

The main difficulty in this case is the first step since the second iterate requires a clipping and the third iterate is a simple dual update. Therefore, following the same analysis which is made for ADMM, the first step of AMA requires solving following KKT equation

$$\begin{bmatrix} H & F^T \\ F & 0 \end{bmatrix} \begin{bmatrix} z_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} -(h + \rho G^T \tau_k) \\ f \end{bmatrix} \tag{3.77}$$

In terms of convergence speed, the step-size selected is

$$0 < \rho < \frac{2\sigma_f}{\lambda_{max}(F^T F)} \tag{3.78}$$

3.3 Accelerated Variants

The size of residuals indicates how far the iterates are from a solution. There are multiple ways to accelerate the convergence of the first-order methods such as over-relaxation, hot starting, the usage of variable step-size or the Nesterov's acceleration.

In this work, accelerated versions of basic algorithms are obtained such that these residuals decay quickly. Particularly, only Nesterov's acceleration will be considered to increase the convergence speed of these two methods before shown. Thus, considering the QP (3.1), Fast AMA and Fast ADMM are explained.

Algorithm 4 Indirect AMA

```
Require: Initialize $\tau_0$
  $\rho = \frac{\sigma_f}{\lambda_{max}(F^T F)}$
  while $\|r_{k+1}\| \geq \epsilon_{primal}$ or $\|s_{k+1}\| \geq \epsilon_{dual}$ do
    $z_{k+1} = solve(3.77)$
    $\omega_{k+1} = max\{0, -Gz_{k+1} - \tau_k + g\}$
    $y_{k+1} = y_k + Gz_{k+1} + \omega_{k+1} - g$
  end while
```

3.3.1 Fast AMA

By applying the Nesterov-type acceleration step to the dual variable τ, an accelerated method is obtained.

Algorithm 5 Fast AMA

```
Require: $\tau_0 = \hat{\tau}_1$, $\alpha_1 = 1$
  while $\|r_{k+1}\| \geq \epsilon_{primal}$ or $\|s_{k+1}\| \geq \epsilon_{dual}$ do
    $z_{k+1} = -(H)^{-1}[h + \rho A^T \hat{\tau}_k]$
    $\upsilon_{k+1} = max\{0, -Gz_{k+1} - \hat{\tau}_k^g + g\}$
    $\tau_{k+1} = \hat{\tau}_k + Az_{k+1} + B\upsilon_{k+1} - c$
    $\alpha_{k+1} = \frac{1+\sqrt{1+4\alpha_k^2}}{2}$
    $\hat{\tau}_{k+1} = \tau_{k+1} + \frac{\alpha_k - 1}{\alpha_{k+1}}(\tau_{k+1} - \tau_k)$
  end while
```

The primal residual

$$r_{k+1} = Az_{k+1} + B\upsilon_{k+1} - c \tag{3.79}$$

And the dual residual

$$s_{k+1} = \frac{A^T}{\rho}(\tau_{k+1} - \hat{\tau}_k) \tag{3.80}$$

Step size selection for FAMA

A theorical analysis of the convergence of FAMA can be found in [Pu, 2016]. The dual formulation obtained in Eq.(3.74) is taking into account in convergence analysis. FAMA is the dual method of FISTA and their well-studied convergence properties can be transferred to FAMA. The condition that ensures convergence in AMA is established in a similar manner using the duality with the FBS method. Thus, the step-size is defined as

$$0 < \rho < \frac{1}{L(\nabla F(\nu))} \tag{3.81}$$

3.3.2 Fast ADMM for Strongly Convex Problems

According to [Goldstein et al., 2014], an accelerated variant of ADMM are listed in Algorithm 6. The accelerated method is simply ADMM with a predictor-corrector type acceleration step. This step is stable only in the special case where both objective terms in QP are strongly convex.

Algorithm 6 Fast ADMM for Strongly Convex Problems

```
Require: $\upsilon_0 = \hat{\upsilon}_1, \tau_0 = \hat{\tau}_1$, $\rho > 0$, $\alpha_1 = 1$
  while $\|r_{k+1}\| \geq \epsilon_{primal}$ or $\|s_{k+1}\| \geq \epsilon_{dual}$ do
    $z_{k+1} = -(H + \rho A^T A)^{-1}[h + \rho A^T(B\hat{\upsilon}_k + \hat{\tau}_k - c)]$
    $\upsilon_{k+1} = max\{0, -Gz_{k+1} - \hat{\tau}_k^g + g\}$
    $\tau_{k+1} = \hat{\tau}_k + Az_{k+1} + B\upsilon_{k+1} - c$
    $\alpha_{k+1} = \frac{1+\sqrt{1+4\alpha_k^2}}{2}$
    $\hat{\upsilon}_{k+1} = \upsilon_{k+1} + \frac{\alpha_k - 1}{\alpha_{k+1}}(\upsilon_{k+1} - \upsilon_k)$
    $\hat{\tau}_{k+1} = \tau_{k+1} + \frac{\alpha_k - 1}{\alpha_{k+1}}(\tau_{k+1} - \tau_k)$
  end while
```

In the accelerated case, the primal residual is unchanged $r_{k+1} = Az_{k+1} + B\upsilon_{k+1} - c$; a simple derivation yields the following new dual residual

$$s_{k+1} = \rho A^T B(\upsilon_{k+1} - \hat{\upsilon}_k) \tag{3.82}$$

3.3.3 Fast ADMM for Weakly Convex Problems

In this case, ADMM can still be accelerated, however it cannot guarantee a global convergence rate as in the strongly convex case. For weakly convex problems, it must be enforced stability using a restart rule. This rule relies on a combined residual, which measures both primal and dual error simultaneously.

$$c_k = \frac{1}{\rho}\|\tau_{k+1} - \hat{\tau}_k\|^2 + \rho\|B(\upsilon_{k+1} - \hat{\upsilon}_k)\|^2. \tag{3.83}$$

The first term of the combined residual measures the primal residual whereas the second term is closely related to the dual residual but differs in that is does not require multiplication by A^T.

Algorithm 7 Fast ADMM with Restart

```
Require: $\upsilon_0 = \hat{\upsilon}_1, \tau_0 = \hat{\tau}_1$, $\rho > 0$, $\alpha_1 = 1$, $\eta \in (0, 1)$
  while $\|r_{k+1}\| \geq \epsilon_{primal}$ or $\|s_{k+1}\| \geq \epsilon_{dual}$ do
    $z_{k+1} = -(H + \rho A^T A)^{-1}[h + \rho A^T(B\hat{\upsilon}_k + \hat{\tau}_k - c)]$
    $\upsilon_{k+1} = max\{0, -Gz_{k+1} - \hat{\tau}_k^g + g\}$
    $\tau_{k+1} = \hat{\tau}_k + Az_{k+1} + B\upsilon_{k+1} - c$
    $c_k = \rho^{-1}\|\tau_{k+1} - \hat{\tau}_k\|^2 + \rho\|B(\upsilon_{k+1} - \hat{\upsilon}_k)\|^2$
    if $c_k < \eta c_{k-1}$ then
      $\alpha_{k+1} = \frac{1+\sqrt{1+4\alpha_k^2}}{2}$
      $\hat{\upsilon}_{k+1} = \upsilon_{k+1} + \frac{\alpha_k - 1}{\alpha_{k+1}}(\upsilon_{k+1} - \upsilon_k)$
      $\hat{\tau}_{k+1} = \tau_{k+1} + \frac{\alpha_k - 1}{\alpha_{k+1}}(\tau_{k+1} - \tau_k)$
    else
      $\alpha_{k+1} = 1$, $\hat{\upsilon}_{k+1} = \upsilon_k$, $\hat{\tau}_{k+1} = \tau_k$
      $c_k = \eta^{-1} c_{k-1}$
    end if
  end while
```

Algorithm 7 involves a parameter $\eta \in (0, 1)$. Because it is desirable to restart the method as infrequently as possible, a value for η close to 1 is recommended. In [Goldstein et al., 2014], $\eta = 0.999$ is used.

Step-size selection for FADMM

There is no publication dealing with an optimal step-size ρ^* for FADMM. However, in case of a QP with inequality constraints , the step-size described in [Ghadimi et al., 2015] can be used as heuristic depending on the row rank of the constraint matrix. On the other hand, in case of the QP has both equality and inequality constraints, the step-size developed in [Raghunathan and Di Cairano, 2014] can also be considered as a heuristic approach.

Chapter 4

Null Basis for Embedded MPC

4.1 Reduced-Hessian Method

According to [Dang et al., 2017], *null space method* in optimization is usually referred to as reduced Hessian approach. The basic idea is to describe the equality constraints by using a null basis and a particular solution.

The main purpose of using this method is to reduce the number of decision variables in a QP problem with sparse formulation. This reduction can be achieved eliminating the equality constraints that represent the system dynamics using the null space of these constraints. On this way, a new QP problem is generated with a reduced-size matrices.

Considering a *sparse* QP formulation and an *Alternating* Variable Stacking

$$\begin{aligned} \min_{z} \quad & \frac{1}{2} z^T H z \\ s.t. \quad & Fz = f \\ & Gz \leq g \end{aligned} \tag{4.1}$$

The solution of equality constraints of the QP can be parametrised by

$$z = z_0 + Yy \tag{4.2}$$

where z_0 is the **particular solution** of $Fz_0 = f$ and Y is a **null basis** of F ($FY = 0$). Therefore, QP (4.1) can be solved via solving the following equivalent problem:

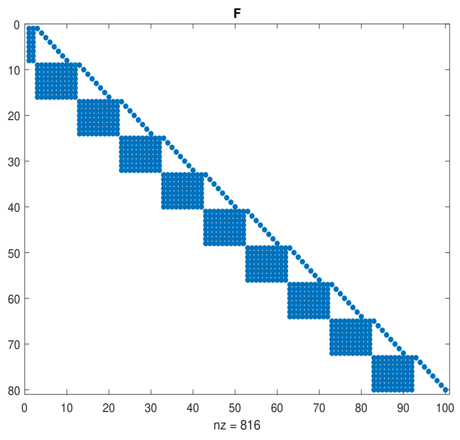
$$\begin{aligned} \min_{y} \quad & \frac{1}{2} y^T \tilde{H} y + (z_0^T H_0) y \\ s.t. \quad & \tilde{G} y \leq g - Gz_0 \triangleq \tilde{g} \end{aligned} \tag{4.3}$$

where $\tilde{H} = Y^T HY$, $\tilde{G} = GY$ and $H_0 = HY$.

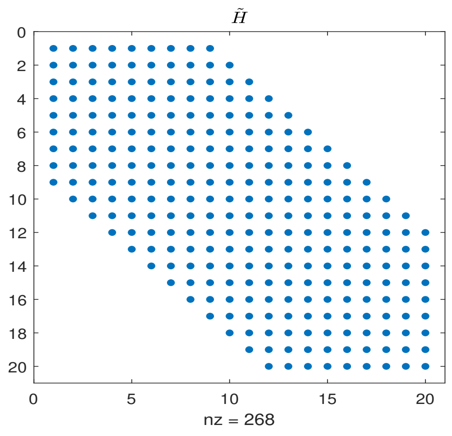
The number of decision variables of QP (4.3) is Nn_u; whereas it is $N(n_x + n_u)$ in the original QP (4.1). If $n_u << n_x$, the reduction in number of decision variables can be quite substantial. It is worth highlighting that this size reduction also decreases the computational time of the algorithm.

4.2 Null basis to solve KKT system

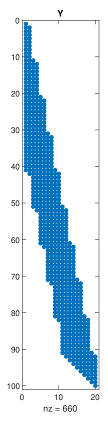
Null basis is not unique. A banded null basis as well as a null dense basis can be calculated. In sparse QP formulation of MPC, the equality constant matrix (F) has a banded structure. For a banded matrix, there may exist a banded null basis for its null space.



(a) Equality constraints matrix



(b) $\tilde{H}$ is the new Hessian matrix



(c) Null space

Figure 4.1: Example of banded matrices

If Y is a banded matrix, the matrices $\tilde{H}, \tilde{G}$ and H_0 will have a banded pattern. For instance, in Fig. 4.1, a new banded QP problem is obtained defined with banded matrices.

In order to construct a banded null basis Y for the null space of F, **Turnback algorithm** is used by [Berry et al., 1985].

In case of a banded null basis not being necessary, a dense null basis (MATLAB command: $Y = null(F)$) would be used which results in dense matrices in the new QP (4.3).

With respect to QP solver, it can be evaluated with splitting algorithms as described in the previous section, AMA, ADMM and their variants. Thus, slack variable $\nu \geq 0$ is introduced in Eq.(4.3) giving $\tilde{G}y + \nu = \tilde{g}$.

4.2.1 Turnback algorithm

A banded null basis based on an algorithm from structural engineering community called Turnback Algorithm was developed originally by [Topcu, 1979]. The banded null basis is used to obtain a smaller size QP than the original sparse QP while maintaining banded structures of matrices.

First-order algorithms can be employed for exploiting banded matrices to solve this new QP. In case of polytopic constraints, there are several existing ADMM and AMA variants using slack variables to deal with. The applicability of the proposed algorithm depends on whether or not Turnback algorithm can construct a banded null basis.

The interpretation of the turnback method as a matrix factorization was given in [Kaneko et al., 1982]. [Gilbert and Heath, 1987] and [Berry et al., 1985], in later extensions, confirm that turnback algorithms are the most effective algorithms for constructing a banded null basis.

Algorithm 8 Turnback Algorithm [Dang et al., 2017]

Given $F \in \mathbb{R}^{m\times n}(m < n)$, a banded null basis $Y \in \mathbb{R}^{n\times r}(r = n - m)$ can be constructed where $FY = 0_{m\times r}$.
Denote F_i, $i = 1, 2, \ldots, n$, the $i - th$ column. S a subset of $\{1, 2, \ldots, n\}$, F_S is a set of columns of F which indices are in S, $Y_i(S)$ are elements in $i - th$ column of Y which indices are in S. Thus, the main 3 steps are as follows:

1. **Column dependency identification step**
   Find active indices $c_1, c_2, \ldots, c_r$ so that the set of columns $\{F_1, F_2, \ldots, F_{c_k}\} \setminus \{F_{c_1}, \ldots, F_{c_k-1}\}$ are linearly dependent ($k = 1, 2, \ldots, r$ is increased to find a corresponding c_k. $\{F_{c_1}, \ldots, F_{c_k-1}\} = \emptyset$ if $k = 1$).

2. **Turn-back step**

   (a) $k = 1$, $\Gamma = \emptyset$

   (b) Set column F_{c_k} as an active column

   (c) Find $t_k = max\{j | j < c_k \;\;\&\;\; F_{S_k} \triangleq \{F_{c_k}, F_{c_{k-1}}, \ldots, F_j\} \setminus \{F_\Gamma\}$ are linearly dependent $\}(S_k = \{c_k, c_{k-1}, \ldots, j\} \setminus \Gamma$ is a set of indices in descending order). Find linear dependent coefficients vector b_k for F_{S_k} : $F_{S_k} b_k = 0$ and the last element of b_k is set to 1. Assign $Y_k(S_k) = b_k$.

   (d) $k = k + 1, \Gamma = \Gamma \cup \{t_k\}$. If $k \leq r$, return to set columns as actives columns (2.b), else quit and return Y.

Basically, as is shown in Algorithm 8, this algorithm has three phases. In the first phase, it searches for a set of active columns c_k, where an active column is a column such that columns starting from 1 to c_k, excluding previously found active columns, are linearly dependent.

In the second one, the turnback phase, starting from an active column c_k, the search is conducted in the reserved direction to find a turnback column t_k. And the third phase, the banded null basis is constructed.

4.3 Compute a particular solution z_0

Recall that $z = (u_0, x_1, u_1, x_2, \dots, u_{N-1}, x_N)$, and the constraints $Fz = f$, $f = [Ax_0; 0; \dots; 0]$ (x_0 is given), is the compact form of the following constraints:

$$\begin{aligned} x_1 &= Ax_0 + Bu_0, \\ x_2 &= Ax_1 + Bu_1, \\ &\vdots \\ x_N &= Ax_{N-1} + Bu_{N-1}. \end{aligned} \tag{4.4}$$

Then a particular solution of $Fz = f$ is any z which elements satisfy the above constraints. One such particular set is:

$$\begin{gathered} u_0 = u_1 = \dots = u_{N-1} = 0 \\ x_1 = Ax_0, x_2 = Ax_1, \dots, x_N = Ax_{N-1}. \end{gathered} \tag{4.5}$$

4.4 Banded and Dense Null Basis ADMM

Algorithm 9 B-ADMM

```
I. Offline: $Y$ (by **turnback**), $\tilde{H}, \tilde{G}, H_0$
II. Online:
  2.1. Compute particular solution $z_0$
  2.2. $\omega^T = z_0^T H_0$, $\tilde{g} = g - Gz_0$
while $\|r_{k+1}\| \geq \epsilon_{primal}$ or $\|s_{k+1}\| \geq \epsilon_{dual}$ do
  $y_{k+1} = -(\tilde{H} + \rho\tilde{G}^T\tilde{G})^{-1}[\omega + \rho\tilde{G}^T(\nu_k + \tau_k - \tilde{g})]$
  $\nu_{k+1} = max\{0, -\tilde{G}y_{k+1} + \tilde{g} - \tau_k\}$
  $\tau_{k+1} = \tau_k + \tilde{G}y_{k+1} + \nu_{k+1} - \tilde{g}$
end while
Obtain: $y_{stop}$. Solution: $z = z_0 + Yy_{stop}$
```

Algorithm 10 D-ADMM

```
I. Offline: $Y$(by **null(F)**), $\tilde{H}, \tilde{G}, H_0$
II. Online:
  2.1. Compute particular solution $z_0$
  2.2. $\omega^T = z_0^T H_0$, $\tilde{g} = g - Gz_0$
while $\|r_{k+1}\| \geq \epsilon_{primal}$ or $\|s_{k+1}\| \geq \epsilon_{dual}$ do
  $y_{k+1} = -(\tilde{H} + \rho\tilde{G}^T\tilde{G})^{-1}[\omega + \rho\tilde{G}^T(\nu_k + \tau_k - \tilde{g})]$
  $\nu_{k+1} = max\{0, -\tilde{G}y_{k+1} + \tilde{g} - \tau_k\}$
  $\tau_{k+1} = \tau_k + \tilde{G}y_{k+1} + \nu_{k+1} - \tilde{g}$
end while
Obtain: $y_{stop}$. Solution: $z = z_0 + Yy_{stop}$
```

Being the primal residual

$$r_{k+1} = \tilde{G}y_{k+1} + \nu_{k+1} - \tilde{g} \tag{4.6}$$

And, the dual residual

$$s_{k+1} = \rho \tilde{G}^T (\nu_{k+1} - \nu_k) \tag{4.7}$$

In both cases, residuals are equal due to the fact that the only modification is in the construction of the null basis Y.

Step-size selection

According to [Dang et al., 2017], the step-size ρ for this ADMM formulation can be chosen heuristically based on [Ghadimi et al., 2015]:

$$\rho = \left(\sqrt{\lambda_{min,>0}(\tilde{G}\tilde{H}^{-1}\tilde{G}^T)\lambda_{max,>0}(\tilde{G}\tilde{H}^{-1}\tilde{G}^T)} \right)^{-1} \tag{4.8}$$

4.5 Banded and Dense Null Basis AMA

Algorithm 11 B-AMA

```
I. Offline: $Y$ (by **turnback**), $\tilde{H}, \tilde{G}, H_0$
II. Online:
   2.1. Compute particular solution $z_0$
   2.2. $\omega^T = z_0^T H_0$, $\tilde{g} = g - Gz_0$
while $\|r_{k+1}\| \geq \epsilon_{primal}$ or $\|s_{k+1}\| \geq \epsilon_{dual}$ do
   $y_{k+1} = -\tilde{H}^{-1}[\omega + \rho\tilde{G}^T(\tau_k)]$
   $\nu_{k+1} = max\{0, -\tilde{G}y_{k+1} + \tilde{g} - \tau_k\}$
   $\tau_{k+1} = \tau_k + \tilde{G}y_{k+1} + \nu_{k+1} - \tilde{g}$
end while
Obtain: $y_{stop}$. Solution: $z = z_0 + Yy_{stop}$
```

Algorithm 12 D-AMA

```
I. Offline: $Y$(by **null(F)**), $\tilde{H}, \tilde{G}, H_0$
II. Online:
   2.1. Compute particular solution $z_0$
   2.2. $\omega^T = z_0^T H_0$, $\tilde{g} = g - Gz_0$
while $\|r_{k+1}\| \geq \epsilon_{primal}$ or $\|s_{k+1}\| \geq \epsilon_{dual}$ do
   $y_{k+1} = -\tilde{H}^{-1}[\omega + \rho\tilde{G}^T(\tau_k)]$
   $\nu_{k+1} = max\{0, -\tilde{G}y_{k+1} + \tilde{g} - \tau_k\}$
   $\tau_{k+1} = \tau_k + \tilde{G}y_{k+1} + \nu_{k+1} - \tilde{g}$
end while
Obtain: $y_{stop}$. Solution: $z = z_0 + Yy_{stop}$
```

Step-size selection

Considering that the dual problem is of (4.3), the dual function is defined as

$$D(\nu) = -f^*(-\tilde{G}^T\nu) - g^*(-\nu) - \nu^T\tilde{g} \tag{4.9}$$

where ν is the Lagrangian parameter. $f(y)$ is a quadratic and $g(\nu)$ is an indicator function.

Taking into account that the conjugate of a quadratic for a strictly convex matrix is

$$f^*(y) = \frac{1}{2}(y-q)^T A^{-1}(y-q) - d \tag{4.10}$$

assuming the quadratic function $f(x) = \frac{1}{2}\|x\|_Q + q^T x + d$ with $Q \in S_{++}$ where S_{++} is a set of symmetric positive definite real matrices; and the conjugate of the Indicator on the Non-Negative Orthant is $g^*(-\nu) = g(\nu) = I_+(\nu)$. The dual function can be rewritten as

$$D(\nu) = \underbrace{-\frac{1}{2}(\tilde{G}^T\nu + h)^T \tilde{H}^{-1}(\tilde{G}^T\nu + h)}_{F(-\nu)} - I_+(\nu) - \nu^T \tilde{g} \tag{4.11}$$

which gives $L(\nabla F(-\nu)) = L(\nabla F(\nu)) = \lambda_{max}(\tilde{G}\tilde{H}^{-1}\tilde{G}^T)$. Thus, the step size condition takes the form

$$0 < \rho < \frac{2}{L(\nabla F(\nu))} = \frac{2}{\lambda_{max}(\tilde{G}\tilde{H}^{-1}\tilde{G}^T)} \tag{4.12}$$

Chapter 5

Computational Analysis

The purpose of this chapter is to carry out a computational study of these algorithms applied to MPC in such a way that allows to draw conclusions about the effects of MPC parameters on the different formulations, so that clear criteria about the most convenient formulation can be established for the implementation of embedded controllers in the most efficient way.

5.1 Scalability

In order to study how varying the size of matrices due to the prediction horizon affects the convergence rate and execution time of each algorithm when solving at each sampling step a QP formulated using first order algorithms as it is mentioned in previous chapters.

To aim this, 500 synthetic random MPC problems are generated. Particularly, 100 MPC problem per prediction horizon. Here, QPs are obtained arising from each random MPC.

These random MPC problems are created by generating 500 random sets of matrices A, B, Gx, Gu, Gn= Gx with parameters Q=I and R=I. The size of these matrices will be defined taking into account 8 states and 3 inputs. Tests are carried out on a PC to obtain the convergence property, in terms of computational time and the number of iterations of each algorithm.

To ensure that the generated QPs are feasible, a random initial state (x_0) is generated and a particular solution for the equality constraints is obtained ($Fz_0 = f$). Then, by substituting z_0 to the inequality constraints, g_x, g_N and g_u is chosen such that the inequality is feasible. In addition, prediction horizon evaluated along the simulations are $N = [5, 10, 15, 20, 25]$.

It is worth highlighting that the total time until the convergence of the algorithm in each simulation is measured in the main loop of the algorithm without considering previous calculations as step-size or Cholesky factorization of the left-side of the linear system.

Appendix A contains all the results obtained in this experiment. Figure 5.1 shows the average time per iteration obtained for all the algorithms under study in the 100 random MPC problems for each prediction horizon.

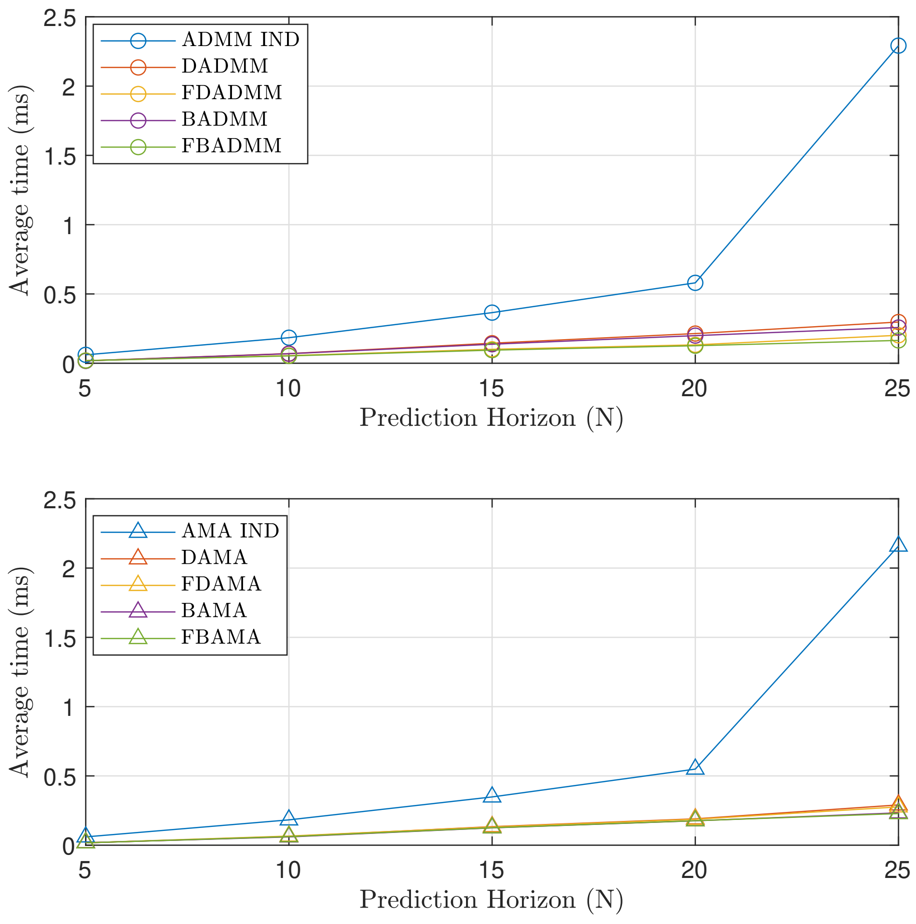


Figure 5.1: Graph of the average time per iteration for 100 random MPC problems

As it is shown, both in AMA variants and ADMM variants, the Indirect Indicator formulations are slower than the formulations by reduced-Hessian. This makes sense since as the prediction horizon increases, the size of the matrices in the QP problem are larger and it takes more time for finding a solution. It occurs since a dense formulation (considers only inputs) has less decision variables than a sparse form (consider state and inputs) which it is characteristic when a problem is solved with a second-order optimization algorithm.

In addition, this increment of the problem is substantial if the number of states is greater. On the contrary, in the case of using reduced-Hessian approach where the original problem is reduced and the states are not considered, this solution will take less time.

To visualise better the performance of the algorithms in each prediction horizon, the graph in the Figure 5.1 is converted in a bar plot (Figure 5.2). In this figure, it is

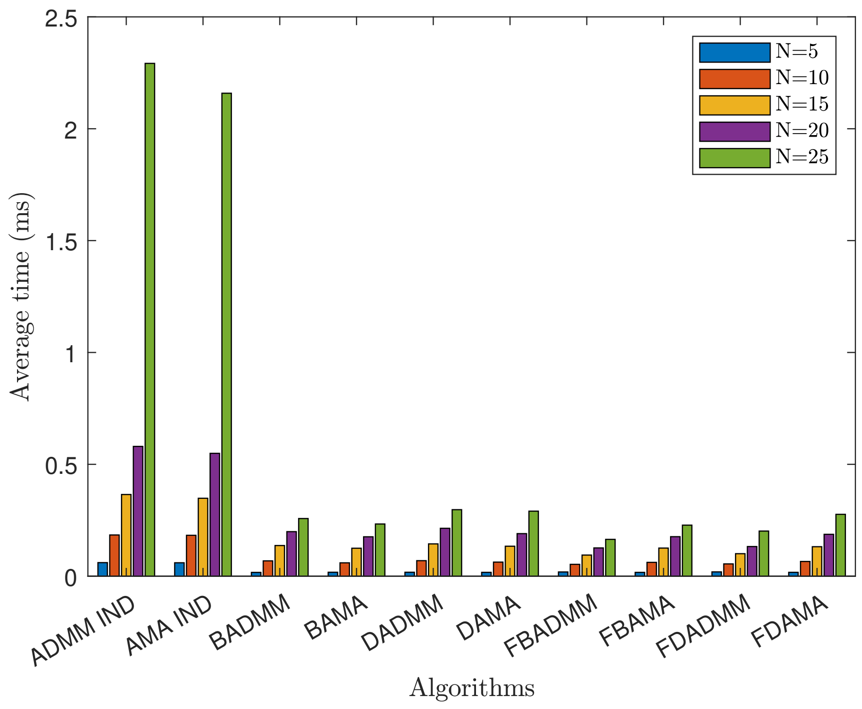


Figure 5.2: Bar plot for the average time per iteration for 100 random MPC problems

proven the average time taken by the indirect formulation is bigger than the others. However, once demonstrates that the reduced-Hessian approach takes less time, it is convenient to analyse these approaches in detail.

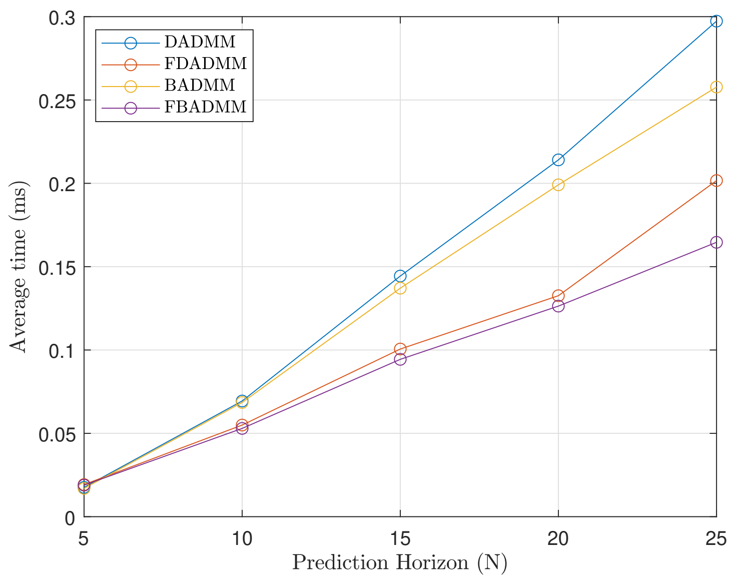


Figure 5.3: Average time per iteration for ADMM variants using the reduced-Hessian method

Figure 5.3 shows the average time per iteration for the ADMM algorithms using the reduced-Hessian method. Here, it is noted that using a dense null basis the time is not the best whereas using a banded null space, the average time is reduced. This reduction is more visual as the prediction horizon increases. Furthermore, with respect to the accelerated variants, these have the best average time in the comparison and the banded approach maintains its trend with respect to the corresponding basic algorithm. It is the same for the dense approach which time is better than the DADMM and BADMM but worse than the accelerated variant for the banded ADMM.

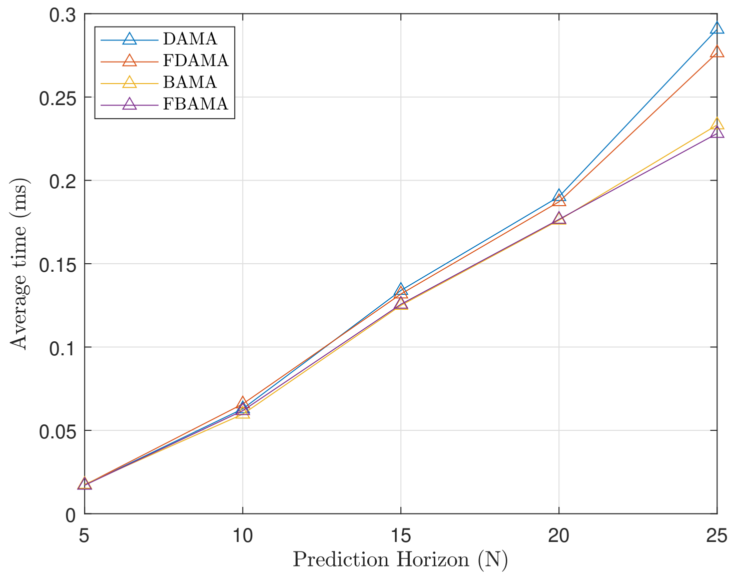


Figure 5.4: Average time per iteration for AMA variants using the reduced-Hessian method

With respect to the AMA algorithms using the reduced-Hessian approach, Figure 5.4 is presented. According to this graph, the average time for banded AMA is also better than the dense variant and the difference between them increases as the prediction horizon is bigger. In addition, accelerated variants are slightly better than the basic algorithms.

Once described both algorithms separately, it is necessary to evaluate a comparison which includes all the algorithms using reduced-Hessian. Therefore, Figure 5.5 is presented where the accelerated variants are not considered due to the fact that they always tend to improve the convergence speed and analysing this acceleration is the same than evaluate original algorithms. For this reason, only the performance of the basic algorithms will be shown.

Figure 5.5 shows that the shortest average time in the comparison is the banded approach for the AMA algorithm. This makes sense since it is the algorithm with the minimum number of operations to obtain a solution due to the banded matrices and simple updates.

After BAMA, Dense AMA and Banded ADMM are the shortest average times. During the first four prediction horizon, AMA algorithm is more stable; however, in the last prediction horizon, the AMA variant increases considerably unlike the ADMM

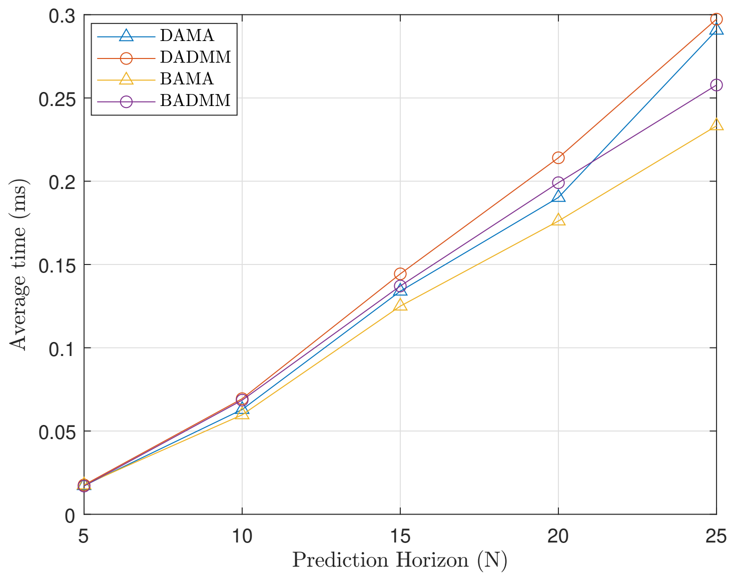


Figure 5.5: Average time per iteration for all the algorithms using the reduced-Hessian method

variant. Thus, considering all the horizons, it is said that the banded ADMM is more stable throughout the horizons.

With respect to the time of Dense ADMM, it increases in each horizon; thus, it is considered the worst in the graph. Despite that, the difference between the longest and shortest average time is not huge.

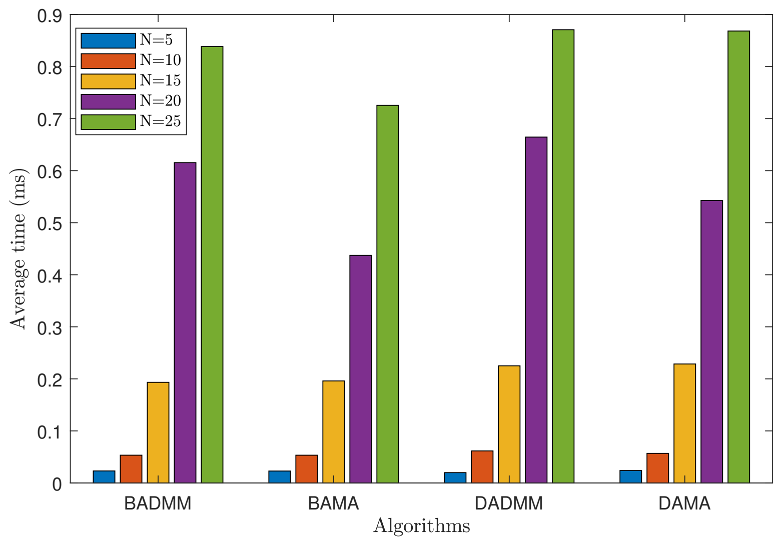


Figure 5.6: Bar plot of average time per iteration for all the algorithms using the reduced-Hessian method

On the same way, a bar plot in Figure 5.6 is shown to consider a better visualization of the information obtained. As it is shown, in the first horizon, all responses are practically the same; thus, in case of having a problem with this prediction horizon, probably it is not necessary to distinguish between variants such as banded or dense formulation using reduced-Hessian approach.

In the second horizon, AMA variants has a slightly better behaviour with respect to the ADMM algorithms. Here, the differences between algorithms could start appearing because the size of the constructed matrices increases according to the horizon selected.

In the third and fourth horizon, the differences are really noted and it is highlighted the average time of the banded AMA approach which has been the shortest.

Finally, in the last prediction horizon, as it is shown in Figure 5.6, banded AMA results with the best performance, following by the banded approach for the ADMM algorithm.

Once analysed the results in term of average time per iteration for all the algorithms, it is necessary to take into account the total time of them. As time total is referred to the multiplication of the time taken for the algorithm until the convergence and the number of iteration taken.

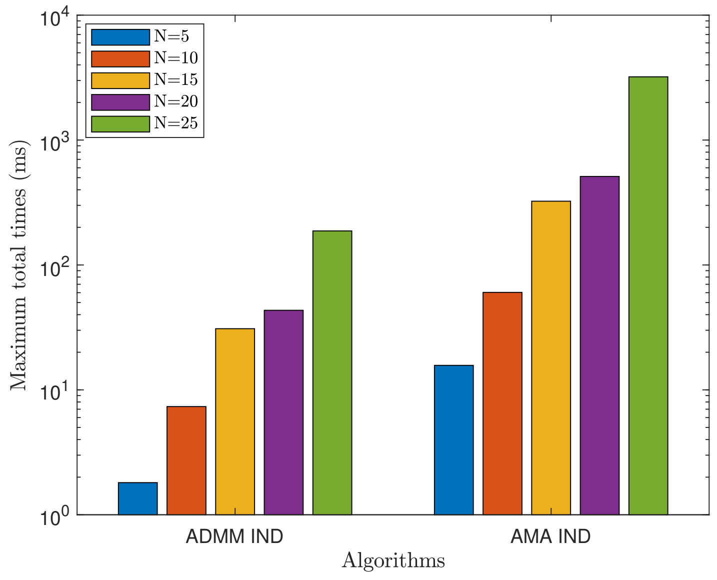


Figure 5.7: Bar plot of maximum total times for Indirect formulations

In Figure 5.7, the maximum total time for Indirect formulations are shown. In this case, considering all the problems, the average time per iteration shown in Figure 5.6 is multiplied for the maximum number of iterations obtained. On this way, it is shown that the ADMM is quite better than the AMA variant. It makes sense since the step-size of the Indirect ADMM is considered as optimal unlike the Indirect AMA its step-size is selected following a general condition.

With respect to the reduced-Hessian variants, in Figure 5.8, the maximum total time for all the algorithms is shown. It is obtained in the same way than the indirect formulations by multiplying the average time per iteration described in Figure 5.6

and the maximum iterations obtained for each algorithm according to the prediction horizon.

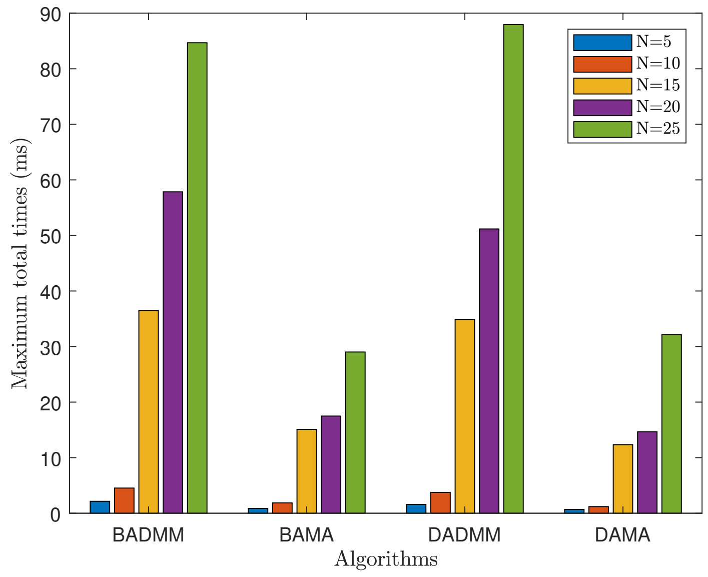


Figure 5.8: Bar plot of maximum total time for formulations using reduced-Hessian method

As it is presented in Figure 5.8, Banded variants are better than the Dense variants both in ADMM and AMA. The largest differences between algorithms are obtained as the prediction horizon is increased. Thus, in the prediction horizon equal to 25, it is proven that the banded variants consume less total time than dense variants, and the AMA algorithms use a shorter time than ADMM methods.

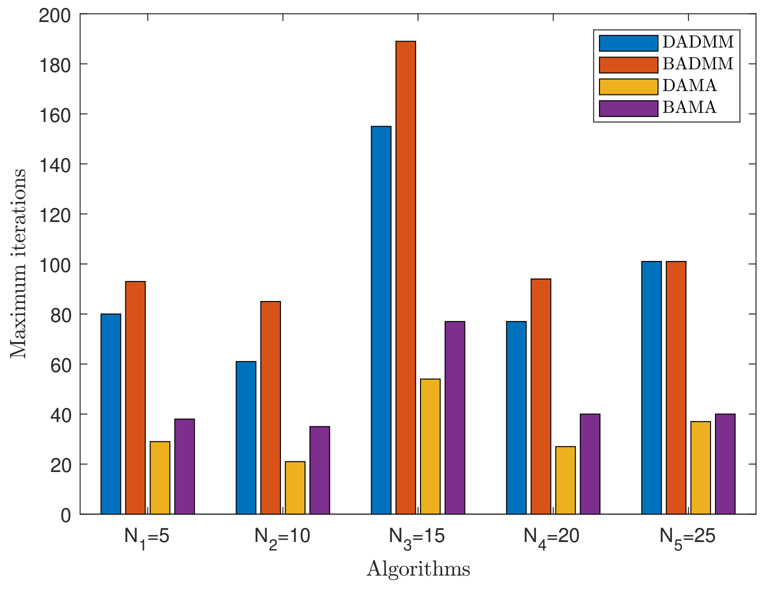


Figure 5.9: Bar plot of maximum iterations for formulations using reduced-Hessian method

In Figure 5.9, the data for the maximum iterations is presented. Here, the analysis is according to the performance of all the algorithms in each prediction horizon since the same problems for each algorithm tested in each horizon are solved. Thus, this Figure shows the comparison of these maximum iterations for all the algorithms solving the same problems.

In this Figure 5.9, the AMA variants requires less iterations than the ADMM variants. This makes sense since the AMA algorithm is easier to solve than the ADMM method due to the first update (linear solver) of their approach as first-order algorithms.

In addition, the banded approach demonstrates a smooth increase in its iterations unlike the dense approach. This performance could occur due to the construction of the new problem which only differs in the null space.

In summary, both algorithms with banded reduced-Hessian approach applied have a good performance throughout prediction horizons evaluated. Therefore, in order to obtain more information about their performance, they will be tested in the next section, with two reference second-order algorithms.

5.1.1 Data dispersion

The measurements of central tendency are not always adequate to describe data. Thus, to describe data, it is necessary to know the extent of variability. This is given by the measures of dispersion. Range, interquartile range, and standard deviation are the three commonly used measures of dispersion.

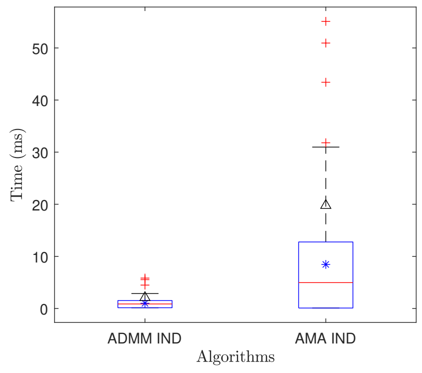
In this case, a high dispersion means that the time or number of iterations of a specific algorithm changes easily and this can get worse its performance. Thus, suffering minimal variations in its initial conditions, this algorithm can increase substantially its iterations, and its time until convergence, unlike others. Therefore, the purpose of this study is to identify these algorithms with high dispersion.

In this project, standard deviation of the data will be used to analyse this dispersion. Therefore, the following tables will describe the standard deviation for all the resulting data obtained in this experiment for each prediction horizon. Also, a box plot will be presented to reinforce the standard deviation results.

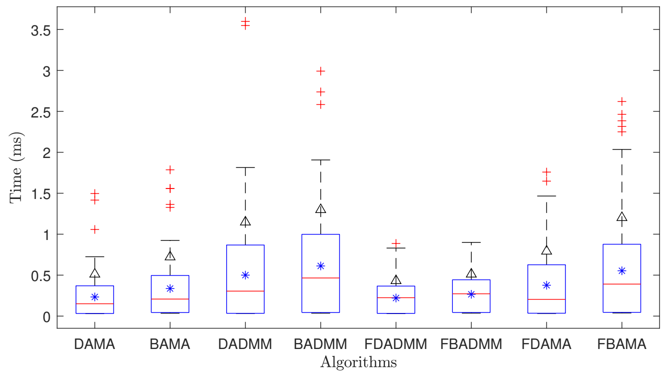
Prediction horizon $N = 5$

Table 5.1: Standard deviation for all the algorithms in $N = 5$

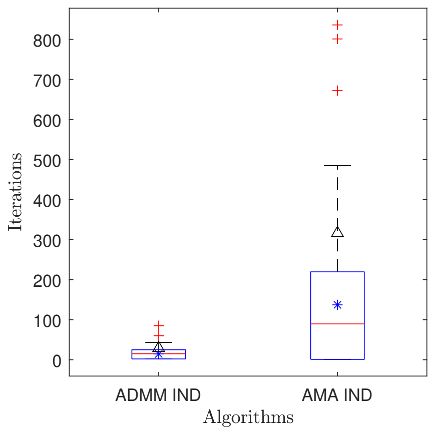
$N = 5$										
Algorithms	**AMA IND**	**ADMM IND**	**DAMA**	**DADMM**	**BAMA**	**BADMM**	**FDAMA**	**FDADMM**	**FBAMA**	**FBADMM**
Times (ms)	11.3157	1.0826	0.2774	0.6471	0.3851	0.6862	0.4148	0.2107	0.6481	0.2452
Iterations	179.0662	14.4881	16.4043	33.5194	22.2644	42.0923	25.0738	11.1554	36.0941	13.8858



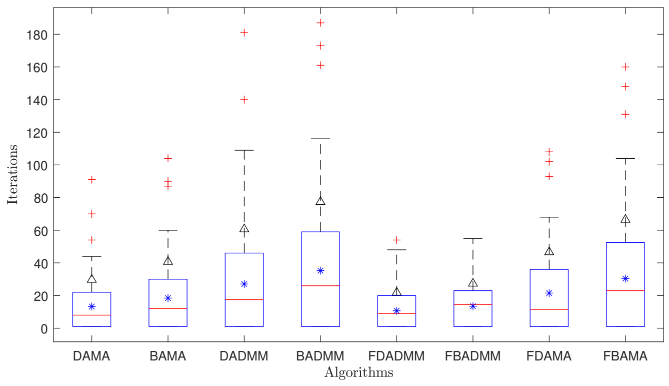
(a) Times in ms for Indirect variants



(b) Times in ms for reduced-Hessian variants



(c) Iterations for Indirect variants



(d) Iterations for reduced-Hessian variants

Figure 5.10: Box plot of resulting data in $N = 5$. $(*)$ is the mean and (Δ) is the standard deviation.

Table 5.2: Standard deviation for all the algorithms in $N = 10$

$N = 10$										
Algorithms	**AMA IND**	**ADMM IND**	**DAMA**	**DADMM**	**BAMA**	**BADMM**	**FDAMA**	**FDADMM**	**FBAMA**	**FBADMM**
Times (ms)	33.5631	2.8299	1.0487	2.3677	1.4772	2.9435	1.7787	0.5934	2.5503	0.7352
Iterations	172.2242	13.7054	15.6593	31.5848	21.8968	40.0529	24.9273	10.3625	37.6894	13.1985

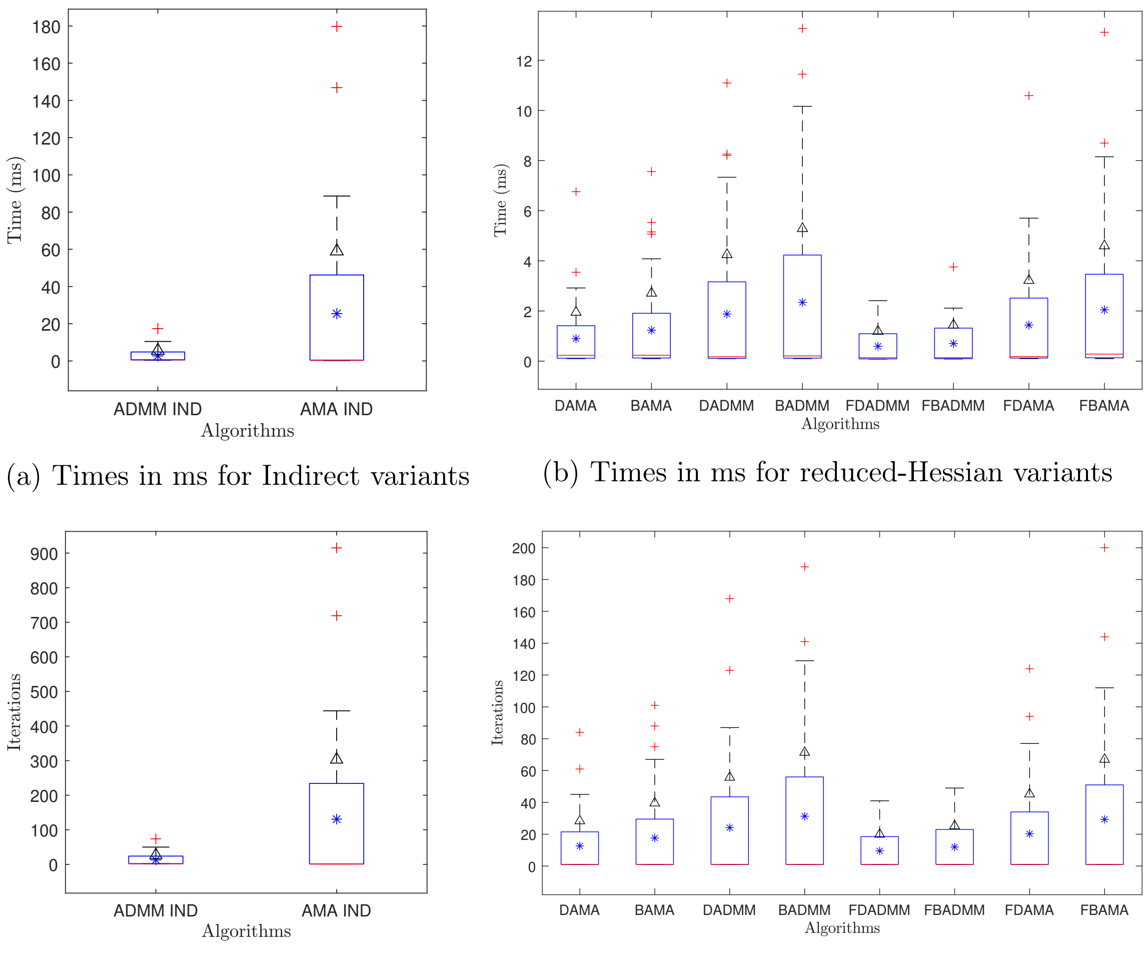
Prediction horizon $N = 10$

Prediction horizon $N = 15$

Table 5.3: Standard deviation for all the algorithms in $N = 15$

$N = 15$										
Algorithms	**AMA IND**	**ADMM IND**	**DAMA**	**DADMM**	**BAMA**	**BADMM**	**FDAMA**	**FDADMM**	**FBAMA**	**FBADMM**
Times (ms)	67.7019	5.7446	2.3018	5.1300	3.2370	7.0110	3.8297	1.1383	5.6175	1.3747
Iterations	189.9563	15.3225	17.3202	35.7266	25.0657	48.7116	27.3023	11.1957	44.0504	15.2457

Prediction horizon $N = 20$



(a) Times in ms for Indirect variants

(b) Times in ms for reduced-Hessian variants

(c) Iterations for Indirect variants

(d) Iterations for reduced-Hessian variants

Figure 5.11: Box plot of resulting data in $N = 10$. $(*)$ is the mean and (Δ) is the standard deviation.

Table 5.4: Standard deviation for all the algorithms in $N = 20$

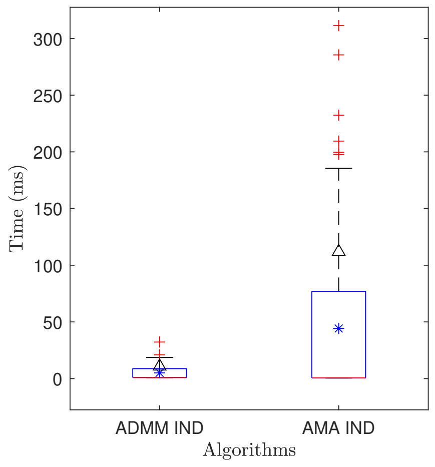
$N = 20$										
Algorithms	**AMA IND**	**ADMM IND**	**DAMA**	**DADMM**	**BAMA**	**BADMM**	**FDAMA**	**FDADMM**	**FBAMA**	**FBADMM**
Times (ms)	99.8839	9.8187	2.9222	7.8764	3.7318	9.7715	4.5877	1.6582	6.6372	1.9974
Iterations	172.1373	15.4262	15.0622	35.7777	21.7885	47.1399	24.2222	11.6742	37.0207	15.4829

Prediction horizon $N = 25$

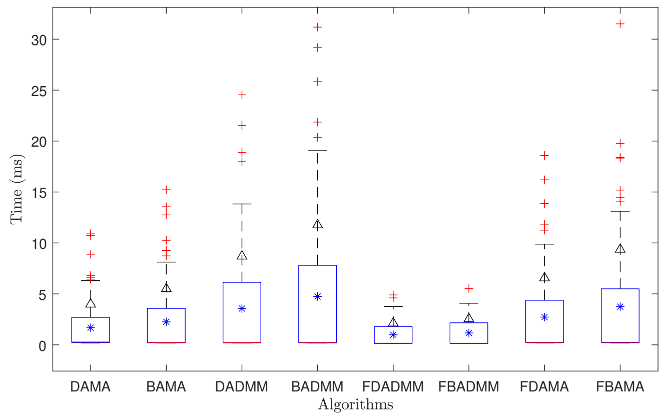
Table 5.5: Standard deviation for all the algorithms in $N = 25$

$N = 25$										
Algorithms	**AMA IND**	**ADMM IND**	**DAMA**	**DADMM**	**BAMA**	**BADMM**	**FDAMA**	**FDADMM**	**FBAMA**	**FBADMM**
Times (ms)	389.7153	106.2469	4.3675	28.3437	5.6840	26.0245	6.3188	2.9678	8.4758	2.9178
Iterations	163.1267	42.2142	14.7999	86.8310	20.6462	89.5890	22.7808	15.2542	33.5062	17.4530

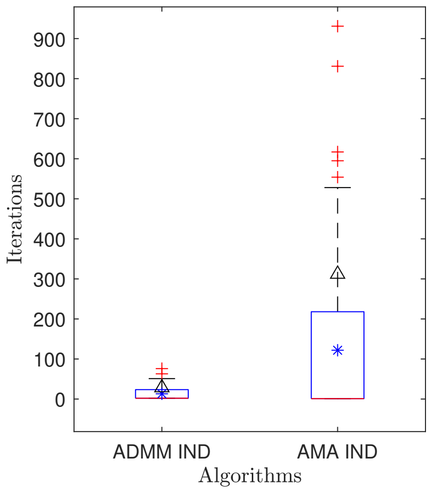
Taking into account all the data dispersion shown in the box plots (Figure 5.10-5.14) and Tables 5.1-5.5, it is noted that Indirect AMA has the bigger dispersion of all the algorithms throughout the prediction horizons. As it is mentioned previously, this could occur due to the step-size selection for this algorithm which was chosen based on the general restriction for AMA which involves the convexity modulus and



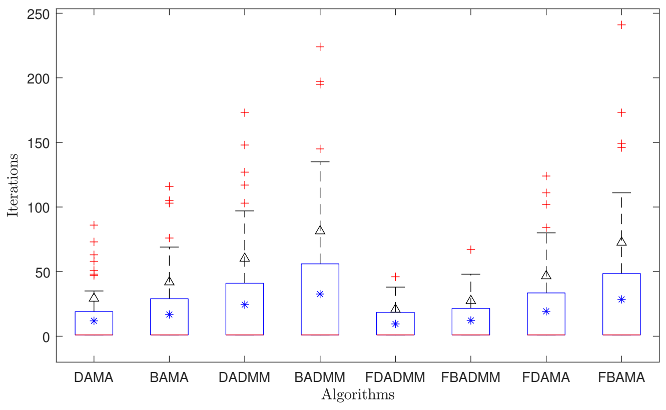
(a) Times in ms for Indirect variants



(b) Times in ms for reduced-Hessian variants



(c) Iterations for Indirect variants



(d) Iterations for reduced-Hessian variants

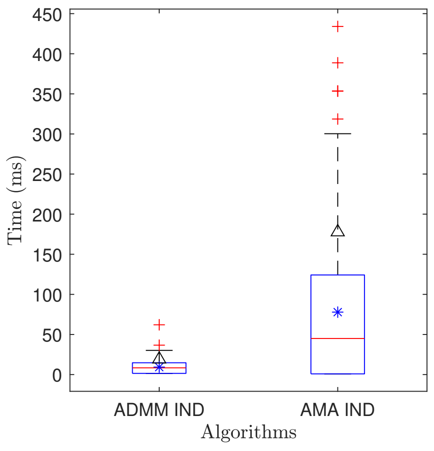
Figure 5.12: Box plot of resulting data in $N = 15$. $(*)$ is the mean and (Δ) is the standard deviation.

the eigenvalues of the equality constraints matrix.

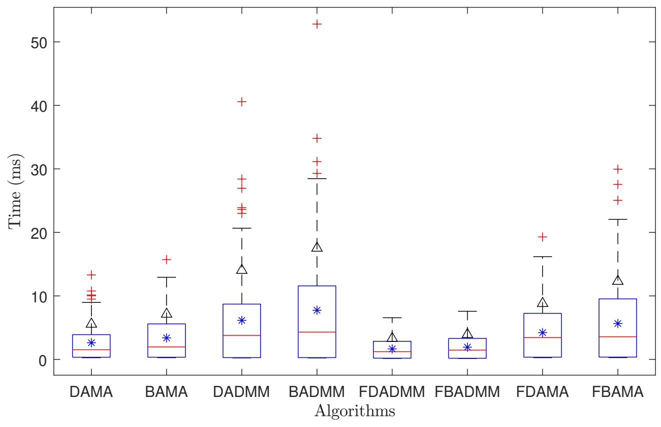
Moreover, the ADMM variants has lower dispersion unlike the AMA variants. This means that the stability in terms of time and iterations is more robust than the rest of the other approach. This could be due to the fact that the step-size selection for the ADMM variants are considered as optimal whereas there are AMA variants which step-sizes are not optimized (e.g Indirect AMA).

5.2 Comparison with second-order algorithm

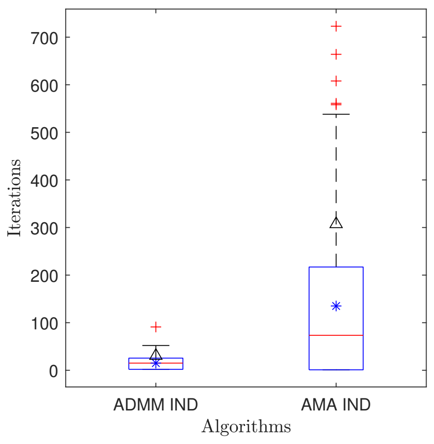
In order to analyse the behaviour of the best first-order algorithms resulting of the previous study, it is necessary to compare them with a second-order algorithm. For this work, the simplified Interior Point (IP) developed in [del Rio Ruiz and Basterretxea, 2019]



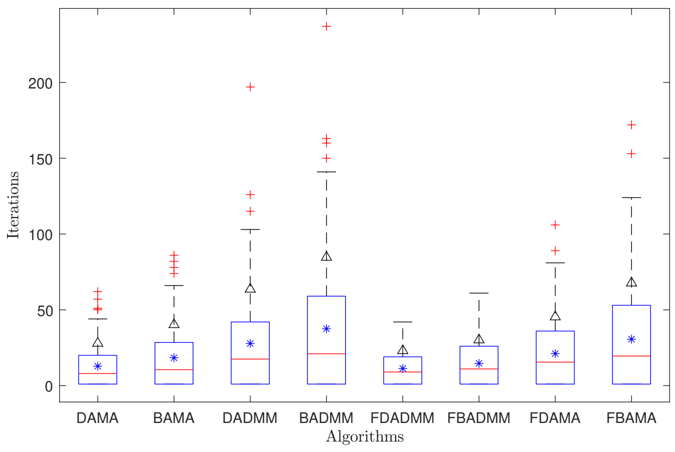
(a) Times in ms for Indirect variants



(b) Times in ms for reduced-Hessian variants



(c) Iterations for Indirect variants



(d) Iterations for reduced-Hessian variants

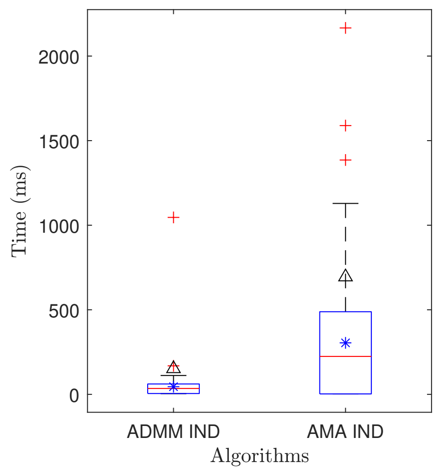
Figure 5.13: Box plot of resulting data in $N = 20$. $(*)$ is the mean and (Δ) is the standard deviation.

is used as second-order method. In addition, the two first-order algorithms considered for this study are the banded AMA and banded ADMM.

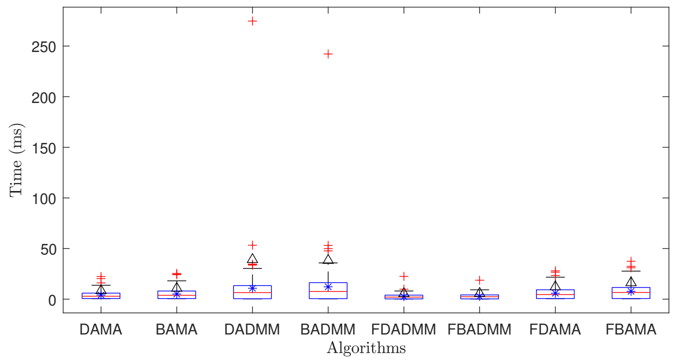
These algorithms are tested with a model considered as a benchmark in [Liu et al., 2014], [Jerez et al., 2011] and many recent works use it to validate their hardware controller. This problem has been selected since it has the advantage to be easily modified and this allows to establish constraints in the state and input variables obtaining hard or soft problems.

In addition, this benchmark allows to vary the size of the resulting QP problem by modifying the number of masses and inputs, meanwhile the control objective can be easily changed by varying the configuration of the system (e.g. attached to walls in each end), the desired reference and the size of the QP problem.

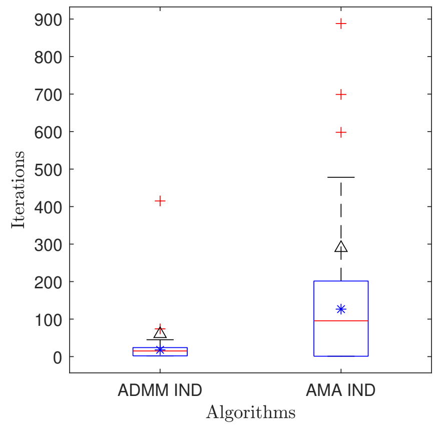
This benchmark example consisting of a set of oscillating masses attached to walls is considered. In this case, the system is sampled every 0.5 seconds assuming masses and spring constants with a value of 1kg and $1Nm^{-1}$, respectively. The system has



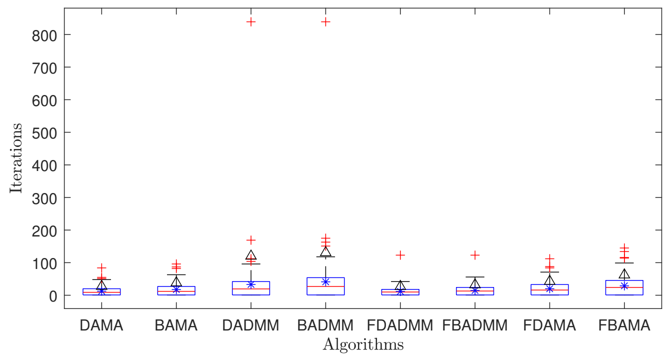
(a) Times in ms for Indirect variants



(b) Times in ms for reduced-Hessian variants



(c) Iterations for Indirect variants



(d) Iterations for reduced-Hessian variants

Figure 5.14: Box plot of resulting data in $N = 25$. $(*)$ is the mean and (Δ) is the standard deviation.

two control inputs, three masses and two states for each mass, its position and velocity, for a total of six states. The prediction horizon used is a range defined as $N = [5, 10, 15, 20, 25]$.

The objective of this controller is to track a reference for the position of each mass while satisfying the system limits beginning from an initial state. In this study, this initial state is randomly generated in order to evaluate how stable are the performance of these algorithms starting from different states. Thus, 100 random initial states are generated for each prediction horizon.

On the other hand, in order to analyse the performance of the algorithms when the constraints are different, two types of experiment will be tested when there are both soft and hard constraints. For soft constraints, $\|x_k\|_\infty \leq 3.5$ and $\|u_k\|_\infty \leq 0.5$ are defined. Whereas for hard constraints, $\|u_k\|_\infty \leq 0.1$ is selected.

The algorithms tested have the following configurations: For the simplified IP: $sigma = 0.1$, $gamma = 0.1$ and $mulimit = 0.1$. And the first-order algorithms have, as stopping conditions, a primal and dual error equal to $1e^{-4}$ with step-size defined in [Ghadimi et al., 2015] for ADMM and $\rho = 0.99 * 2/\lambda_{max}(GH^{-1}G^T)$ for AMA.

Therefore, taking into account the above-mentioned statements, results are obtained which consist on sets of histograms that identify the number of occurrences of each algorithm and their iterations according to the initial state generated randomly.

Both, soft and hard constraints are evaluated in order to identify the performance of the algorithm in term of time and number of iterations when the problem is stricter since using only soft constraints could offer unfair conclusions regarding second-order algorithms.

It is expected that the first-order algorithms increase substantially their total time due to the increase in the complexity of the problem, and that the simplified IP barely changes its number of iterations and therefore, its total time.

5.2.1 Soft constraints

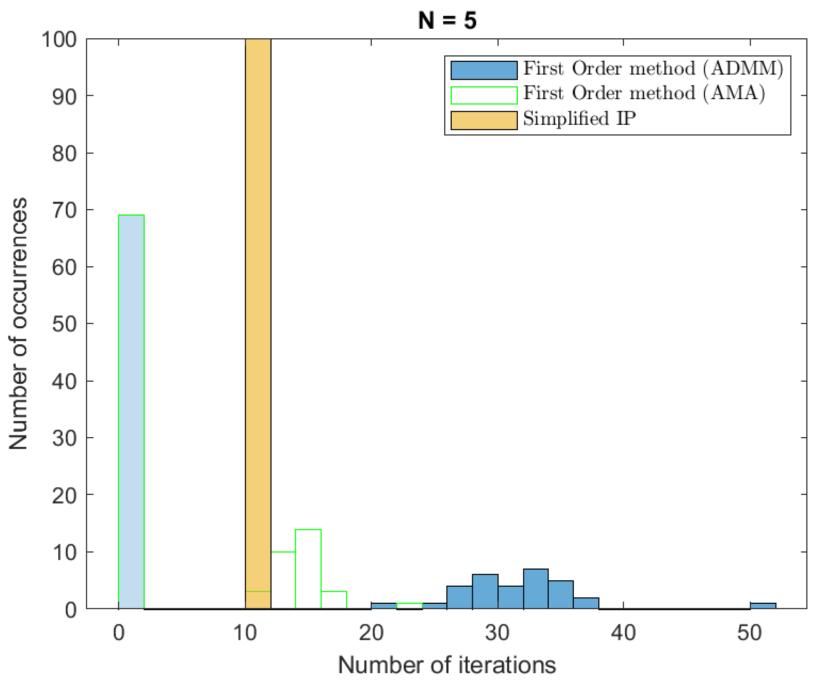


Figure 5.15: Resulting histograms for N=5

In Figure 5.15, results are shown for a prediction horizon equal to 5. According to these results, the first-order algorithms vary their iterations depending on the initial state. With respect to the ADMM variant, it takes more iterations than the AMA variant for the same random initial states and it has a range of iterations between 20 and 52.

In addition, with respect to banded AMA, its range of iterations are located between 10 and 24. Thus, it is proven that banded AMA takes less iterations than ADMM. This makes sense since the computational burden for the first update in both first-order algorithm is more complex in BADMM than BAMA.

It is worth highlighting that there is a number of occurrences for both first-order algorithms where the initial state makes that the problem is solved in 1 iteration. This occurs due to the random initial states corresponding to these occurrences are near to the stabilization and the algorithm only needs one iteration to found the optimum solution.

With regard to the second-order algorithm, this algorithm has the relevant characteristic that the range of iterations are the same for all the random initial states. Thus, it is proven the stability of the second-order algorithms unlike the first-order algorithms that increase their iterations when the initial state is not convenient to found the solution quickly.

Based on these results, it is expected that as the prediction horizon increases, the range of the number of iterations for the first-order algorithms will be wider. On the contrary, in the case of the second-order algorithms, it is expected that their range is maintained throughout the prediction horizons.

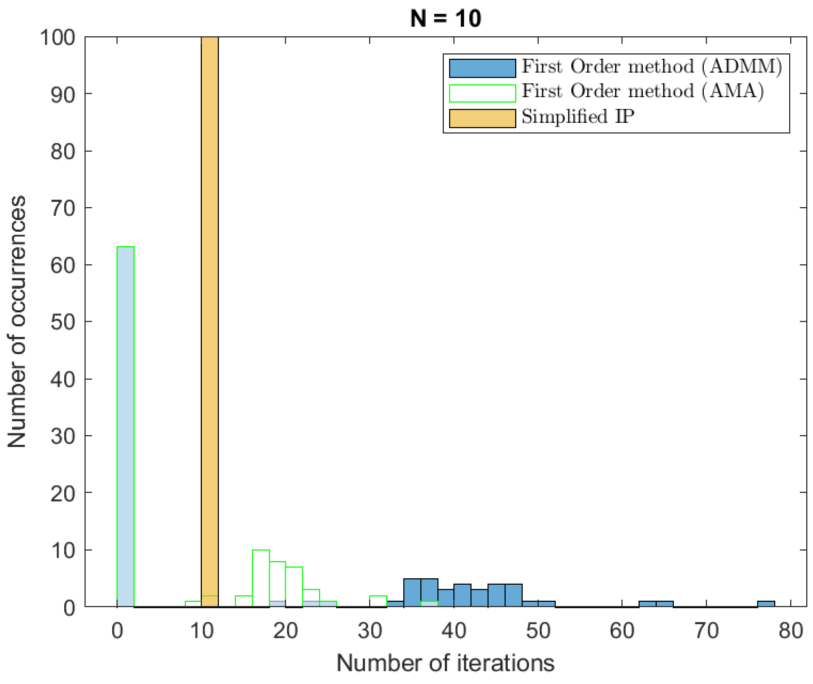


Figure 5.16: Resulting histograms for N=10

In the Figure 5.16, the pattern is the same for the prediction horizon equal to 10. First-order algorithms have a range wider being the ADMM the variant with more iterations whereas the simplified IP maintains its range. With respect to the simplified IP, it demonstrates more stability in iterations maintaining its range in all the random initial states.

On the same way, Figure 5.17 for the prediction horizon equal to 15 is presented. In this case, the ADMM variant requires more iterations that the rest of the algorithms. This makes sense since the complexity of the problem increases as the initial state is

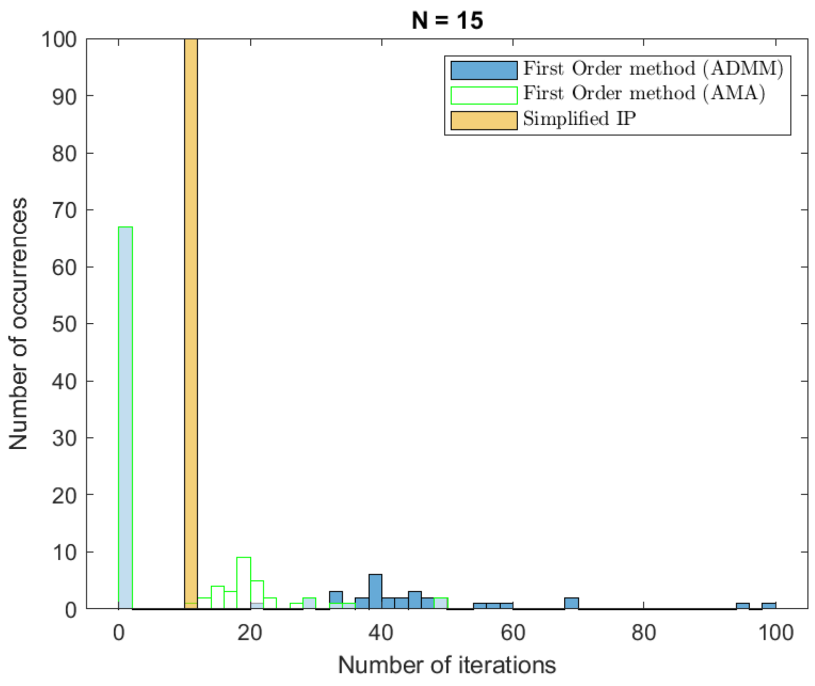


Figure 5.17: Resulting histograms for N=15

far of the solution and therefore, the calculation of the linear solver takes more time.

Furthermore, the banded AMA approach has its range of iterations between 12 and 58 iterations (without considering the random initial states that allow the problem to be solved in one iteration), and it has the best performance of the first-order algorithms.

The simplified IP maintains its range despite the fact that the prediction horizon increases for all the initial states. The same pattern is presented for *Quadprog* which also maintains its range.

For the prediction horizon equal to 20 and 25, Figure 5.18 and Figure 5.19 are shown, respectively. For $N = 20$, the pattern is the same as the previous horizons being the simplified IP more stable than the first-order algorithms. In addition, AMA variant continues being the best of the splitting methods.

With respect to the plot corresponding to $N = 25$, the iterations for banded ADMM are really wider than for the rest of the algorithms. This response is not the same for banded AMA which iterations are not increase considerably.

The iterations for the simplified IP continues being the same for this horizon and this probes that the random initial states do not affect its convergence.

With regard to the resulting histograms, it is noted that the warm starting is substantial in the first-order algorithms since they also depend on the initial state selected to improve the convergence speed. Warm starting refers to consider an initial state close to the solution; on the contrary, it is named a Cold starting. However, it does not occur with the second-order algorithm where the number of iterations are exactly

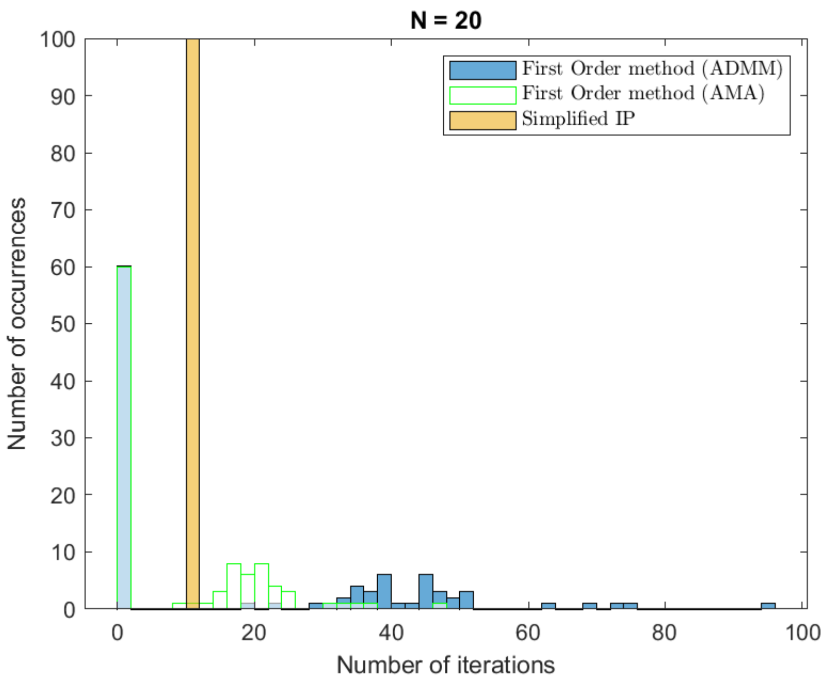


Figure 5.18: Resulting histograms for N=20

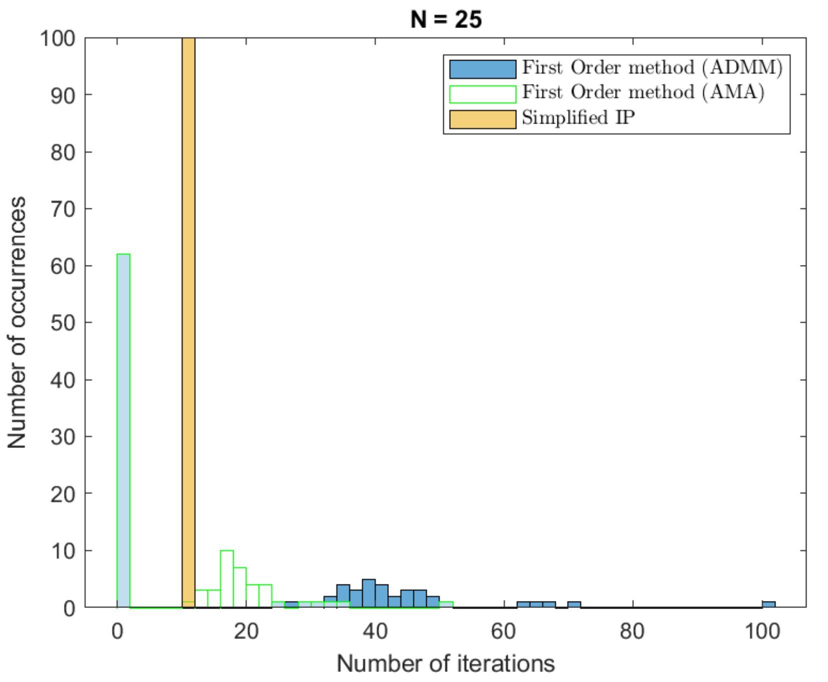


Figure 5.19: Resulting histograms for N=25

the same for any initial state and any prediction horizon.

On the other hand, to obtain more information about the behaviour of these two types of algorithms, total times must be presented in each prediction horizon both first and second-order methods.

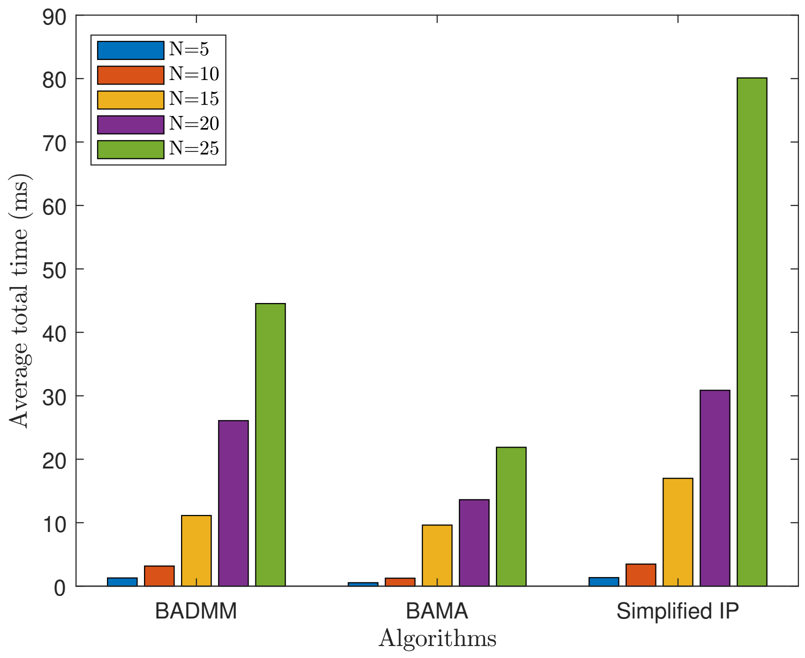


Figure 5.20: Average total time in milliseconds for each prediction horizon in both first and second-order algorithms following soft constraints

In Figure 5.20, the average total time for all the algorithms tested is shown. These values were calculated by multiplying the average time per iteration of each algorithm and the maximum number of iterations considering the simulations performed. In this Figure, the banded AMA has the shortest average total time of the comparison; however, to define a fair approach in terms of total time, it is necessary to consider both the maximum number of iterations and the maximum time taken until the convergence in each algorithm since it is the maximum time to converge what will condition the maximum sampling rate to be applied to the control system.

In Figure 5.21, it is noted that the simplified IP has a more stable maximum total time than the first-order methods. This makes sense since the number of iterations in the first-order methods increase considerably when the complexity of the problem is bigger. This does not occur with the second-order algorithms since their iterations barely change even though the initial state is randomly generated.

Therefore, when the constraints will be harder, it is expected that the iterations of the first-order methods severely increase unlike the simplified IP its iterations will be maintained throughout the random initial states.

Data dispersion

In this section, the data dispersion will be presented in terms of box plots of the total time and their standard deviations in each prediction horizon.

Taking into account the box plots and standard deviation table, it is noted that the Banded ADMM has the largest dispersion of the comparison. It makes sense since

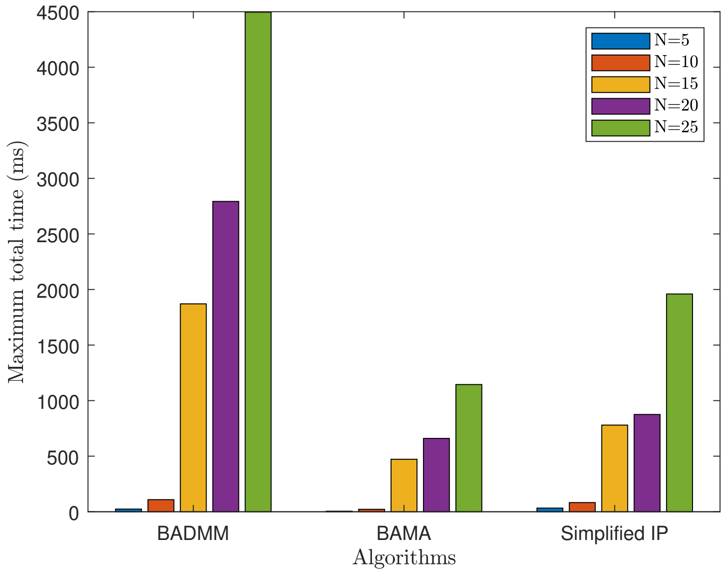


Figure 5.21: Maximum total time in milliseconds for each prediction horizon in both first and second-order algorithms following soft constraints

Table 5.6: Standard deviation of total time (ms)

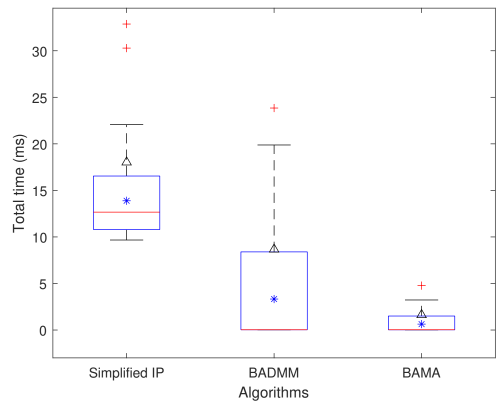
Prediction Horizon	**Simplified IP**	**BADMM**	**BAMA**
5	55.2920	10.2771	2.3369
10	21.2650	55.9403	11.4043
15	53.6094	219.2546	55.8375
20	94.8759	688.7603	155.4223
25	199.0841	490.9147	88.9247

it is a first-order method and their iterations change too much in the simulations even though the constraints are soft. Indeed, this algorithm requires the biggest number of iterations to converge of all the experiment. In addition, the data dispersion increases as the prediction horizon is bigger.

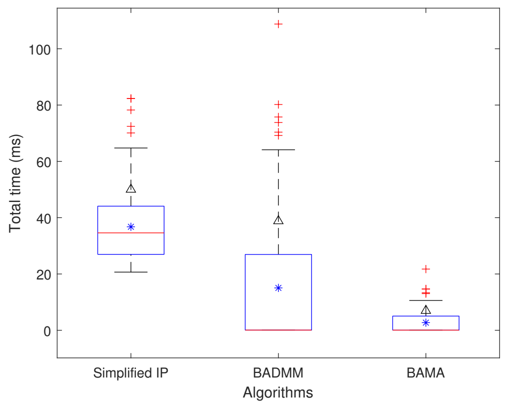
In summary, first-order methods changes their iterations as the size of the QP is bigger and the initial state is selected. Therefore, in order to establish more performance differences between these two types of algorithms, hard constraints are used in the benchmark example. It is expected that the number of iterations increases considerably for the first-order algorithms whereas in the second-order methods, they maintain their iterations in a short range.

5.2.2 Hard constraints

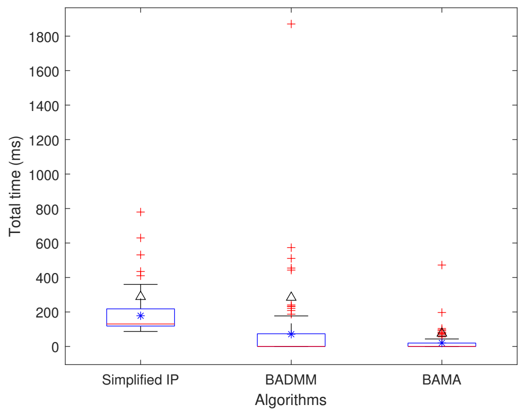
In Figure 5.24, the response of the algorithms under hard constraints for $N = 5$ is presented. As it is shown, the first order algorithms are the most affected group in this comparison since their iterations are increased with respect to the 5.15. This behaviour is due to the reduction of the input constraints and the corresponding initial state. Despite that behaviour, the banded AMA is better than the ADMM variant since its iterations are located between 8 and 36 iterations.



(a) Prediction horizon $N = 5$

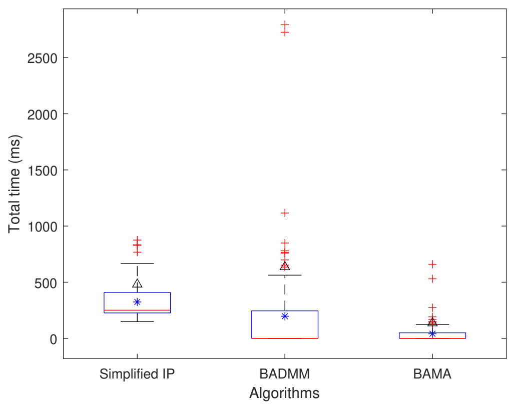


(b) Prediction horizon $N = 10$

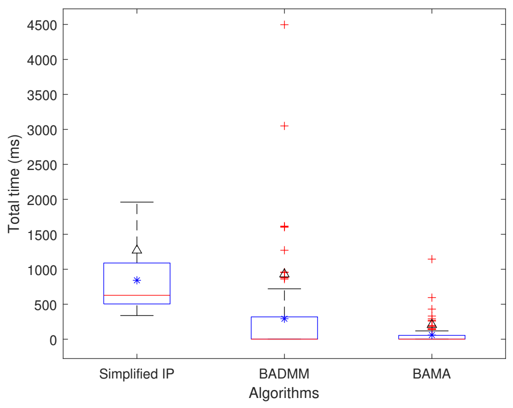


(c) Prediction horizon $N = 15$

Figure 5.22: Box plot of resulting data. ($*$) is the mean and (Δ) is the standard deviation.



(a) Prediction horizon $N = 20$



(b) Prediction horizon $N = 25$

Figure 5.23: Box plot of resulting data. ($*$) is the mean and (Δ) is the standard deviation.

With respect to the second-order method, it is really stable in iterations. However, it is evident that the variation will not be noted since it is the first horizon and the

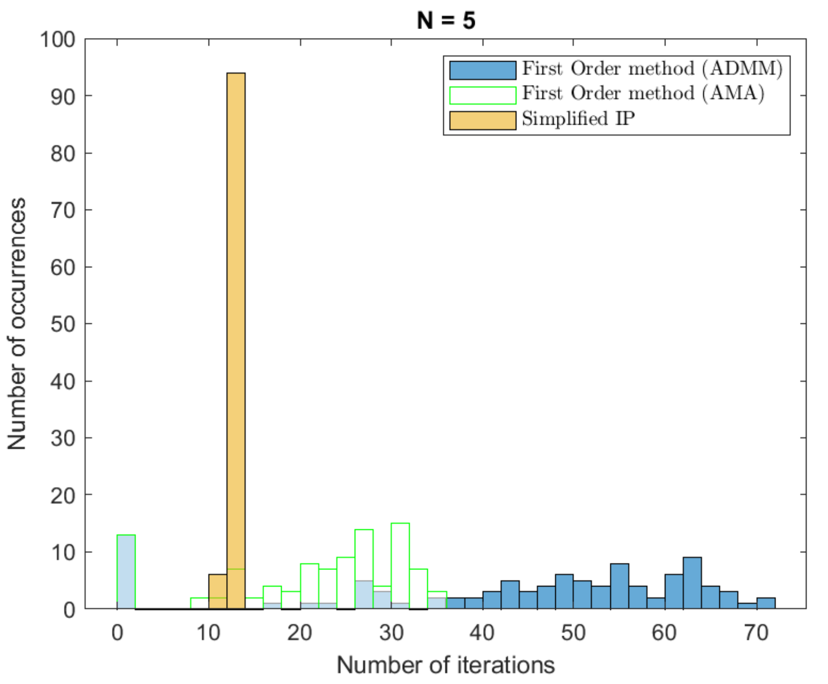


Figure 5.24: Resulting histograms for N=5 using hard constraints

construction of the matrices are not big. Thus, it is expected that the algorithms have a slightly variation in their iterations as the prediction horizon increases.

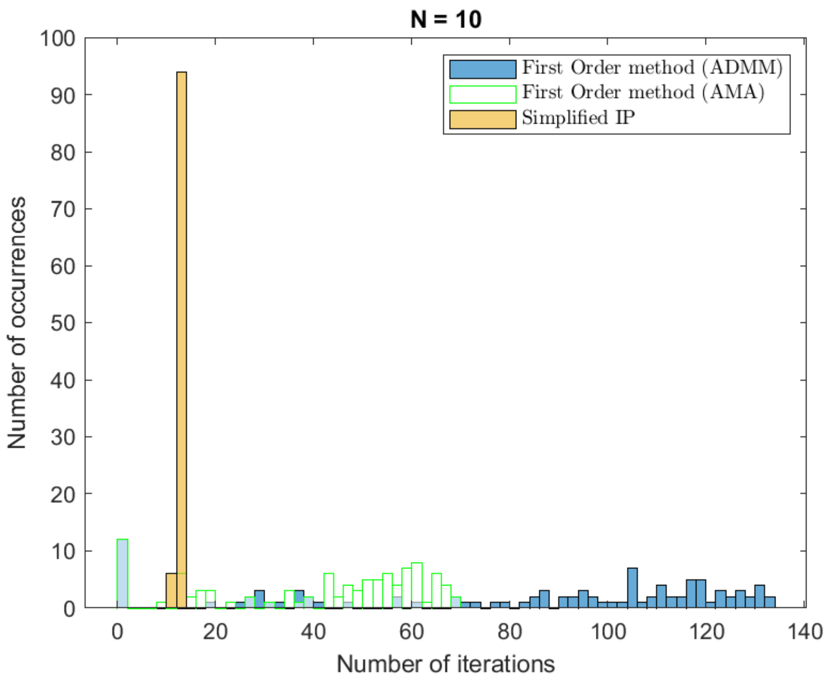


Figure 5.25: Resulting histograms for N=10 using hard constraints

In the Figure 5.25, the results corresponding to the prediction horizon equal to 10 is presented. According to these histograms, the banded ADMM iterations are considerably increased. The same pattern is shown for AMA variant but the number of iterations is less than ADMM and it does not exceed the 70 iterations.

In the case of the simplified IP, it maintains its range during all the random initial states and this continues proving the stability in terms of iterations of this second-order method. At this point, it is expected for the rest of prediction horizon, the same performance.

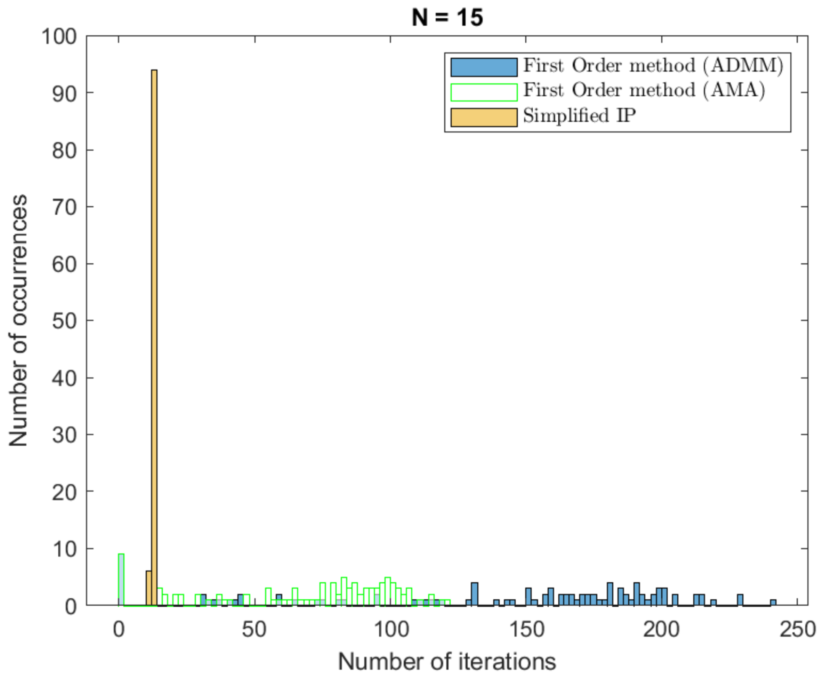


Figure 5.26: Resulting histograms for N=15 using hard constraints

With respect to the graph for the prediction horizon equal to 15, Figure 5.26 is shown. In this case, unlike the second-order methods which have a stable performance in iterations, the first-order algorithms continue increasing such that the banded ADMM has the range of iterations up to 250 iterations. This is due to the fact that the construction of the matrices in the QP problem are increasing and the random initial state is in some occurrences not convenient for solving the problem.

In addition, banded AMA results with less iterations than ADMM variant since its range is up to 125 iterations. Thus, it is noted that the differences between both banded approaches are bigger when the prediction horizon increases considerably.

In Figure 5.27, the results for a prediction horizon $N = 20$ is shown. Here, the differences between both first-order algorithm are notorious since problems for ADMM tend to be solved in up to 360 iterations and for banded AMA, the major concentration of solved problems is located around 200 iterations.

With respect to the second-order algorithm, the simplified IP has increased its range solving the half of the total problems in a range between 12 and 14 iterations. This new performance for the simplified IP reflects the increase of the complexity in the solution of the problems and the sizes of the constructed matrices in the QP.

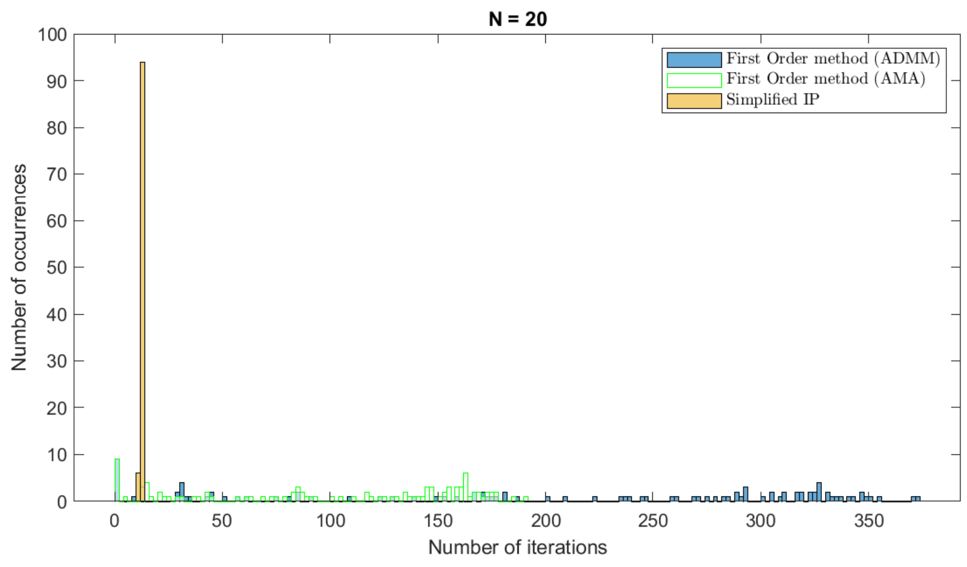


Figure 5.27: Resulting histograms for N=20 using hard constraints

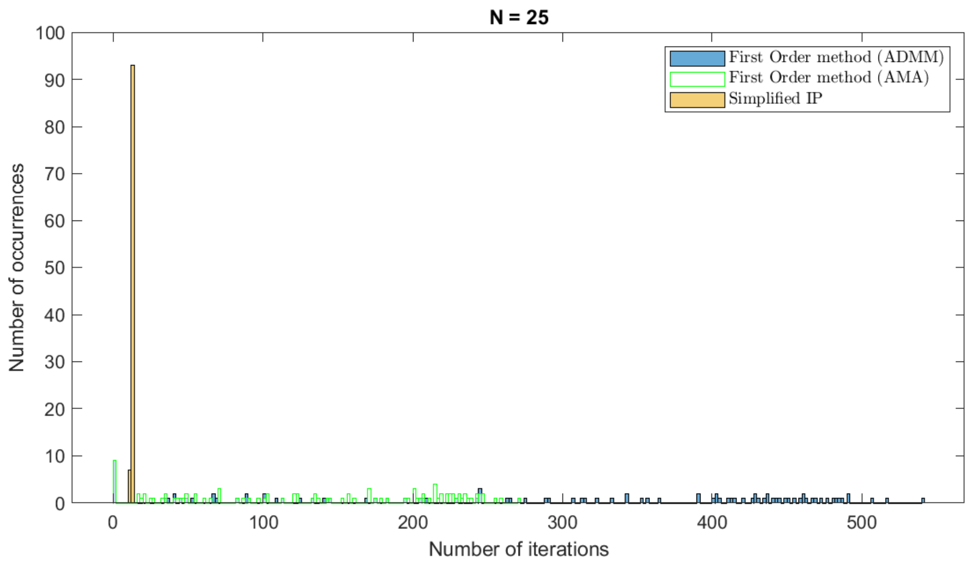


Figure 5.28: Resulting histograms for N=25 using hard constraints

In the last prediction horizon tested, Figure 5.28 shows the performance of the optimizations algorithms according to their number of iterations.

In this case, the ranges from both first-order methods are really separated and it is easy to identify that the banded ADMM need more iterations to solve this kind of problems with a big prediction horizon. In addition, banded AMA is better than the ADMM variant since their number of iterations are smaller.

On the other hand, in order to see clearer how the second-order methods are resulted, Figure 5.29 shows a zoom of the area around these algorithms.

The simplified IP solves all the QP problems between 10 and 14 iterations. As it is seen, comparing these results with the previous prediction horizon, the number of

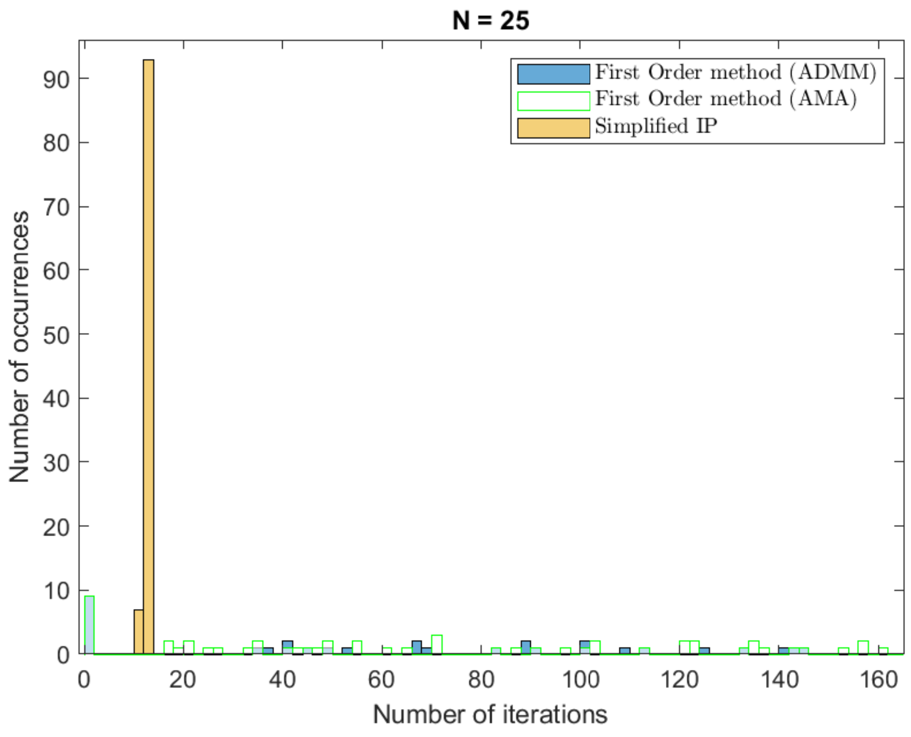


Figure 5.29: Zoom of second-order histograms for N=25 using hard constraints

occurrences for the range from 12 to 14 iterations is increased.

Taking into account the above-mentioned statements according to the pattern shown for each prediction horizon, the simplified IP demonstrates a more stable performance than the first-order algorithms. This makes sense since this is a main characteristic for the second-order methods.

On the contrary, the first-order algorithms deteriorate their performance as the prediction horizon is bigger since it only uses the gradient to solve the problem. Thus, it is concluded that if the problem is stricter due to the size of the QP, harder constraints or the initial states, the total time increase substantially and this is a crucial disadvantage in comparison to the second-order methods.

In addition, it is said that considering hard constraints, the second-order methods maintain their stability despite that the initial state is different. This is not the same for first-order methods which, as it is seen before, they increase their substantially iterations if the problem is bigger and if the initial state is far for the solution.

On the other hand, in order to complete the analysis for hard constraints the measurement of the total times in each prediction horizon is calculated.

In Figure 5.30, the average total time is obtained by multiplying the average time per iteration of the algorithms and the maximum number of iterations performed by each one in this experiment. According to the results, the first-order algorithms have increased their iterations in comparison to the case of soft constraints whereas the

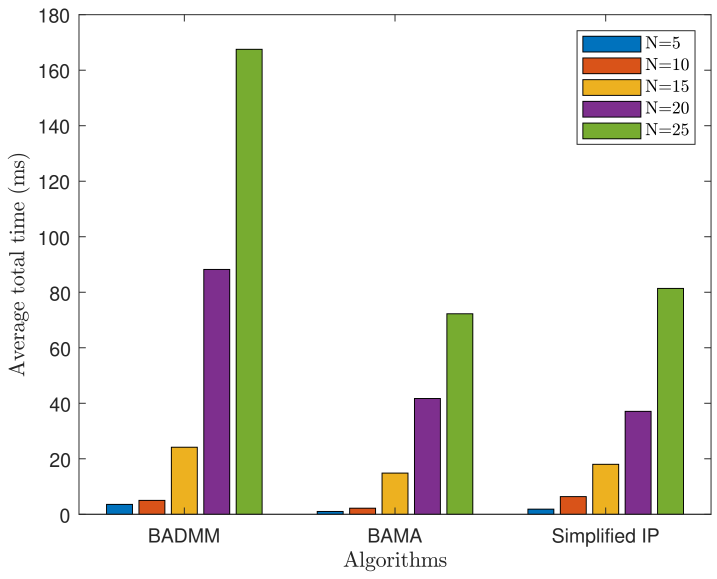


Figure 5.30: Comparison of first and second-order algorithms in terms of average total time (ms)

behaviour of the second-order methods, in terms of iterations, is almost the same. In addition, the banded ADMM increases its average total time due to this algorithm requires more iterations than the rest of the methods.

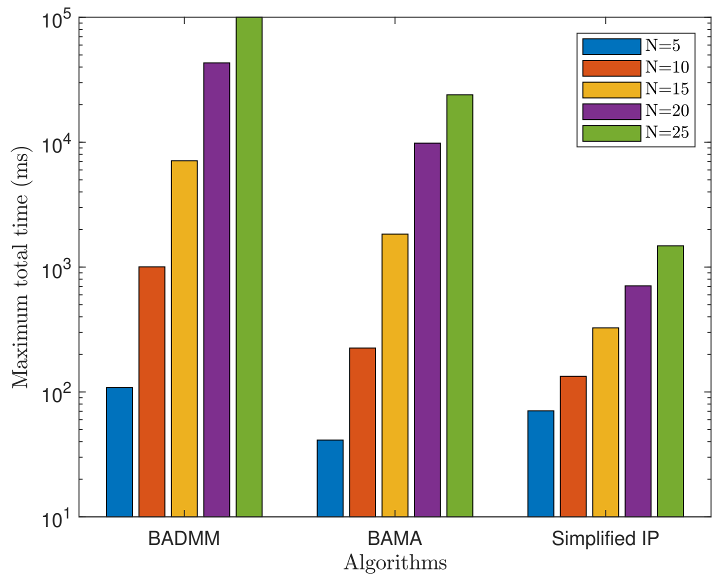


Figure 5.31: Comparison of first and second-order algorithms in terms of maximum total time (ms)

On the same way, in Figure 5.31, the maximum total time is obtained. This variable is calculated by multiplying the maximum time and the maximum number of iterations for each algorithm in the simulations performed. Here, the differences between algorithms are quite substantial since the first-order algorithms has the largest total time of the comparison. Also, the stability of the second-order algorithm, in terms of maximum total time, is noted. This proves, as it is expected, that the number of iterations increases considerably since the complexity of the problem is larger. Also, the stability of the second-order methods is noted which maintains their performance throughout prediction horizons.

Data dispersion

Standard deviation and box plots are described in order to measure the dispersion of the resulting data.

Table 5.7: Standard deviation of total time (ms)

Prediction Horizon	**5**	**10**	**15**	**20**	**25**
Simplified IP	12.2077	26.9078	65.2187	144.4892	286.6258
BADMM	27.2591	229.9912	1669.5254	11874.9354	30263.8017
BAMA	6.77619	48.4888	490.4169	2824.2196	6480.5713

Based on the results presented in the box plots and the Table of the standard deviation, the banded ADMM is the algorithm with most dispersion of the comparison. This occurs due to the iterations of the ADMM variant increase considerably when the complexity of the problem is larger. This behaviour is a main characteristic of the first-order methods since the increase of the iterations also occurs with the AMA variant but less dispersion than the BADMM.

With respect to the second-order algorithm, its iterations and time until convergence do not change considerably and thus, its total time and dispersion are much more stable.

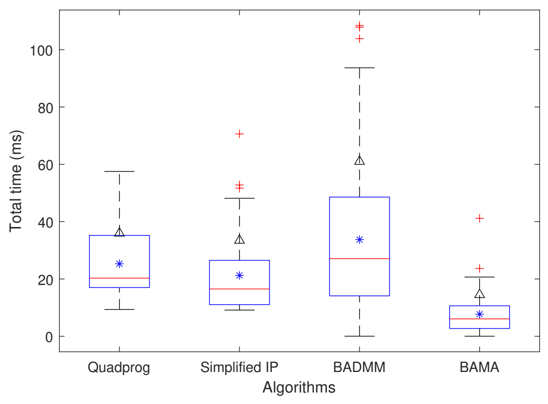
5.3 Closed Loop

In order to analyse the behaviour in closed loop of the first and second-order methods used in the previous section, a study based on simulations in each prediction horizon with a selected control objective is evaluated.

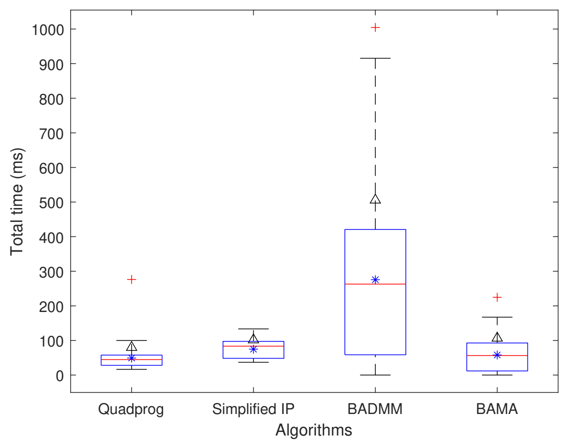
In this case, the benchmark example have the same structure that the previous section. It is described as a set of oscillating masses attached to walls in both extremes by springs. The objective control is to stabilize the masses with the inputs resulting an equilibrium in the states and inputs after a number of steps selected.

This is a soft control objective; however, it can be useful to identify the main characteristics of the algorithms throughout the control loop. In addition, the purpose of this experiment is to recognize how the iterations change during the control as the initial state in each sampling time is redefined by feedback.

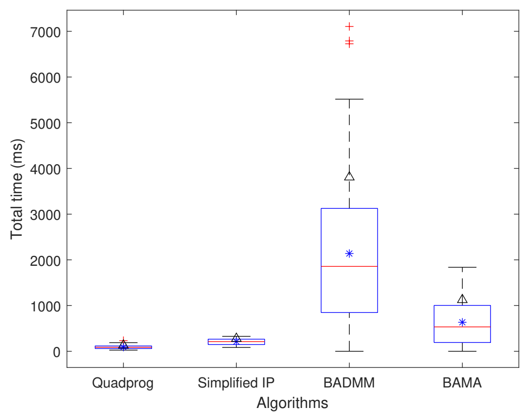
In this case, the system is sampled every 0.5 seconds assuming masses and spring constants with a value of 1kg and $1Nm^{-1}$, respectively. As it is said before, the



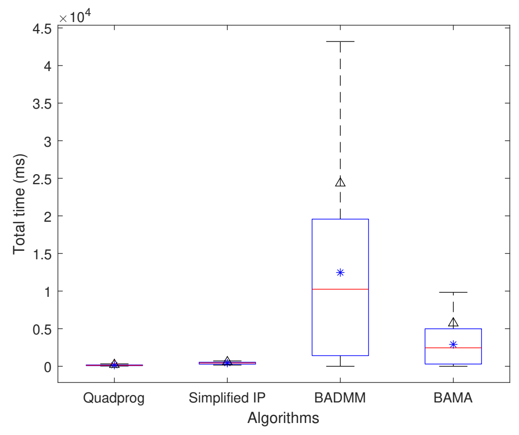
(a) $N = 5$



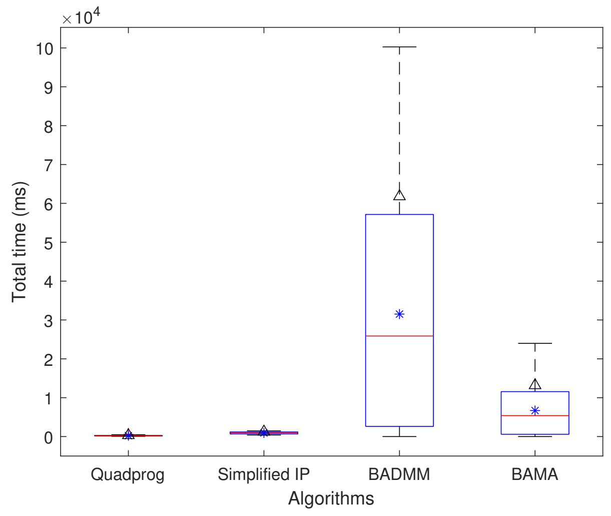
(b) $N = 10$



(c) $N = 15$



(d) $N = 20$



(e) $N = 25$

Figure 5.32: Box plot of resulting data. $(*)$ is the mean and (Δ) is the standard deviation.

system has two control inputs, three masses and two states for each mass, its position and velocity, for a total of six states. The prediction horizon used is defined as $N = [5, 10, 15, 20, 25]$.

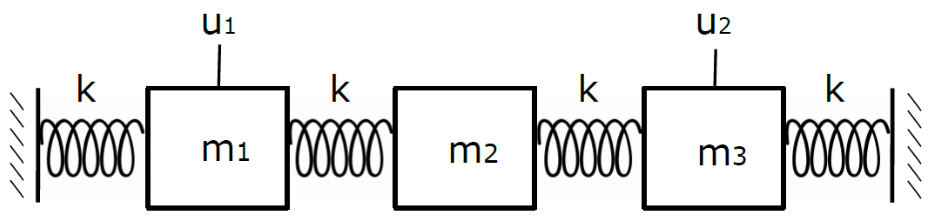


Figure 5.33: System tested in closed loop

The initial state is randomly generated in order to evaluate how stable are the performance of these algorithms starting from different states and how to develop the control throughout the control loop.

Therefore, 100 random initial states are generated for each prediction horizon and the constraints are defined as $\|x_k\|_\infty \leq 3.5$ and $\|u_k\|_\infty \leq 0.5$.

The algorithms tested have the following configurations: For the simplified IP: $sigma = 0.1$, $gamma = 0.1$ and $mulimit = 0.1$. The first-order algorithms have, as stopping conditions, a primal and dual error equal to $1e^{-4}$ with step-size defined in [Ghadimi et al., 2015] for ADMM and $\rho = 0.99 * 2/\lambda_{max}(GH^{-1}G^T)$ for AMA.

Taking into account the parameters above defined, the experiment obtains the following results.

As it is shown in Figure 5.34, the control sequence is the same for the three algorithms; however it is necessary to know how the number of iterations progresses along the steps of the control according to the prediction horizon evaluated.

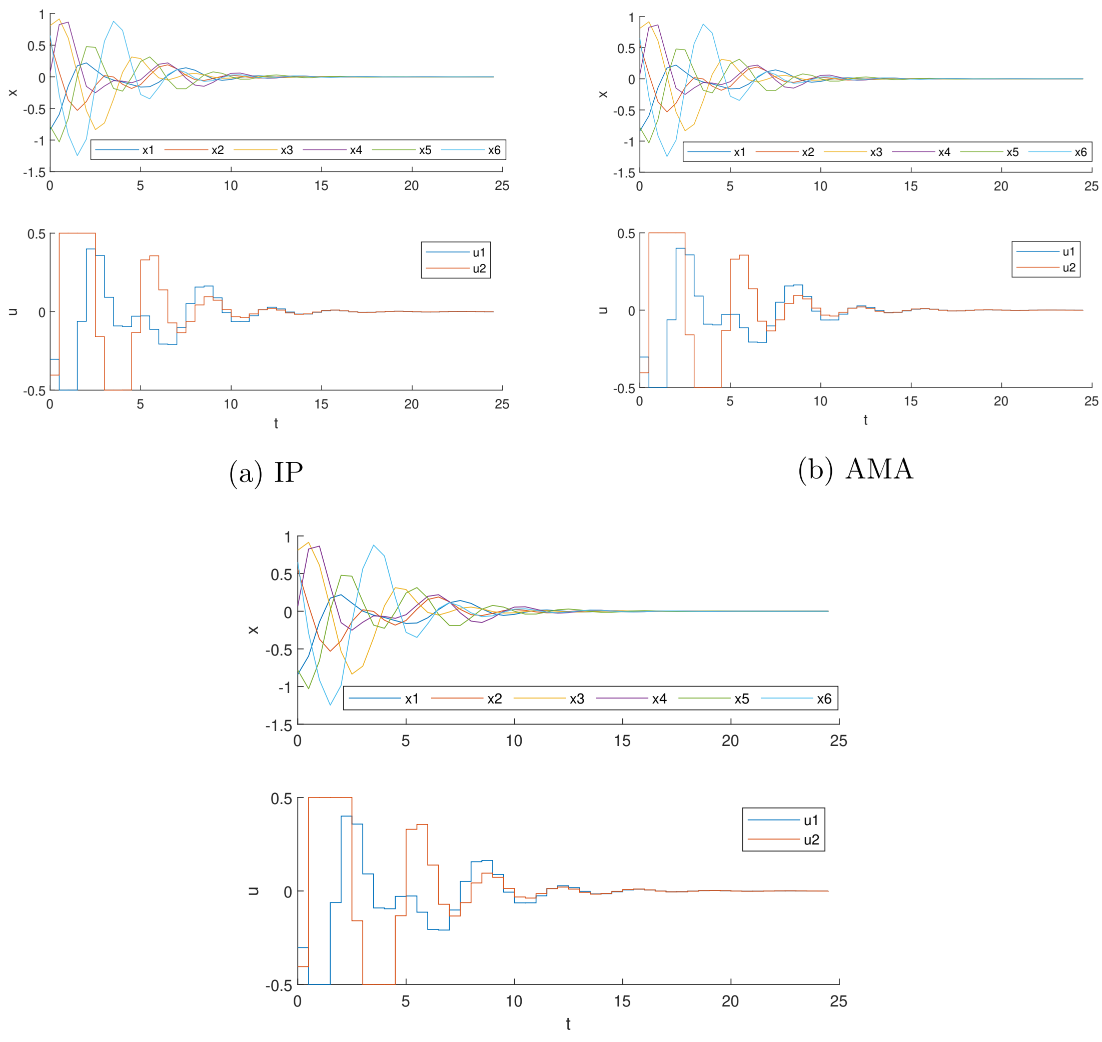
5.3.1 Patterns

In order to establish a pattern in the behaviour of the algorithms, the maximum values in each step, considering the 100 initial states, are calculated obtaining a set of 50 values (equal to the number of steps) for all the control sequence. This calculation is carried out for all the prediction horizons.

As it is shown in Figure 5.35 for a prediction horizon equal to 5, the number of iterations for the simplified IP barely change throughout the number of steps; thus, this proves that the initial state do not alter the response of this second-order method. However, this is not the case of the first-order algorithms which iterations increase in the first part of the control sequence and they reduce considerably their iterations (and logically the time until the convergence) when the control is close to the stabilization.

Comparing the two first-order algorithms, ADMM variant requires more iterations than the banded AMA. Thus, it is expected that the difference of iterations between these algorithms increases as the prediction horizon is bigger.

In Figure 5.36, the pattern for $N = 10$ is presented. The behaviour of all the algorithms is practically the same and the difference between the iterations of BADMM



(a) IP (b) AMA

(c) ADMM

Figure 5.34: Example to resulting control for the three algorithms

and BAMA continues increasing.

In the Figure 5.37 and Figure 5.38, the patterns for $N = 15$ and $N = 20$, respectively, are shown. Here, the first-order algorithms greatly increase their iterations and the stability of the iterations for the simplified IP is noted as the best option when the size of the QP problem is augmented.

On the other hand, in terms of first-order algorithms, banded AMA has better performance than the banded ADMM with the same initial states to solve the same QP problem.

Finally, in Figure 5.39, the pattern for $N = 25$ is presented. As it is shown, the differences of iterations are notorious being the best choice the second order algorithm.

In addition, once presented the bar plots for each prediction horizon, it is worth highlighting that with this control objective, the first-order algorithms tend to solve

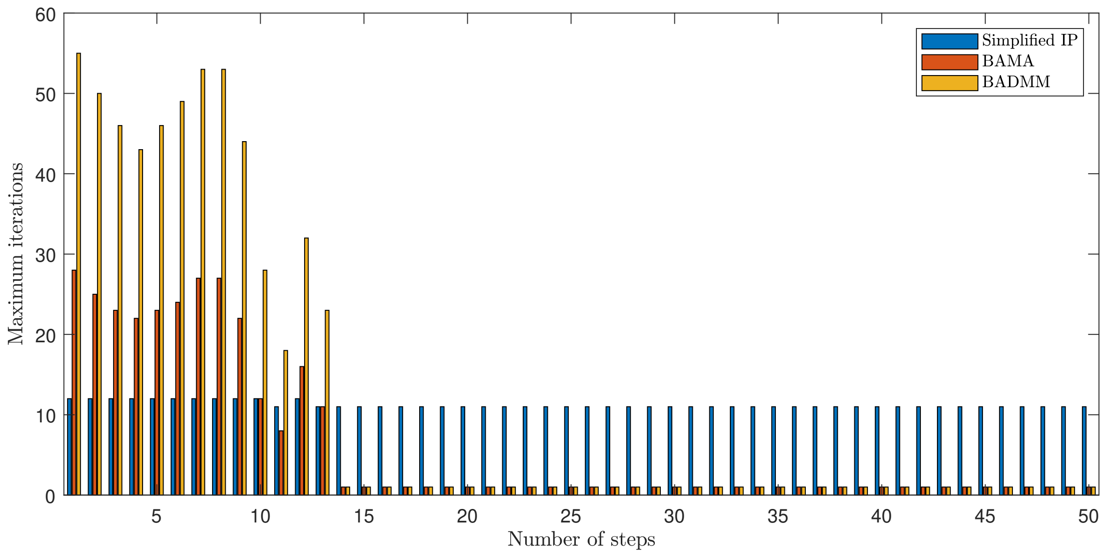


Figure 5.35: Resulting iterations pattern for the closed loop simulations in $N = 5$

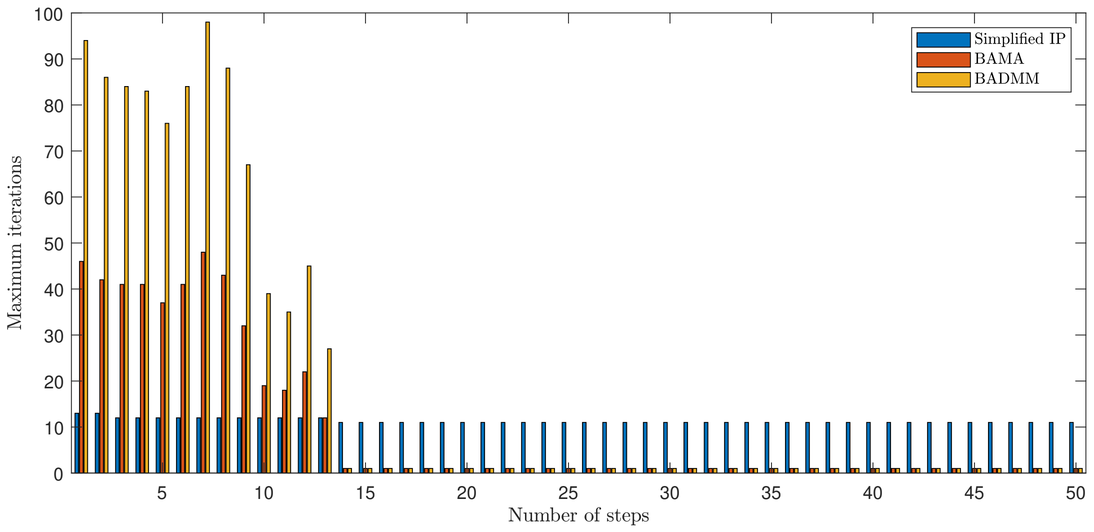


Figure 5.36: Resulting iterations pattern for the closed loop simulations in $N = 10$

the problem in 1 iteration. This occurs only when the control is close to the equilibrium; however, if the control objective is stricter, the iterations will vary throughout the number of steps and this could be a drawback if the sampling time need to be short.

Based on the results before shown, it is noted that the effort of the controller is bigger when the prediction horizon is increased. In this case, the maximum prediction horizon tested is $N = 25$; thus, the worse result of each study will be obtained in this horizon. Therefore, the maximum total time is shown for this prediction horizon and it is calculated by multiplying the time until the convergence and the number of iterations performed for each step control for the 100 problems run. Once multiplied, these values are obtained by selecting the maximum total time for each step in the control loop.

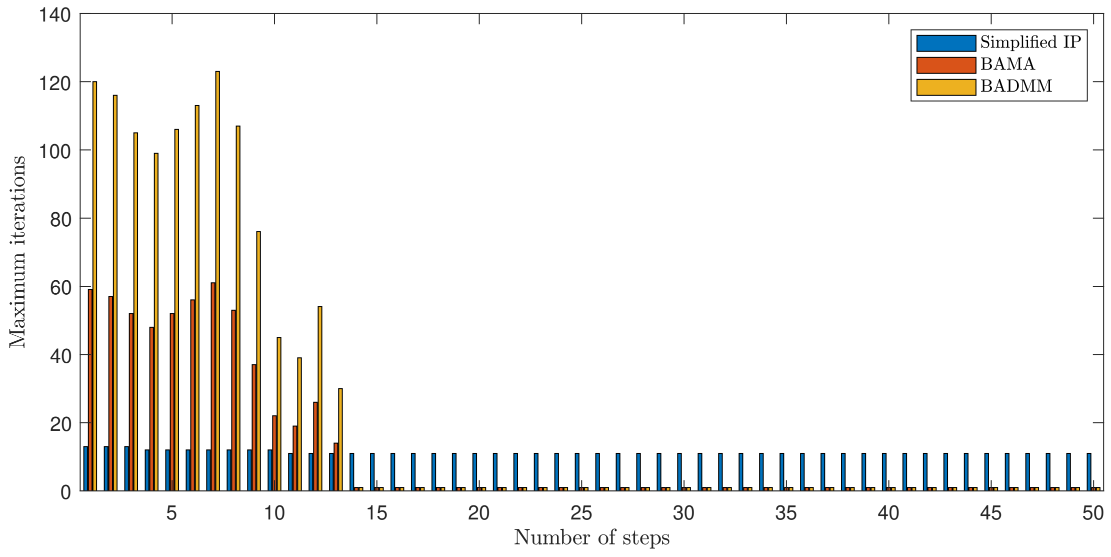


Figure 5.37: Resulting iterations pattern for the closed loop simulations in $N = 15$

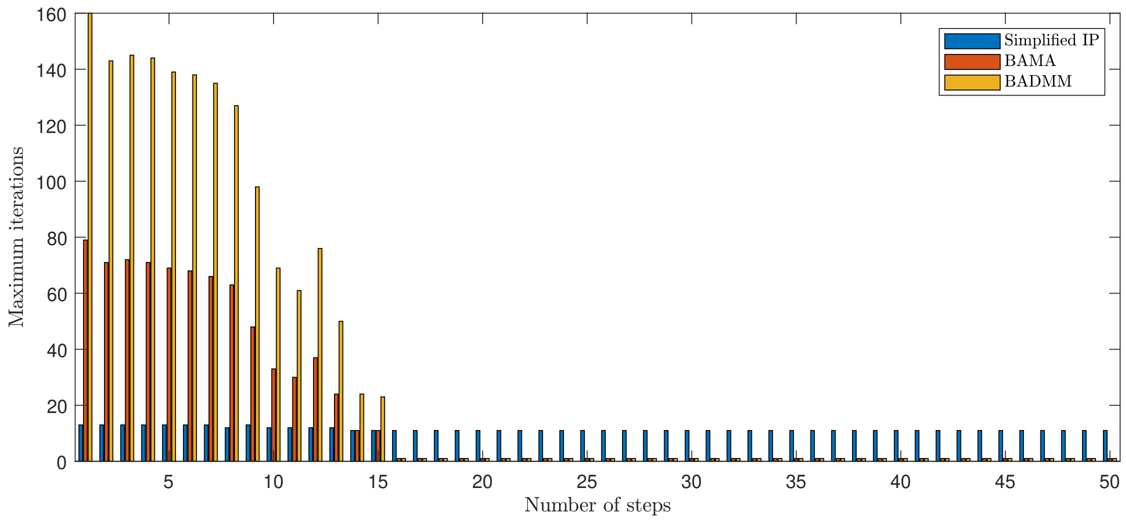


Figure 5.38: Resulting iterations pattern for the closed loop simulations in $N = 20$

As it is shown in Figure 5.40, the results for the maximum total time in the maximum prediction horizon tested is presented. In this case, the first-order algorithms seem to have better performance than the simplified IP; however, this behaviour is noted only for the case when the complexity of the problem is smooth. In addition, as the reference is the equilibrium, the iterations after half the number of steps are equal to 1 due to the effort of the controller at this point is minimal.

It is worth highlighting that the Figure 5.40, is not a decisive plot about the performance of the algorithms due to the fact that they are running according to soft constraints; for this reason, it is necessary to carry out the same interpretation when the constraints are harder.

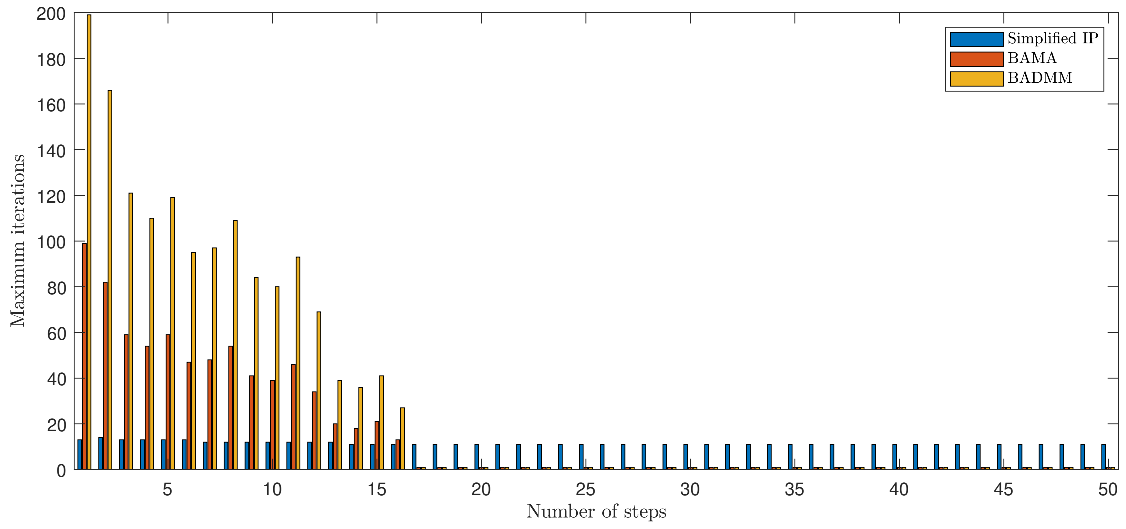


Figure 5.39: Resulting iterations pattern for the closed loop simulations in $N = 25$

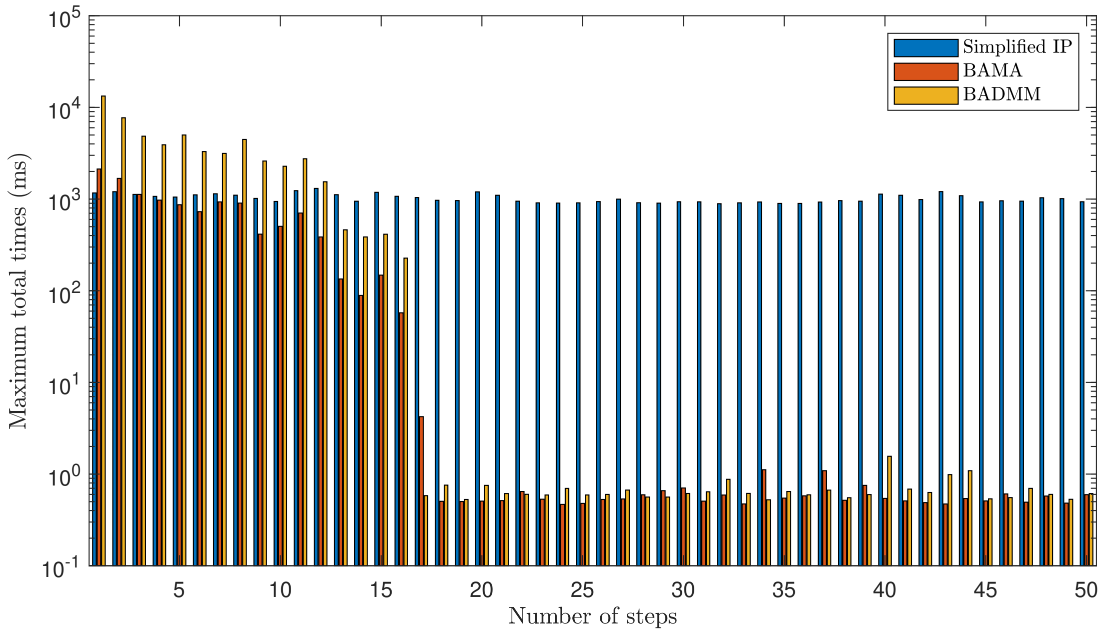


Figure 5.40: Resulting maximum total time pattern for the closed loop simulations in $N = 25$

5.3.2 Total times

Total times give more information about the performance of the algorithms during the closed loop. Thus, it is necessary to obtain the maximum time until the convergence of the algorithms for all the simulations.

In Figure 5.41, the maximum time until the convergence increases for all the algorithms as the size of the QP problem is larger. This is due to the fact that the prediction horizon is evaluated following a scalability.

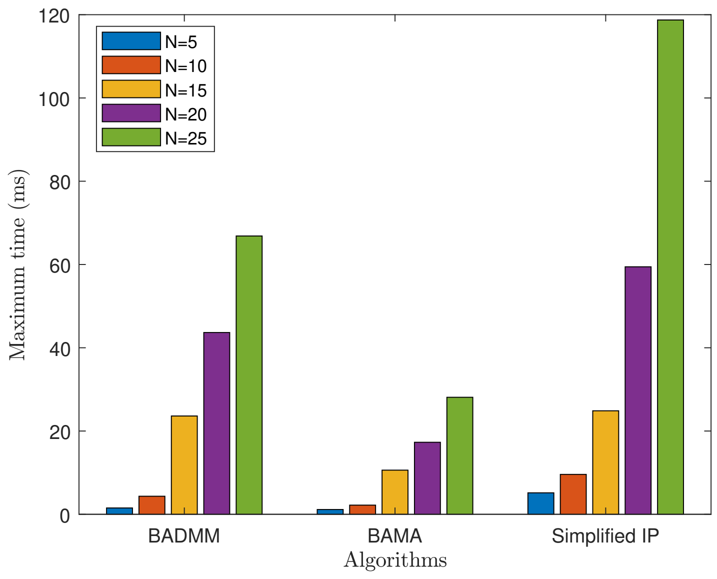


Figure 5.41: Maximum times for 100 random initial states per horizon

With respect to the first-order algorithms, the banded AMA formulation has the shortest maximum time unlike the banded ADMM which maximum time increases quickly.

Despite that the second-order method presents a larger maximum time than the first-order algorithms, this does not mean that the performance of them are better than the simplified IP since the iterations for the first-order methods are reduced to 1 when the control objective is the equilibrium. Thus, it could be consistent to change the control objective for evaluating the case when the iterations of these methods are different to 1.

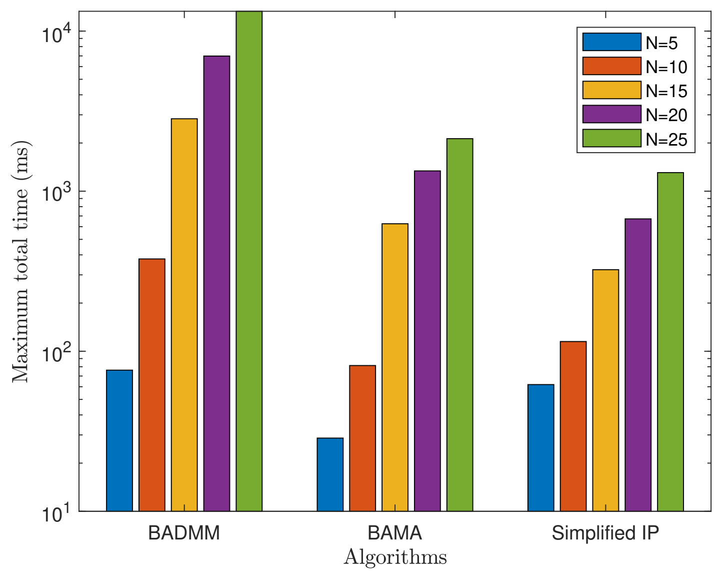


Figure 5.42: Maximum total times for 100 random initial states per horizon

In Figure 5.42, the maximum total times are presented. The total times are obtained by multiplying the time until the convergence and the number of iterations performed for each sampling time of the control sequence. Once multiplied, the maximum total time is obtained by selecting the maximum total value of all the control sequence. Here, the banded ADMM is clearly the worst algorithm of the comparison since their number of iterations are the largest and therefore, the total time increases considerably.

Furthermore, simplified IP demonstrates a good performance since it has the shortest maximum total time of the comparison over all the prediction horizons. This makes sense since their number of iterations does not change so much than the first-order algorithms iterations.

In summary, it is clear the advantages of using the second-order methods to solve QP problems when the complexity of these problems increase due to the constraints, the initial states or the prediction horizon selected.

5.4 Closed Loop with control objective modified

The purpose of this section is to define a stricter control objective in order to identify the performance of the algorithms tested when the complexity of the problem is larger.

The benchmark example is the same that in the previous section . However, there is a main difference in the structure of the system in comparison to the previous problem since the oscillating masses only are attach to wall in a extreme. Thus, the objective control consist on following the reference defined for the inputs and states after a number of steps selected.

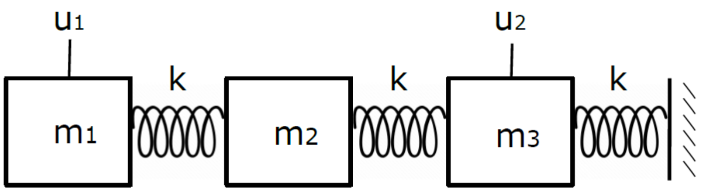


Figure 5.43: Structure of problem tested

This control objective is useful to identify the main characteristics of the algorithms throughout the control loop and how the iterations change during the control as the initial state in each sampling time is redefined by feedback. The initial state is randomly generated in order to evaluate how stable are the performance of these algorithms starting from different states and how to develop the control throughout the control loop.

Therefore, 100 random initial states are generated for each prediction horizon and the constraints are defined as $\|x_k\|_\infty \leq 3.5$ and $\|u_k\|_\infty \leq 0.5$. The algorithms tested have the following configurations: For the simplified IP: $sigma = 0.1$, $gamma = 0.1$ and $mulimit = 0.1$. The first-order algorithms have, as stopping conditions, a primal and dual error equal to $1e^{-4}$ with step-size defined in [Ghadimi et al., 2015] for ADMM and $\rho = 0.99 * 2/\lambda_{max}(GH^{-1}G^T)$ for AMA.

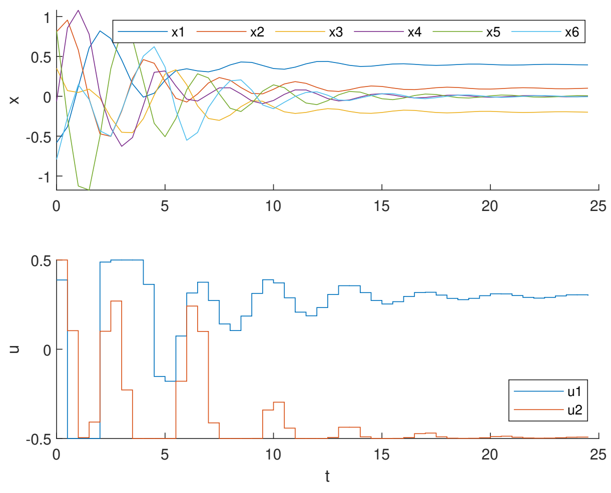


Figure 5.44: Example of closed loop for IP

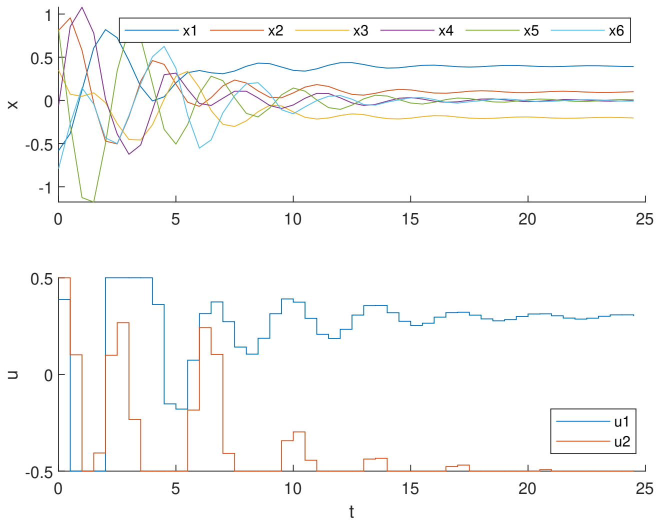


Figure 5.45: Example of closed loop for BAMA

In Figure 5.44 - 5.46, an example of close loop performance for the IP, BAMA and BADMM is presented. As it is shown, the two inputs are different to 0 throughout the control sequence.

It is worth highlighting that the reference is fixed to $x_{ref} = [0.4; 0; -0.2; 0; 0; 0]$ and $u_{ref} = [0; 0]$ for all the problems since only a scenery different to the stabilization of the decision variables (previous section) is needed to identify the performance of these algorithms.

Now, an identical analysis to the previous section is carried out taking into account maximum iterations patterns for each prediction horizon and total times in order to

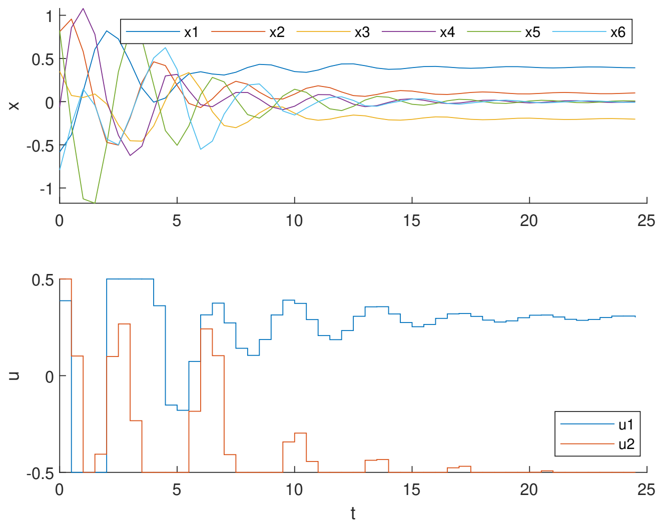


Figure 5.46: Example of closed loop for BADMM

study the behaviour of each algorithm in this different scenery to the stabilization.

5.4.1 Patterns

In this part, the maximum values in each step considering the 100 initial states are calculated obtaining a set of 50 values (equal to the number of steps) for all the control sequence. This calculation is carried out for all the prediction horizons.

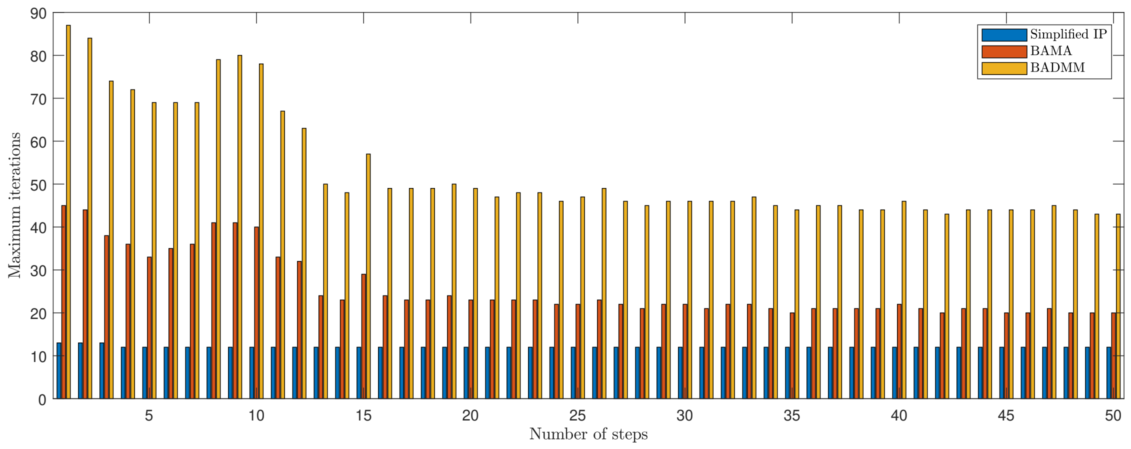


Figure 5.47: Resulting iterations pattern for the closed loop simulations in $N = 5$

In Figure 5.47, the closed loop simulations for all the algorithms in $N = 5$ is presented. As it is shown, the pattern is different to the analysis in the previous section since the iterations are different to 1 in all the number of steps.

Even though the prediction horizon is 5, the differences in number of iterations between algorithms are high and the algorithm with less number of iterations the simplified IP (second-order method). This behaviour makes sense since the stability of the second-order method in terms of iterations is a main characteristic of them when solving QP problems and they are not changed in spite of varying the initial state or the reference trajectory.

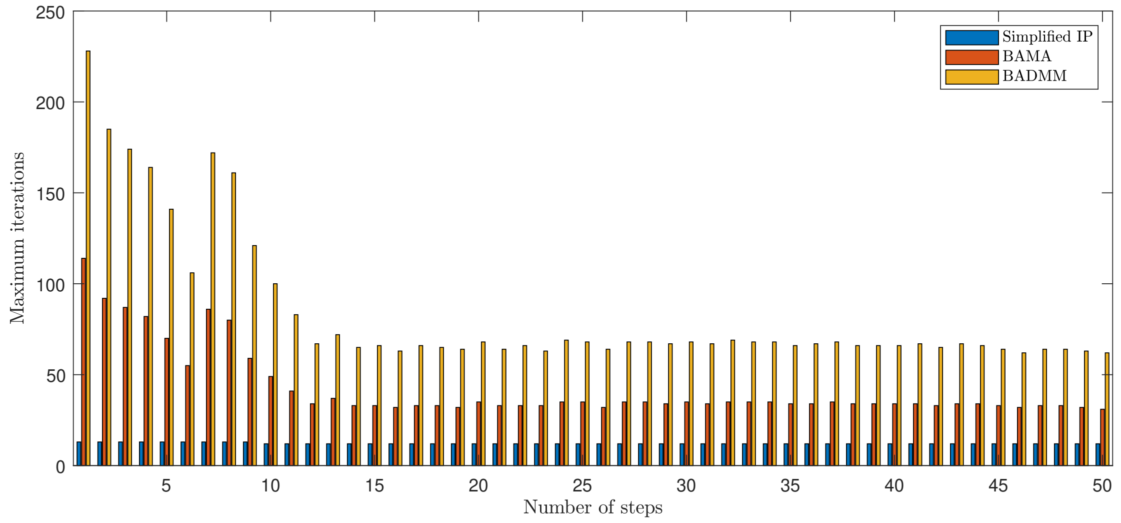


Figure 5.48: Resulting iterations pattern for the closed loop simulations in $N = 10$

As it is shown in Figure 5.48, the number of iterations continues increasing for the first-order methods where the banded ADMM has more iterations than the banded AMA. It is expected that the iterations of banded AMA and banded ADMM increase much more as the prediction horizon is bigger.

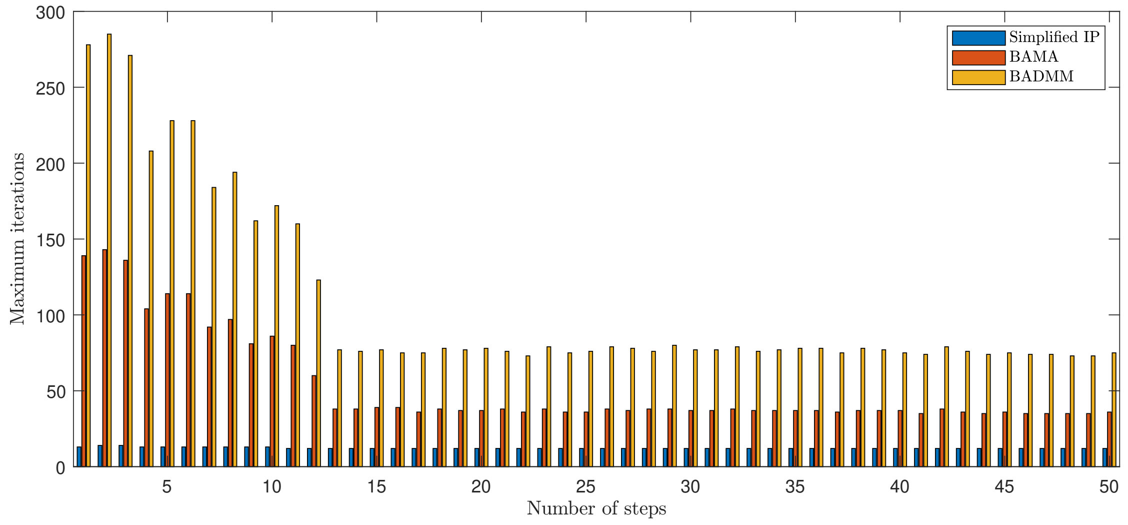


Figure 5.49: Resulting iterations pattern for the closed loop simulations in $N = 15$

Based on the results shown in Figure 5.49 and Figure 5.50, the best performance in these comparisons is the simplified IP. These iterations barely change throughout the number of steps. This result proves that the simplified IP is better when the complexity of the problem is larger.

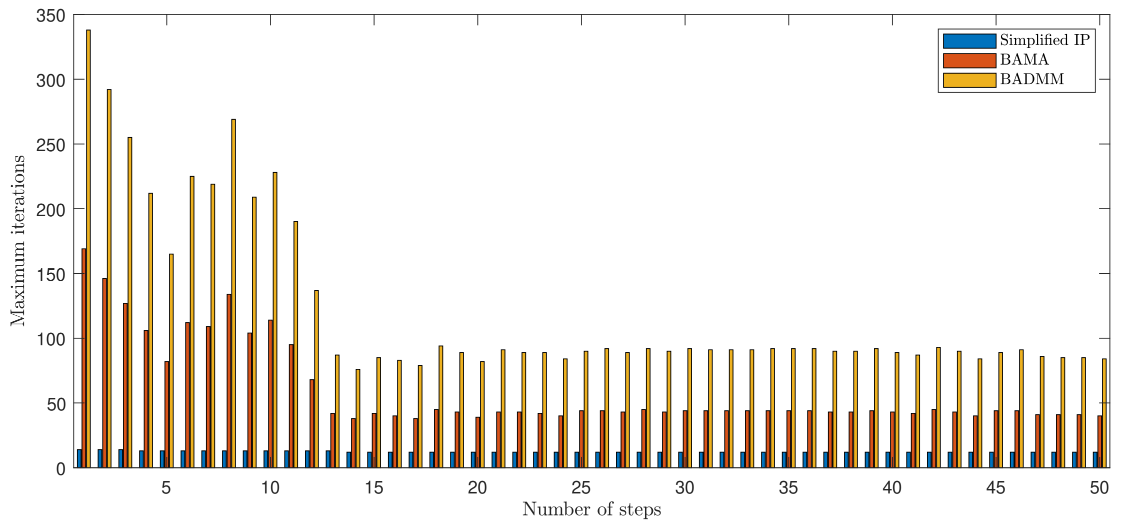


Figure 5.50: Resulting iterations pattern for the closed loop simulations in $N = 20$

It is expected that in a prediction horizon equal to 25, the differences are much more notorious and the first-order methods increase their iterations, with more of them for the banded ADMM.

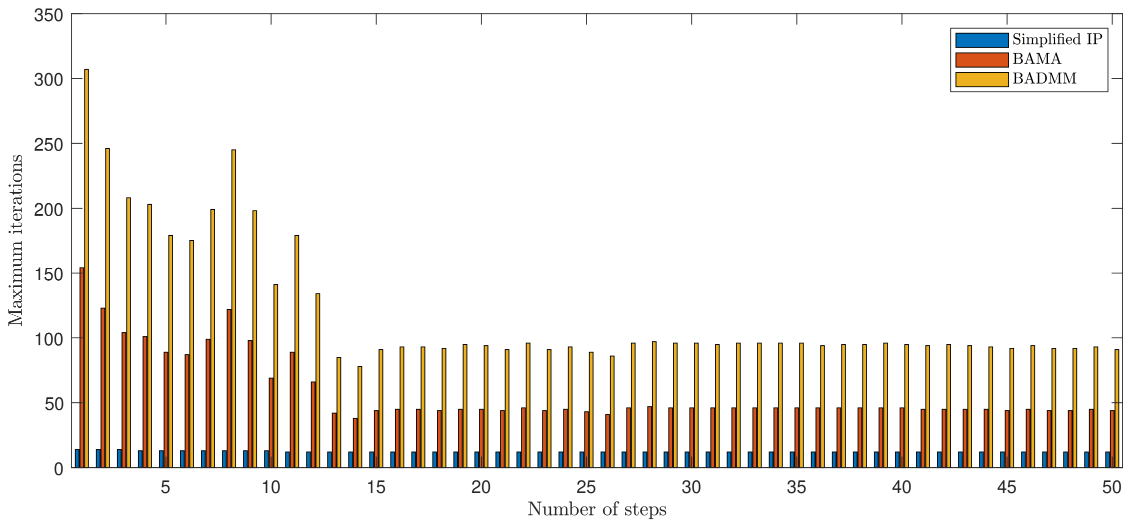


Figure 5.51: Resulting iterations pattern for the closed loop simulations in $N = 25$

According to the results shown in Figure 5.47 - 5.51, it is noted that the first-order methods have a simple structure as algorithm but require a lot of numbers of iterations especially when the problem needs to operate close to the constraints or start in a hard initial state. This is not the case of the second-order algorithm that maintains its iterations in spite of the variations both control objective and QP problem size.

As it was carried out for the problem with soft constraints, the maximum total time of the algorithms is presented. These values are calculated on the same way that for the Figure 5.40. This analysis is focused on how the algorithms change their behaviour (in a hard scenery) in comparison with their performance in a smooth scenery.

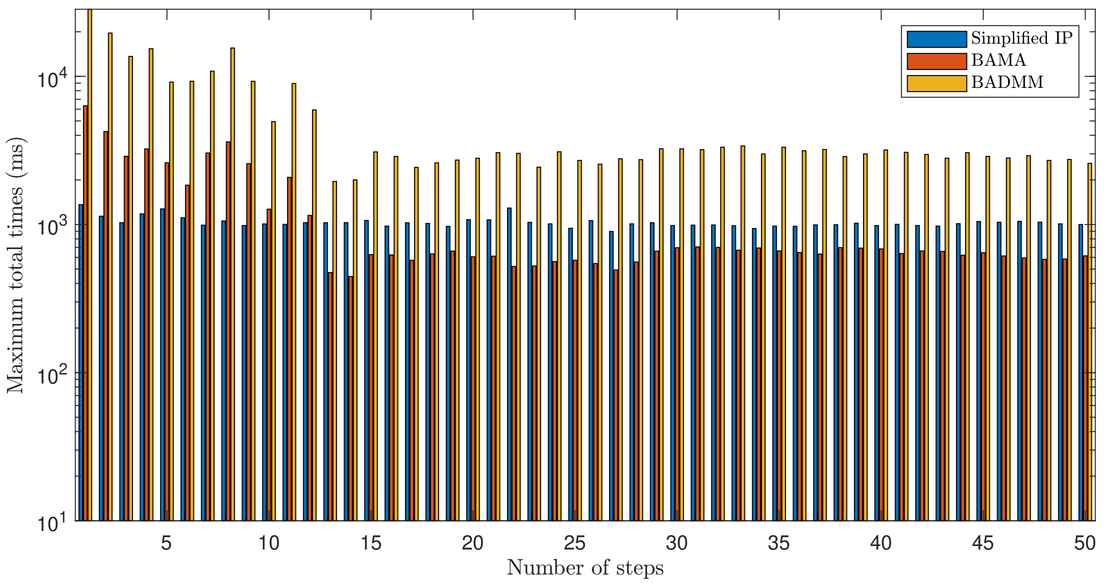


Figure 5.52: Resulting maximum total time pattern for the closed loop simulations in $N = 25$

As it is shown in Figure 5.52, the increase of the maximum total times in the first-order algorithms is evident. This variation is directly related to the increased severity in the constraints. As it is noted, even though the constraints are stricter, the banded AMA could be a good option; however, in case of the complexity of the problem is even more demanding, the maximum total times for the first-order methods will increase substantially and therefore, the simplified IP is selected as the best option due to its performance throughout the different analysis carried out and in particular, when the controller requires demanding operation points along the control sequence.

5.4.2 Total times

Total times need to be calculated to obtain more information about the performance of these algorithms in this new control objective.

In Figure 5.53, the maximum times are calculated by selecting the maximum time until the convergence for all the algorithms and all the steps in each prediction horizon. Compared to Figure 5.41, the maximum times until the convergence for the first-order algorithms increase because the control objective is stricter. In addition, the maximum time for the simplified IP in each prediction horizon is almost the same than their performance in Figure 5.41.

Once presented these maximum times until the convergence of all the algorithms, the maximum total times are calculated. The total times are obtained by multiplying the time until the convergence and the number of iterations performed for each sampling time of the control sequence. Once multiplied, the maximum total time is obtained by selecting the maximum total value of all the control sequence.

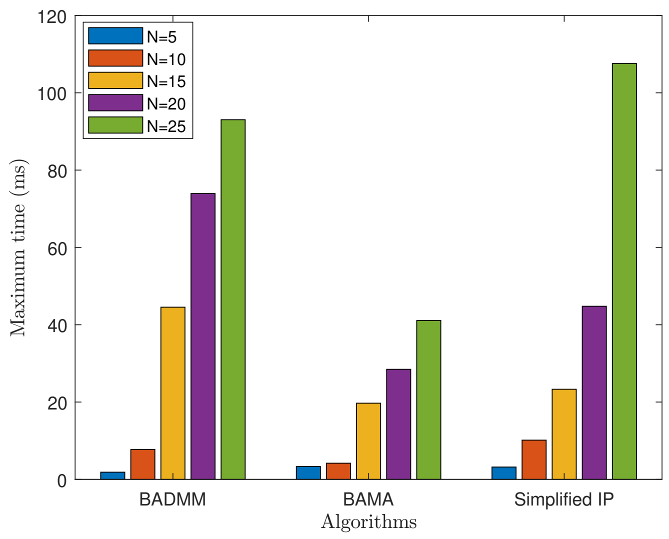


Figure 5.53: Resulting maximum times in ms for the closed loop simulations.

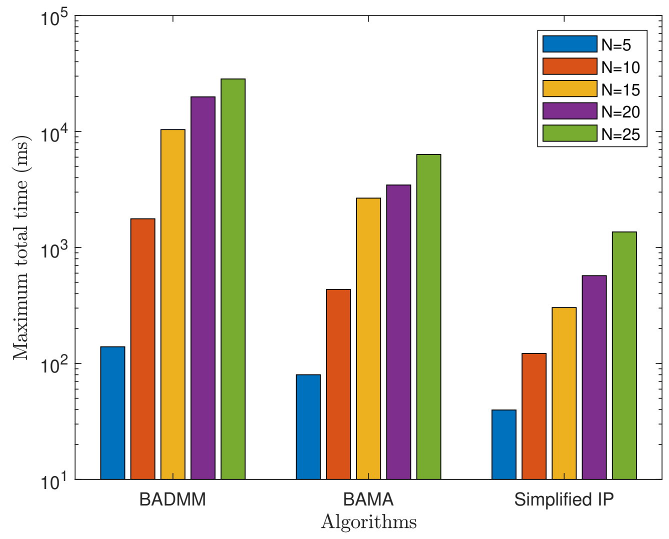


Figure 5.54: Resulting maximum total times in ms for the closed loop simulations.

Based on the results shown in Figure 5.54, first-order algorithms have the largest maximum total times due to their iterations. This is a disadvantage for the first-order methods since the simplified IP barely changes its total time which it is considered as a good performance. Furthermore, these results show how the second-order method does not change both in times and iterations in spite of the parameters are stricter or complexity of the problem is larger.

In summary, the second-order method have demonstrate a better global performance than the first-order methods. This is appreciated in the different comparisons when the first-order methods have increased their statistics whereas the simplified IP barely changes them. Thus, this stability makes the second-order methods more appropriated if the control needs to operate with stricter requirements.

Chapter 6

Experimental results

In this chapter, closed loop problems will be developed to evaluate the embedded behaviour of reduced-Hessian method applied to AMA and ADMM algorithm and the simplified IP, software based MPC controllers on a Xilinx Zynq-7000 XC7020 SoC with embedded dual core ARM Cortex-A9 processor have been tested.

As regards the procedure of implementation, using Matlab Coder R2017b, C-code is generated from the m-file of each variant of the algorithms. The purpose is to analyse how varying the number of decision variables, inequalities and large sized matrices affect the convergence rate and execution time of each algorithm when solving at each sampling step a QP formulated with banded null space.

Furthermore, the functions which are more computationally expensive can be identified in the first-order algorithms and on this way, to analyse the possibility of carrying out a hardware acceleration of these functions.

It will be evaluated two types of control objectives. The first is focuses on referring the control to the equilibrium in the decision variables. And, the second objective is to appreciate the performance of the algorithms when the inputs and states need to operate throughout the control sequence without having the equilibrium as reference.

With regard to the LTI model used in this experiment, it will be the same in each control objective as in the computational study and the MPC controllers will be evaluated with the same horizons N = [5, 10, 15, 20, 25].

6.1 Closed Loop

In this case, the benchmark example have the same structure that the analysis carried out in the computational study. As it was mentioned in section 5.3, the benchmark example is described as a set of oscillating masses attached to walls in both extremes by springs and it is defined as it is shown in Figure 5.33.

The objective control is to stabilize the masses with the inputs resulting an equilibrium in the states and inputs after a number of steps selected.This is a soft control objective; however, it can be useful to identify the main characteristics of the algorithms throughout the control loop and to achieve the identification of variables with a high computational burden. In addition, the purpose of this experiment is to recognize how the iterations change during the control as the initial state in each sampling time is redefined by feedback.

An example to the resulting performance of the algorithms is presented in Figure 6.1 - 6.3.

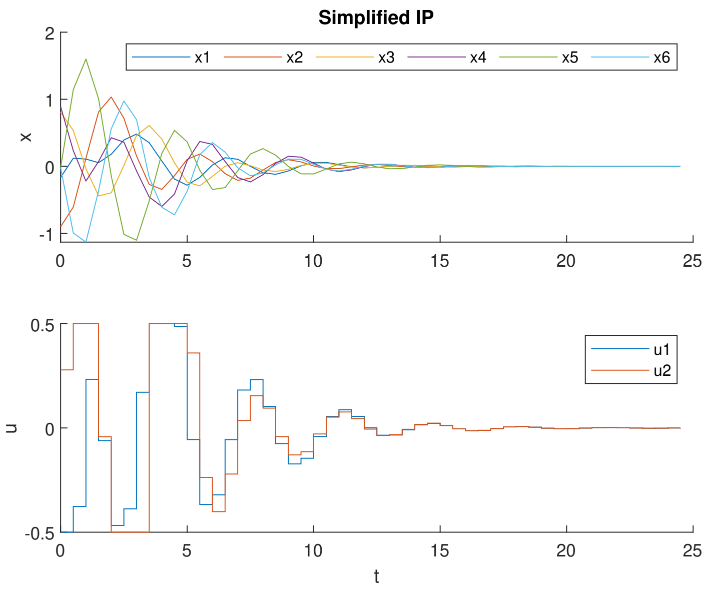


Figure 6.1: Experimental result in closed loop for simplified IP

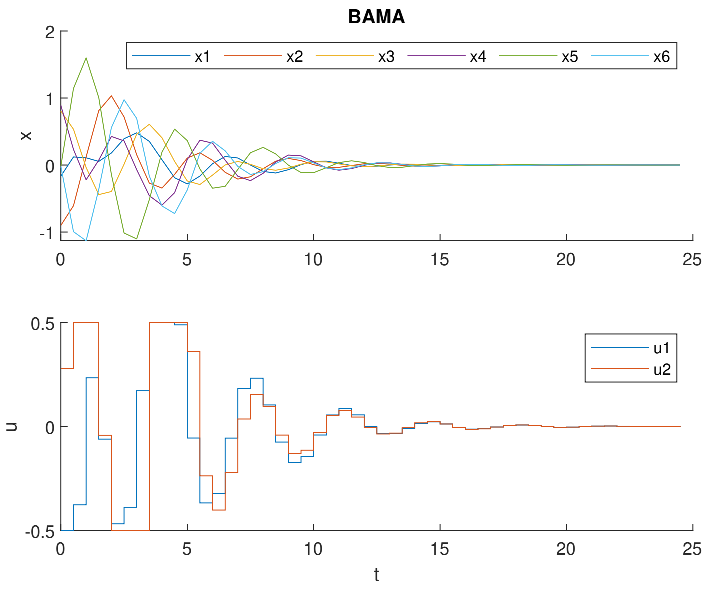


Figure 6.2: Experimental result in closed loop for BAMA

In order to observe the results of the times until the convergence for each algorithm in all the simulations made in the embedded platform, the Table 6.1 is presented. These times were calculated as the time taken by the algorithm in finding the optimal solution

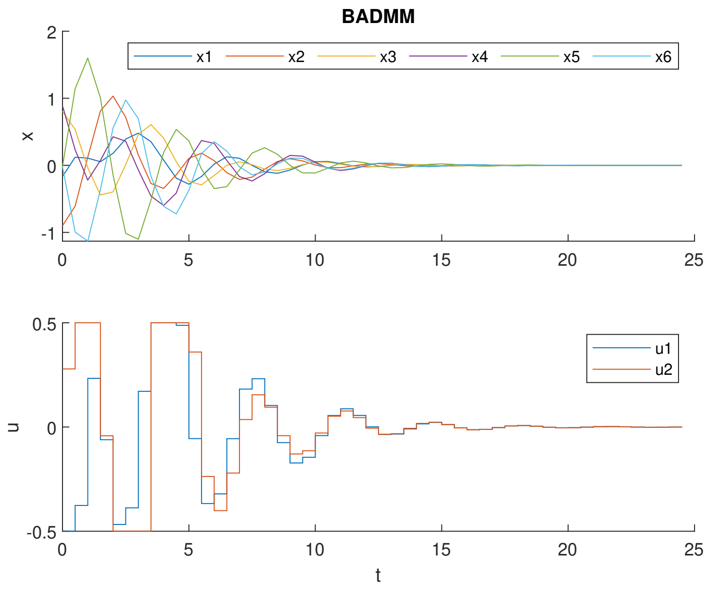


Figure 6.3: Experimental result in closed loop for BADMM

in each sampling time. Thus, there are 5000 values corresponding to the number of steps and the number of problems. Based on these values, the minimum, maximum and mean times are obtained.

Table 6.1: Times in milliseconds of the resulting data considering all the simulations for the algorithms tested

	Simplified IP			**BAMA**			**BADMM**		
Prediction Horizon	Min. value	Max. value	Mean value	Min. value	Max. value	Mean value	Min. value	Max. value	Mean value
N=5	142.7120	176.1680	158.9931	0.6560	20.0600	10.4389	0.8970	50.3640	27.6320
N=10	522.3020	648.7560	584.1297	2.0510	102.9920	56.9920	2.9260	283.0630	159.2666
N=15	1139.395	1416.903	1277.3863	4.2520	325.945	149.34902	6.141	892.87	423.03422
N=20	2057.069	2483.183	2255.03779	7.449	629.837	268.68597	10.909	1760.105	767.92409
N=25	3184.505	3838.465	3487.91597	11.663	1187.366	424.97716	16.906	3280.891	1211.50223

As it is shown in the Table 6.1, for the first-order methods, the times start shorter and they continues increasing as the prediction horizon is bigger. In addition, the variation between their minimum and maximum values are greater in BADMM and BAMA unlike the simplified IP which maintains its times due to the fact that the iterations barely change in these experimental results.

Furthermore, the first-order algorithms demonstrate that the number of iterations severely affects the time until the convergence since these iterations increase when the problem is bigger and therefore, the algorithm requires more time to find a solution. On the same way, BADMM formulation has worse times than the BAMA in all the prediction horizons tested.

On the other hand, another important aspect to take into account is the pattern of the iterations in each prediction horizon. This pattern is expected to be similar to that of the computational study.

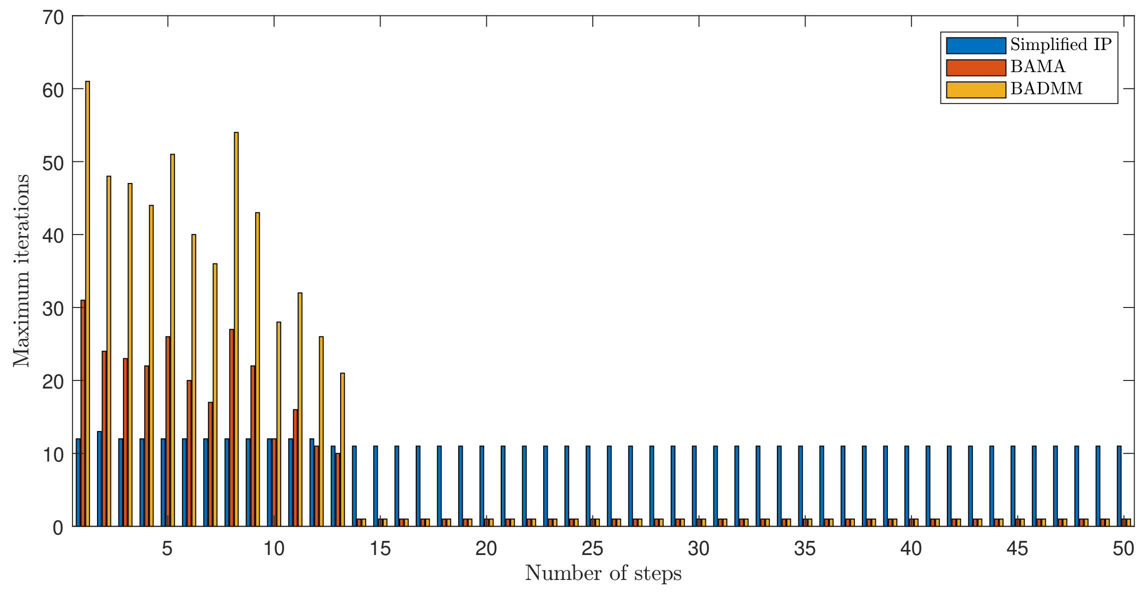


Figure 6.4: Experimental results in closed loop for $N = 5$

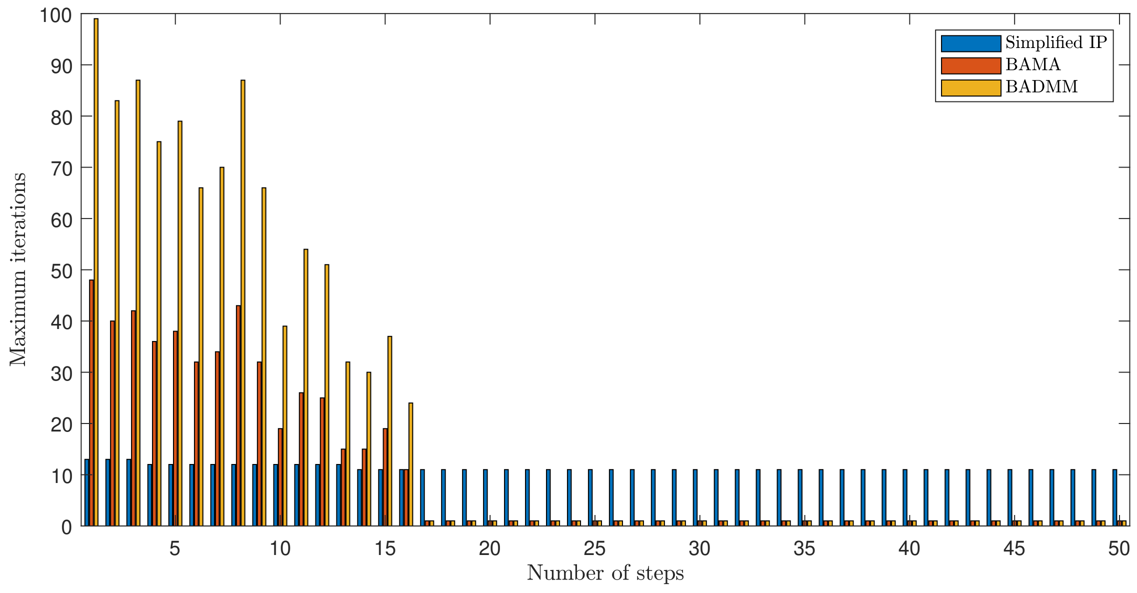


Figure 6.5: Experimental results in closed loop for $N = 10$

Based on results of iterations in Figure 6.4 - 6.8, they are similar to the results obtained in the computational study. Here, the first-order methods also have a part of the steps with a number of iterations equal to 1. This performance is different for the simplified IP since their iterations maintains a short variation range.

Furthermore, both in BAMA and BADMM, the number of iterations increases considerably as the prediction horizon is larger. This makes sense since the size of the QP problem is bigger and it gains complexity.

Therefore, it is expected that if the control objective is harder to maintain in a close

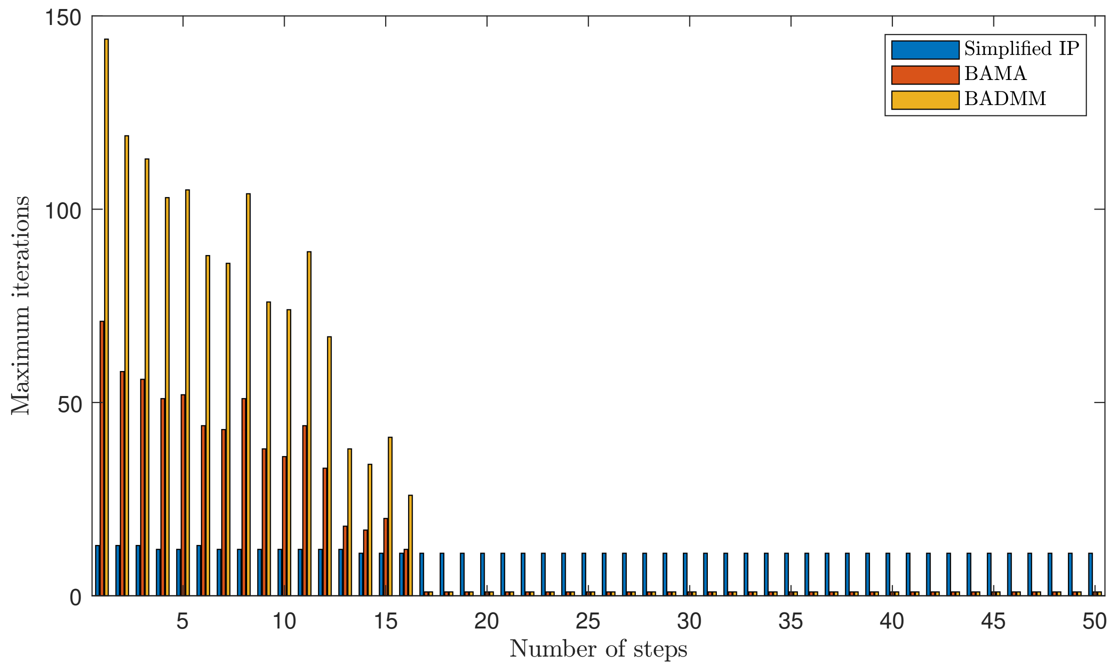


Figure 6.6: Experimental results in closed loop for $N = 15$

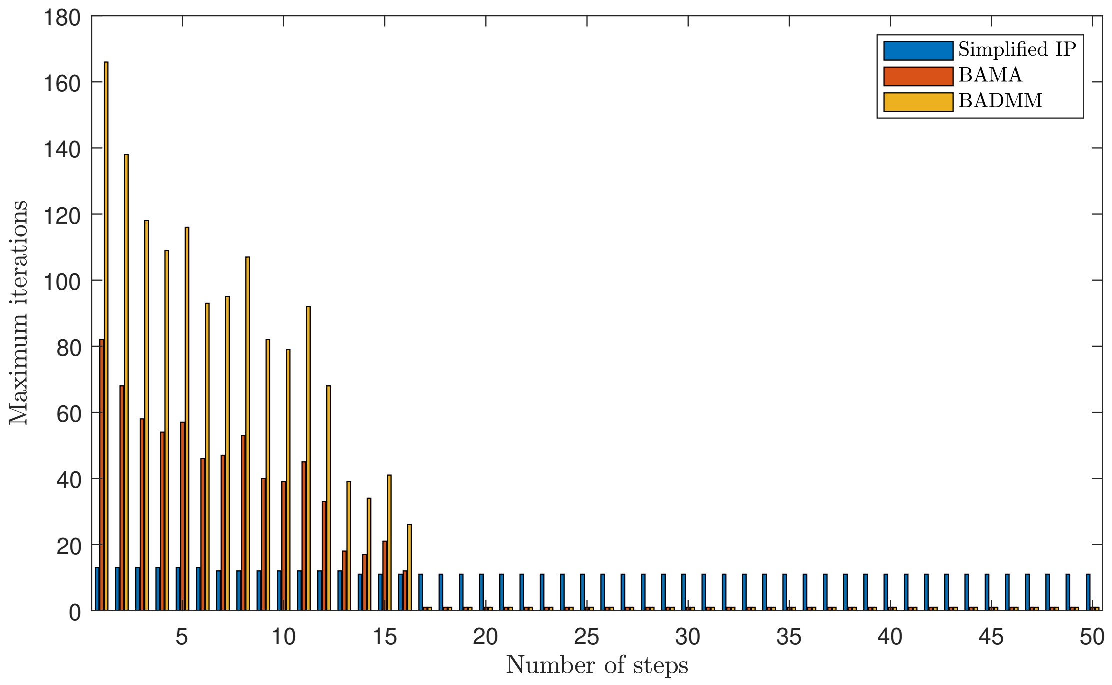


Figure 6.7: Experimental results in closed loop for $N = 20$

loop (with decision variables changing during all the control sequence), the number of iterations will increase much more and making that the possibility of using these controllers not the best choice.

Once presented the pattern of iterations for this control objective, in Figure 6.9,

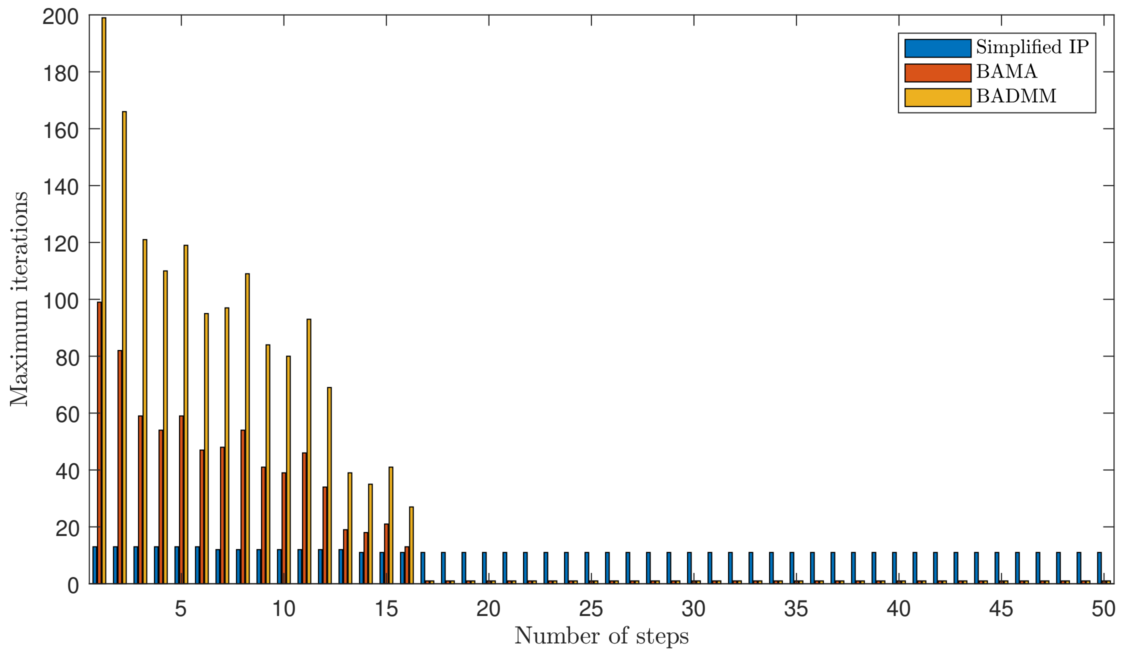


Figure 6.8: Experimental results in closed loop for $N = 25$

the maximum times are calculated. These values do not mean that the simplified IP have a bad performance since it is necessary to take into account the iterations of each algorithm. Despite that, these times gives more information about how the time is modified as the prediction horizon is larger. This variation occurs both first and second-order algorithms.

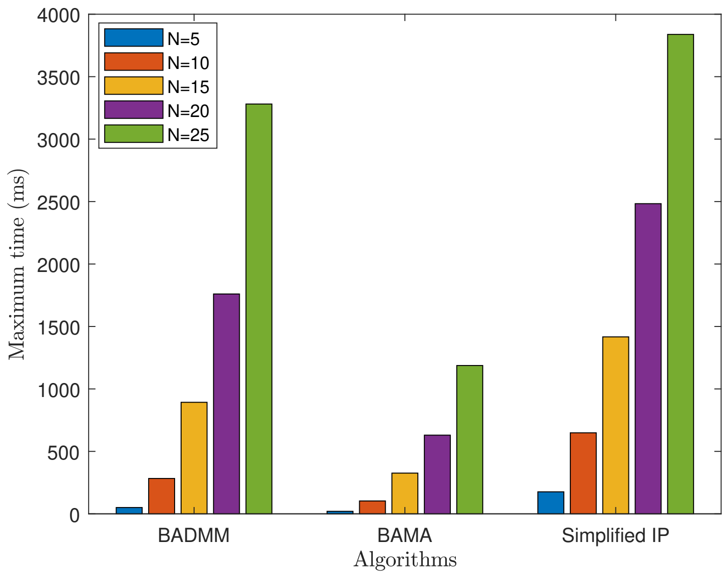
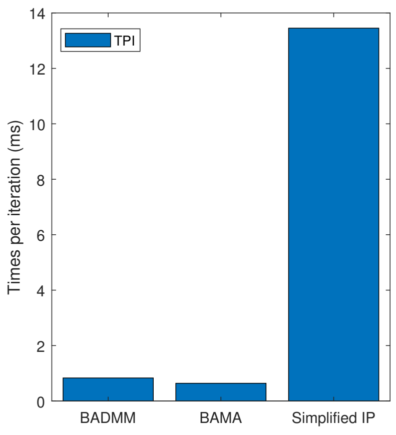
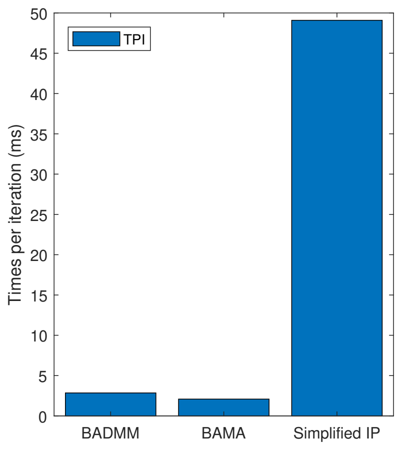


Figure 6.9: Maximum times for each prediction horizon

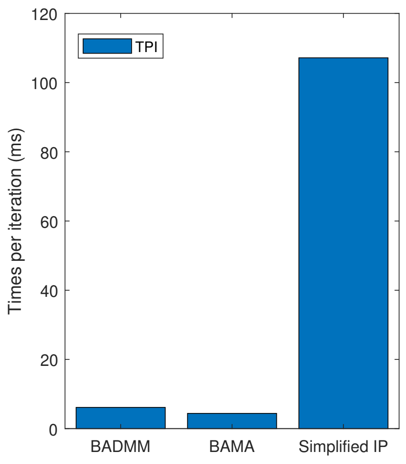
In addition, focused on developing the performance of all the algorithms, the times per iterations are shown for each prediction horizon in Figure 6.10.



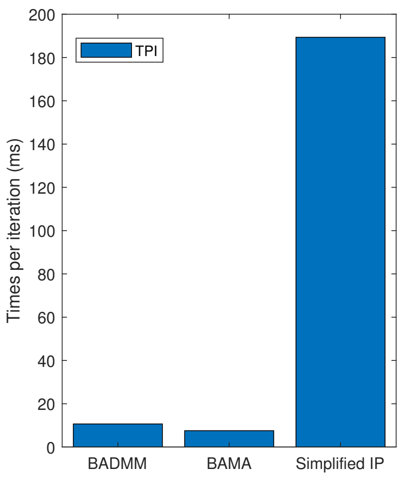
(a) $N = 5$



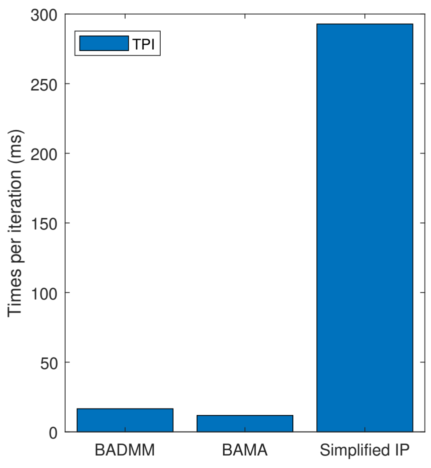
(b) $N = 10$



(c) $N = 15$



(d) $N = 20$



(e) $N = 25$

Figure 6.10: Average times per iteration for each prediction horizon

In this Figure 6.10, the first-order algorithms seem to operate better than the simplified IP; however, it is not a fair comparison since it is necessary to remember that there are steps which iterations are equal to 1. Thus, these results only show a performance with a smooth control objective.

On the other hand, in order to complete this analysis in terms of times until the convergence, total times are calculated in Figure 6.11. Here, the maximum, minimum and mean total times are obtained by multiplying the number of iterations and the time until the convergence of each algorithms for all the prediction horizons.

Based on these results, the banded ADMM has the worst total time of the comparison and simplified IP, the best performance. This is due to the number of iterations of the second-order method which iterations barely change.

6.1.1 Times in repeated variables

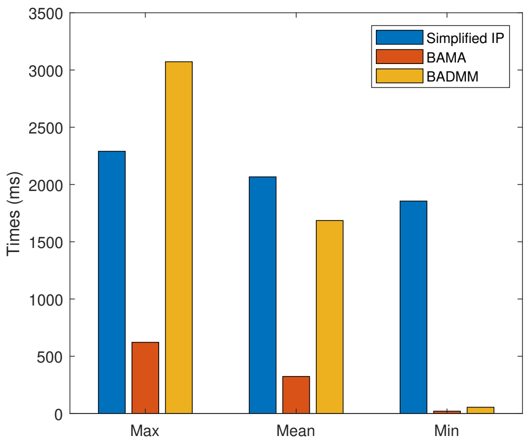
In order to identify the times in repeated variables which are part of the close loop simulations in each first-order algorithm, the maximum and minimum times are obtained. With these times in each prediction horizon, variables with a high computational burden will be chosen and analysed in order to study the possibility of a certain acceleration of them based on hardware description.

It is worth highlighting that in a closed loop simulation, there are two types of repeated variables. The first type is the variables which repetitions occur each step in the control loop, this means that their updates are needed each sampling time. The number of recalculations is defined by the number of steps selected for the control sequence.

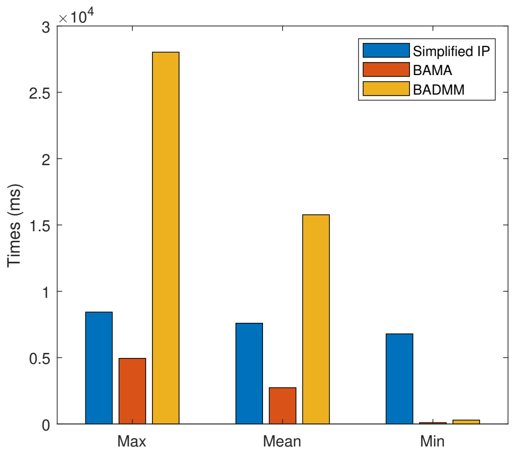
The second type of repeated variables are those which are recalculated in each iteration performed by the algorithm. The first-order algorithms have 3 main updates and two residual comparisons; thus, all of them are measured in this analysis.

Based on the results described in Table 6.2 - 6.6 for each prediction horizon, in the first prediction horizon $N = 5$, the operations which time is larger are the right-side of the inequality constraints matrix g and the particular solution z_0 for the control loop. These variable are repeated the number of steps selected for each sampling time. For the loop of the algorithm which calculation is repeated for each iteration performed, almost all the variables have a large computational burden; however, the first step corresponding to the banded linear solver has the largest time of the comparison. This makes sense since this step involves multiples operations unlike the rest of the variable inside the loop of the algorithm.

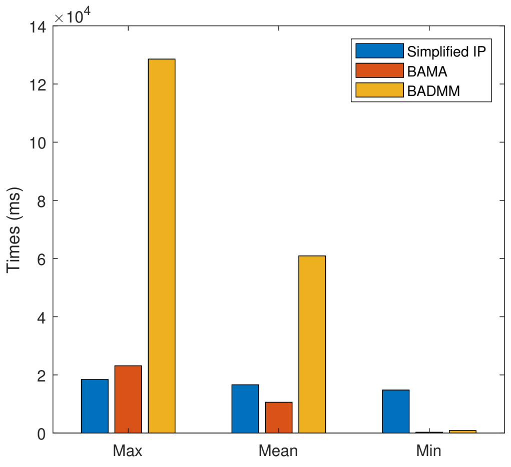
With respect to the prediction horizon equal to $N = 10$, the time taken by particular solution z_0 is notoriously larger than the time in $N = 5$. This is consistent with the theory since in this update, a linear system must be solved and thus, it requires more time to be calculated. Moreover, in the loop of the algorithm, the linear solver, the second update and the calculation of the dual residual have the highest times of the comparison.



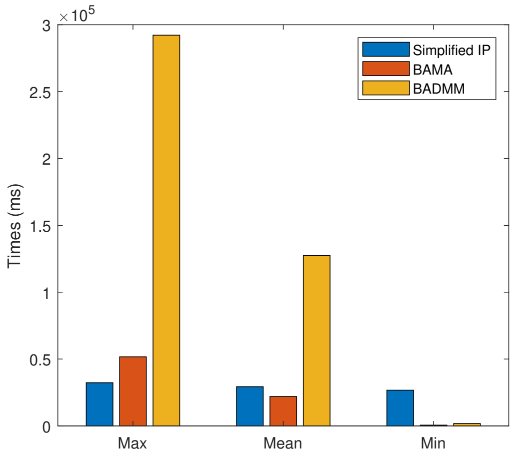
(a) $N = 5$



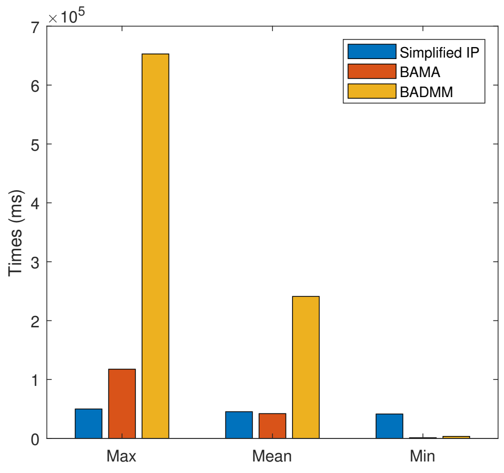
(b) $N = 10$



(c) $N = 15$



(d) $N = 20$



(e) $N = 25$

Figure 6.11: Total times for each prediction horizon

In the Table 6.4 corresponding to the prediction horizon $N = 15$, the calculation of the particular solution z_0 is the most computationally expensive of the control loop

Table 6.2: Resulting times in milliseconds of repeated variables in the closed loop simulations in $N = 5$

Times (ms)					
		BAMA		BADMM	
Variables	**Symbols**	**Min. value**	**Max. value**	**Min. value**	**Max. value**
Control Loop					
Calculation of right-side of equality constraints	f	0.0120	0.0530	0.0120	0.0370
Calculation of second term in quadratic function	h	0.1620	0.2470	0.1620	0.2150
Calculation of right-side of inequality constraints	g	0.3060	0.4270	0.3060	0.4100
Particular solution	z_0	2.4340	2.7240	2.4350	2.7180
Algorithm Loop					
First step (Linear solver)	z	0.1390	4.4740	0.2460	13.3540
Second step	y	0.1280	3.8020	0.1710	10.2540
Third step	τ	0.1120	3.5050	0.1120	6.8580
Calculation of primal and dual residuals	$\|r\|$	0.1010	3.1340	0.1010	6.2080
	$\|s\|$	0.1140	3.8940	0.2050	11.2430

Table 6.3: Resulting times in milliseconds of repeated variables in the closed loop simulations in $N = 10$

Times (ms)					
		BAMA		BADMM	
Variables	**Symbols**	**Min. value**	**Max. value**	**Min. value**	**Max. value**
Control Loop					
Calculation of right-side of equality constraints	f	0.0130	0.0470	0.0130	0.0420
Calculation of second term in quadratic function	h	0.5600	0.7220	0.5620	0.6890
Calculation of right-side of inequality constraints	g	1.0840	1.3070	1.0850	1.2820
Particular solution	z_0	14.3050	15.3930	14.3060	15.4920
Algorithm Loop					
First step (Linear solver)	z	0.4540	23.9000	0.8200	77.5180
Second step	y	0.4010	19.3490	0.5660	56.9080
Third step	τ	0.3740	18.3070	0.3770	38.0070
Calculation of primal and dual residuals	$\|r\|$	0.3530	17.1850	0.3540	35.5930
	$\|s\|$	0.3770	21.1140	0.7220	68.0590

since its time has increased considerably. Furthermore, the times for the loop of the algorithm continues increasing even though there are two variables which times are bigger: The linear solver and the calculation of the dual residual. They have the

Table 6.4: Resulting times in milliseconds of repeated variables in the closed loop simulations in $N = 15$

Times (ms)					
		BAMA		**BADMM**	
Variables	**Symbols**	**Min. value**	**Max. value**	**Min. value**	**Max. value**
Control Loop					
Calculation of right-side of equality constraints	f	0.0140	0.0530	0.0130	0.0630
Calculation of second term in quadratic function	h	1.2000	1.4770	1.2000	1.5830
Calculation of right-side of inequality constraints	g	2.3400	2.6860	2.3430	2.6070
Particular solution	z_0	44.1280	47.5140	44.1230	47.8270
Algorithm Loop					
First step (Linear solver)	z	0.9430	76.6830	1.7180	248.8090
Second step	y	0.8340	60.2660	1.1970	179.0750
Third step	τ	0.7970	58.1880	0.7980	118.1550
Calculation of primal and dual residuals	$\|r\|$	0.7650	55.8810	0.7650	113.0660
	$\|s\|$	0.7960	68.7350	1.5470	221.1170

Table 6.5: Resulting times in milliseconds of repeated variables in the closed loop simulations in $N = 20$

Times (ms)					
		BAMA		**BADMM**	
Variables	**Symbols**	**Min. value**	**Max. value**	**Min. value**	**Max. value**
Control Loop					
Calculation of right-side of equality constraints	f	0.0140	0.0570	0.0150	0.0520
Calculation of second term in quadratic function	h	2.1860	2.4780	2.1940	2.4760
Calculation of right-side of inequality constraints	g	4.1760	4.4900	4.1900	4.5530
Particular solution	z_0	102.6950	107.6010	102.6880	107.6940
Algorithm Loop					
First step (Linear solver)	z	1.6570	144.2250	3.0360	486.5770
Second step	y	1.4570	118.7810	2.1500	356.0440
Third step	τ	1.4070	115.7190	1.4090	234.8120
Calculation of primal and dual residuals	$\|r\|$	1.3670	112.4720	1.3680	227.9940
	$\|s\|$	1.4070	129.8080	2.7440	436.3960

largest time of the comparison but without much difference to the other variables in the loop of the algorithm.

Table 6.6: Resulting times in milliseconds of repeated variables in the closed loop simulations in $N = 25$

Times (ms)					
		BAMA		BADMM	
Variables	**Symbols**	**Min. value**	**Max. value**	**Min. value**	**Max. value**
Control Loop					
Calculation of right-side of equality constraints	f	0.0140	0.0640	0.0150	0.0700
Calculation of second term in quadratic function	h	3.3000	3.6570	3.3070	3.7270
Calculation of right-side of inequality constraints	g	6.4550	7.0110	6.4760	6.9720
Particular solution	z_0	188.9600	196.3140	188.9170	196.1920
Algorithm Loop					
First step (Linear solver)	z	2.6700	278.2050	4.7960	919.6880
Second step	y	2.2190	219.0270	3.2930	655.0300
Third step	τ	2.1620	214.7280	2.1460	428.4560
Calculation of primal and dual residuals	$\|r\|$	2.1110	209.9240	2.0960	418.5710
	$\|s\|$	2.2700	252.4510	4.3460	832.2770

As regards the prediction horizon $N = 25$, the particular solution is clearly the most computationally expensive operation of the control loop. In addition, the linear solver and the dual residual calculation have the highest time of the comparison in the loop of the algorithm. The relevance of the dual residual calculation can be better noted in the banded ADMM than the banded AMA.

In summary, taking into account the results and comments before mentioned, it is noted that in case of using an acceleration strategy, there are two approaches considering both control loop and the loop of the algorithm

In the first approach, the particular solution can be accelerated in order to reduce the time in the control loop which need to be calculated in each sampling time. There is not another variable which increases its number of iterations as the calculation of the particular solution in the process to redefine the QP problem.

In the second approach, corresponding to the loop of the algorithm, it could be exist two procedures to accelerate variables. In the first procedure, only the linear solver (first update) is accelerated since it is the most computationally expensive in this loop; however, the times for the rest of the variables also increases as the prediction horizon is bigger. Thus, the second procedure consists on developing an acceleration considering all the variable involved in the loop of the algorithm. Evidently, the complexity of the second procedure is quite substantial than the first procedure but these operations are calculated each number of iterations and they use to increase considerably when the complexity of the problem is larger.

6.2 Closed Loop with control objective modified

Once presented the performance for a smooth control objective (the equilibrium of the decision variables), it is necessary to change this control objective. Thus, the benchmark problem is modified on the same way that it is carried out during the computational study in section 5.4.

In this modified problem, the oscillating masses only are attached to a wall in a extreme and the inputs are located in the masses of the extremes. On this way, the structure of the problem is the same that the Figure 5.43.

As it is mentioned in the previous chapter, this control objective is useful to identify the main characteristics of the algorithms throughout the control loop and how the iterations change during the control as the initial state in each sampling time is re-defined by feedback.

The initial state is randomly generated in order to evaluate how stable are the performance of these algorithms starting from different states and how to develop the control throughout the control sequence. Therefore, C-code for 100 random initial states are generated for each prediction horizon in the embedded platform where the constraints are defined as $\|x_k\|_\infty \leq 3.5$ and $\|u_k\|_\infty \leq 0.5$.

An example to the resulting performance of the algorithms is presented in Figure 6.12 - 6.14.

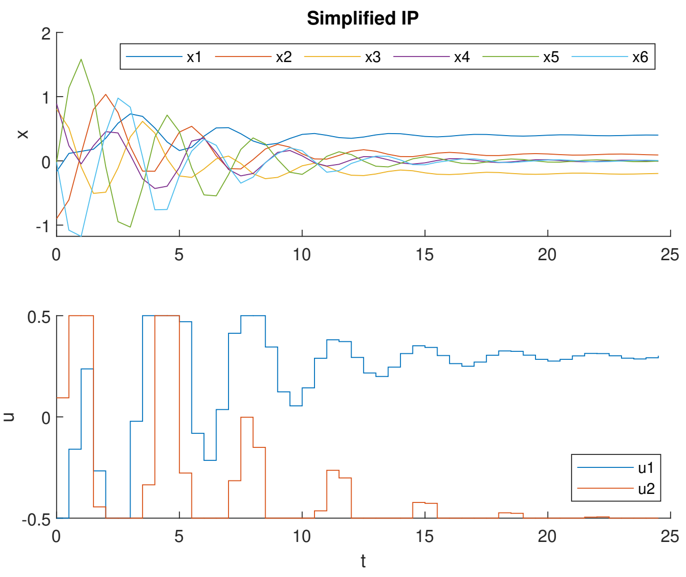


Figure 6.12: Experimental result in closed loop for simplified IP

In order to observe the results of the times until the convergence for each algorithm in all the simulations carried out in the embedded platform, the Table 6.7 is presented. These times were calculated as the time taken by the algorithm in finding the optimal solution in each sampling time. Based on these values, the minimum, maximum and mean times are obtained.

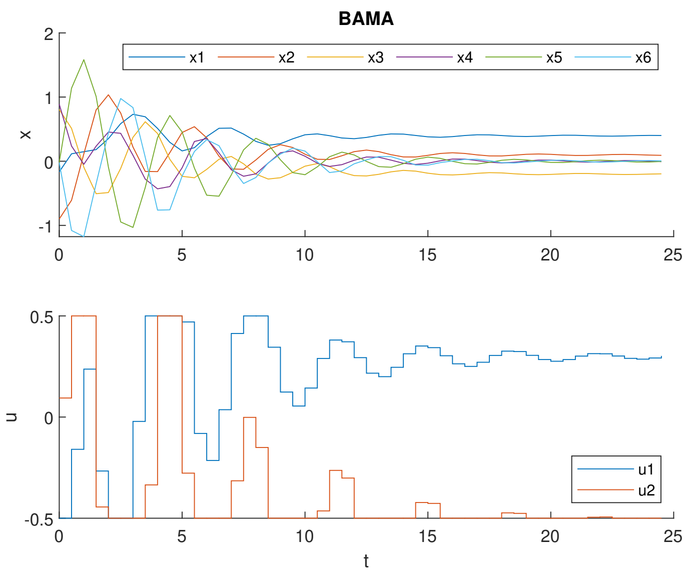


Figure 6.13: Experimental result in closed loop for BAMA

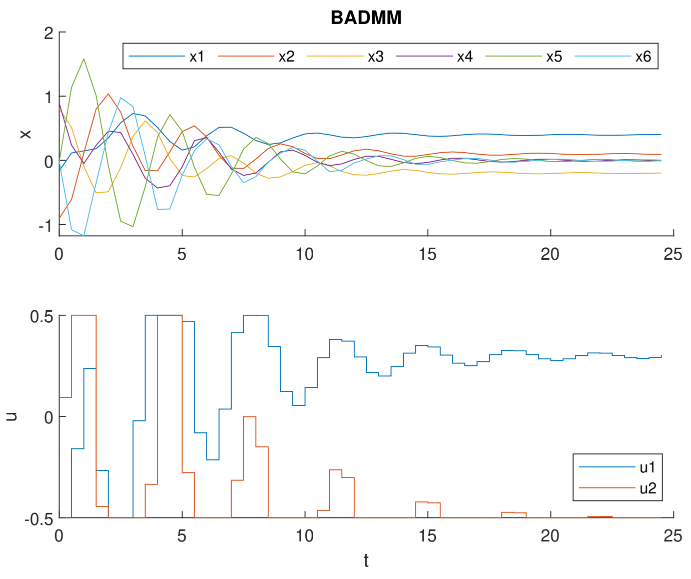


Figure 6.14: Experimental result in closed loop for BADMM

Table 6.7: Times in milliseconds of the resulting data considering all the simulations for the algorithms tested

	Simplified IP			**BAMA**			**BADMM**		
Prediction Horizon	Min. value	Max. value	Mean value	Min. value	Max. value	Mean value	Min. value	Max. value	Mean value
N=5	150.2060	186.8430	169.7600	0.6640	29.7230	18.7523	15.8630	73.4570	48.8654
N=10	553.8940	675.7010	634.2977	23.1900	270.4930	110.5584	77.4860	704.2990	300.4329
N=15	1332.395	1579.462	1411.53911	58.759	682.907	284.64952	187.51	1845.602	788.79878
N=20	2330.812	2764.839	2489.2330	121.613	1167.57	520.37895	392.966	3187.729	1461.72407
N=25	3610.824	4283.231	3873.83998	190.251	1837.568	832.03705	628.299	5071.281	2349.14295

According to the Table 6.7, minimum values for the first-order methods have increased due to the fact that the number of iterations are different to 1 over all the control sequence. This increment is obtained by the change of the control objective. Thus, these times could eve increase much more if the complexity of the problem is bigger than the current.

In addition, if the minimum values are largest, it makes sense that the maximum values also increase and on this way, the mean will be larger for this problem than for the control objective carried out in the previous section.

On the other hand, another important aspect to take into account is the pattern of the iterations in each prediction horizon with this control objective. This new pattern is calculated by selecting the maximum iterations in each step of the control loop for all the simulations, and it is expected to be similar to that of the computational study.

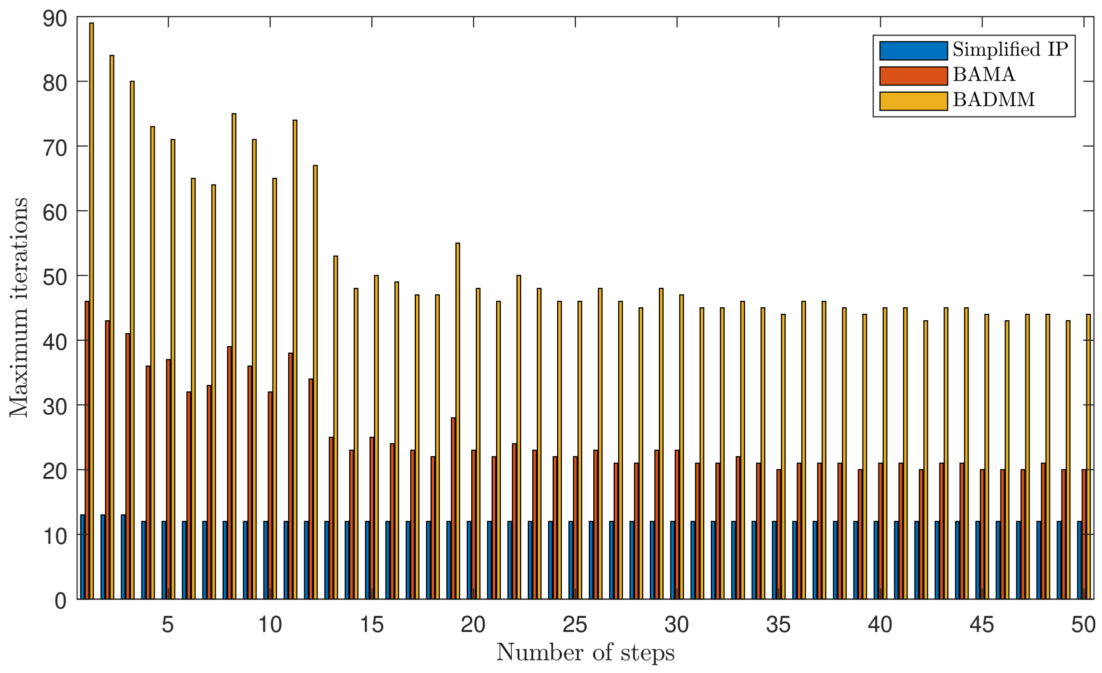


Figure 6.15: Experimental results in closed loop for $N = 5$

In Figure 6.15 - 6.19, patterns for all the prediction horizons are presented. According to these results, the number of iterations are different to 1 in all the control sequence and they increase as the prediction horizon is bigger for banded ADMM and banded AMA.

With respect to the first-order methods, the banded ADMM has the most number of iterations from $N = 5$ until $N = 25$. Even though the banded AMA has less iterations than banded ADMM, the simplified IP is selected as the best choice to solve these problems due to its stability with different initial states and QP problem sizes.

Therefore, it is demonstrate the stability of the second-order method in terms of iterations when the complexity of the problem increase due to prediction horizon, initial state or reference trajectory in the control objective.

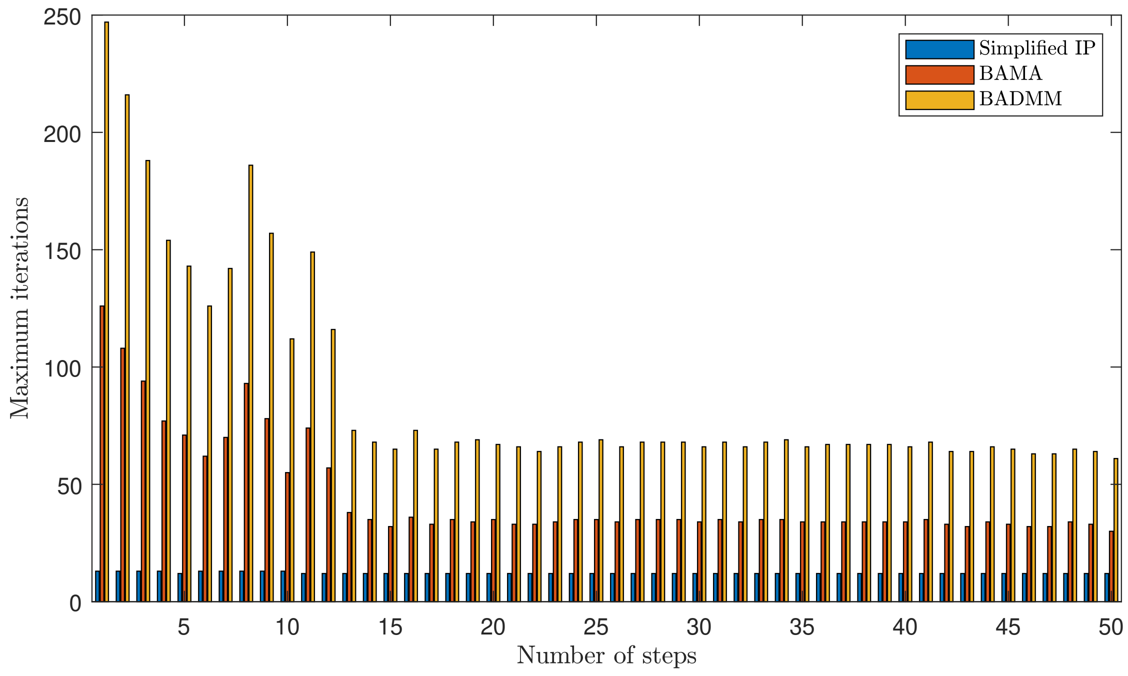


Figure 6.16: Experimental results in closed loop for $N = 10$

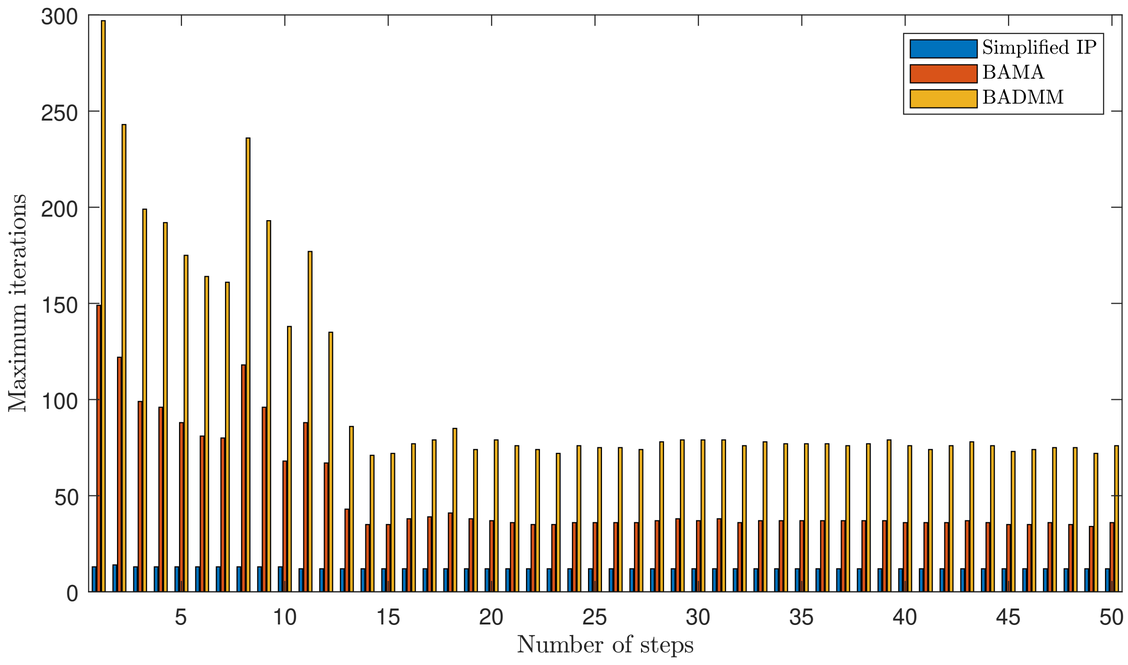


Figure 6.17: Experimental results in closed loop for $N = 15$

Once presented the pattern of iterations for this modified control objective, in Figure 6.20, the maximum times are calculated.

In addition, focused on obtaining all the information about the performance of all

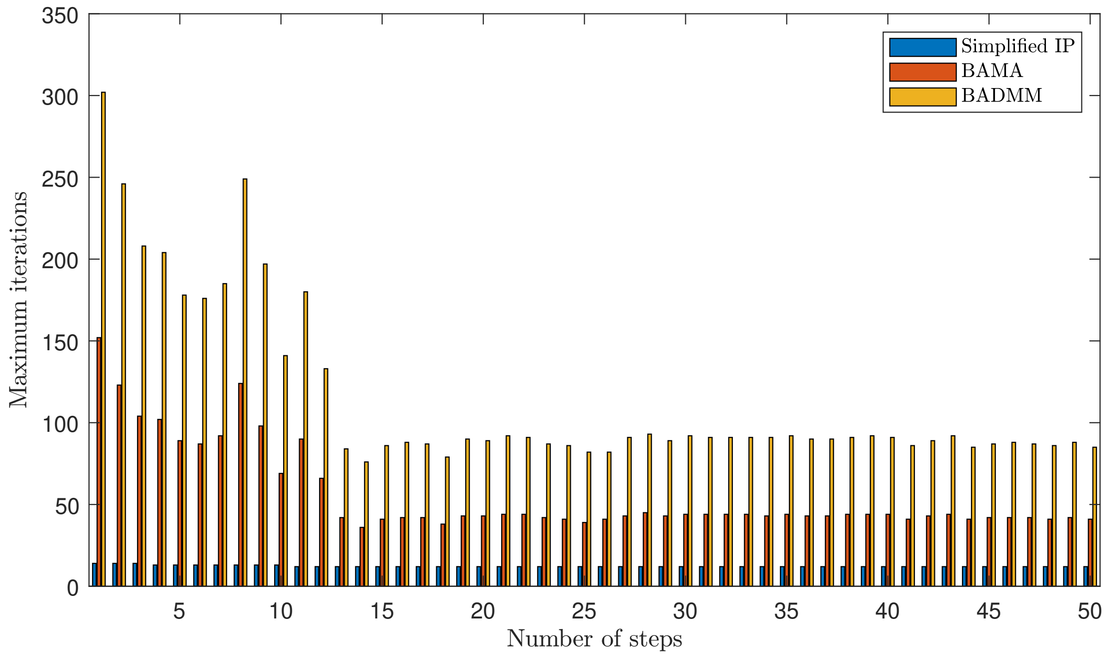


Figure 6.18: Experimental results in closed loop for $N = 20$

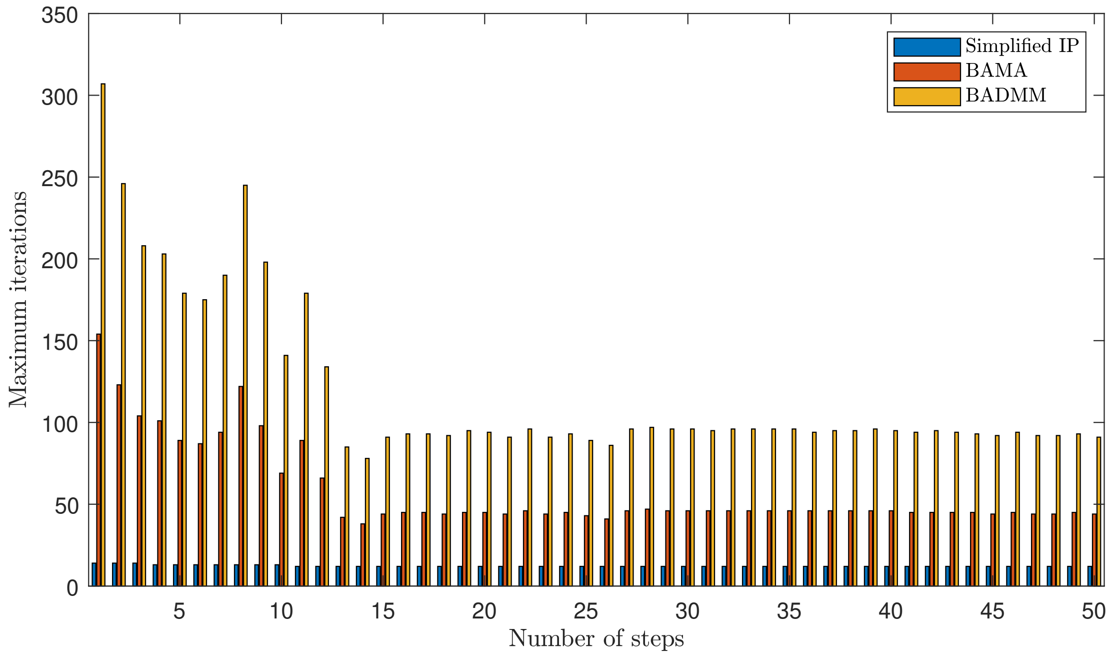


Figure 6.19: Experimental results in closed loop for $N = 25$

the algorithms, the times per iterations are shown for each prediction horizon in Figure 6.21.

In this Figure 6.21, the first-order algorithms seem to operate better than the simplified IP; however, it is not a fair comparison since it is necessary to take into account the number of iterations that the algorithms require to solve the QP problem.

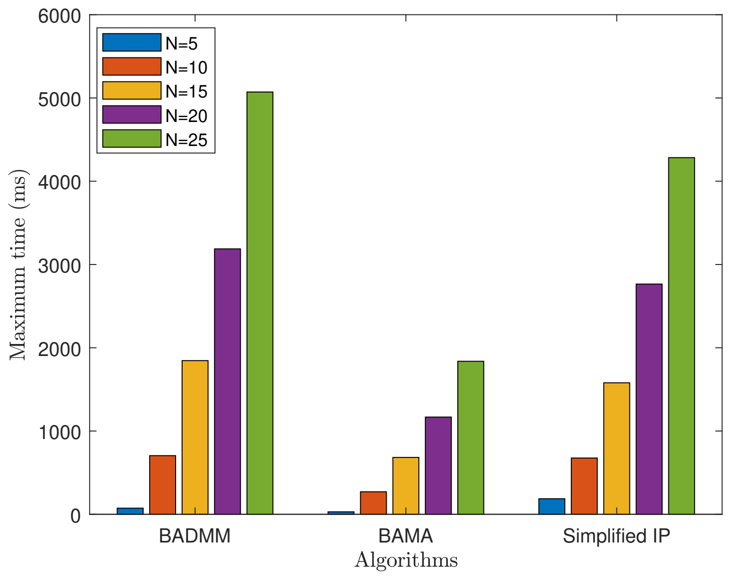


Figure 6.20: Maximum times for each prediction horizon

On the other hand, in order to complete this analysis in terms of times until the convergence, total times are calculated in Figure 6.22. Here, the maximum, minimum and mean total times are obtained by multiplying the number of iterations and the time until the convergence of each algorithms for all the prediction horizons.

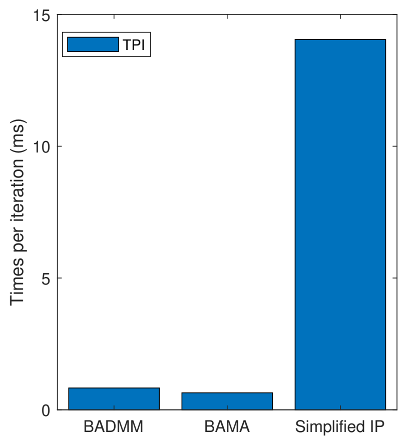
For the prediction horizon $N = 5$, the banded AMA results the best performance of the comparison in terms of total times. However, in the next prediction horizon ($N = 10$), the simplified IP emphasizes its stability since the first-order algorithms increase their iterations and therefore, their total times.

The same performance occurs for the rest of prediction horizons since the second-order method barely change it total time whereas the first-order algorithms continues increasing their times. As mentioned above, this notable increase is due to the number of iterations that can become excessive if the complexity of the problem is even greater in case the control objective is more restrictive and demanding.

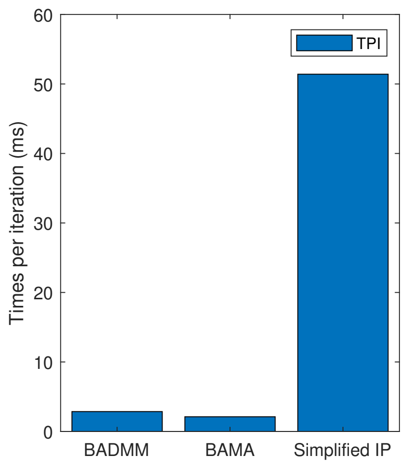
6.2.1 Times in repeated variables

In order to identify the times in repeated variables which are part of the close loop simulations in each first-order algorithm, the maximum and minimum times are obtained. With these times in each prediction horizon, variables with a high computational burden will be chosen and analysed in order to study the possibility of a certain acceleration of them based on hardware description.

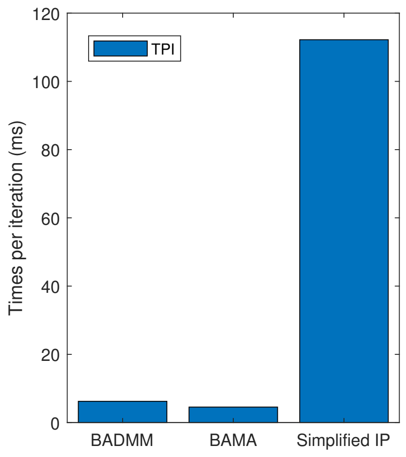
Unlike the resulting times for the smooth control objective, it is expected that the times for the control loop slightly change and the times for the loop of the algorithm increase considerably due to the rise of the complexity of the problem and the number of iterations.



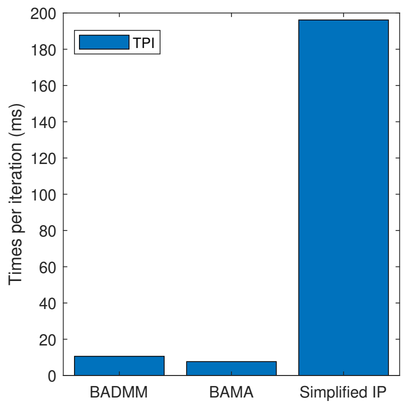
(a) $N = 5$



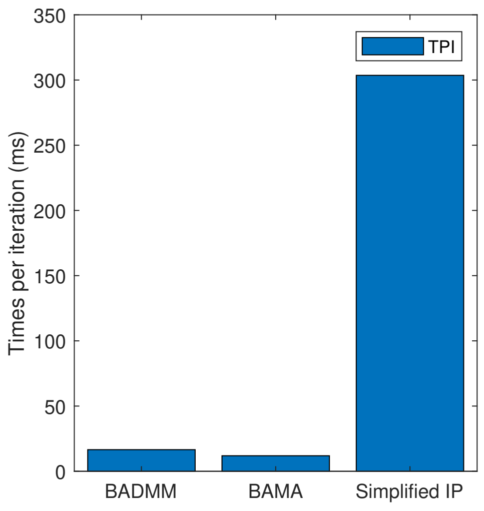
(b) $N = 10$



(c) $N = 15$

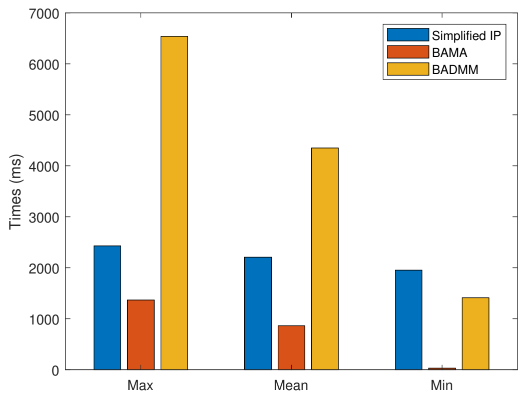


(d) $N = 20$

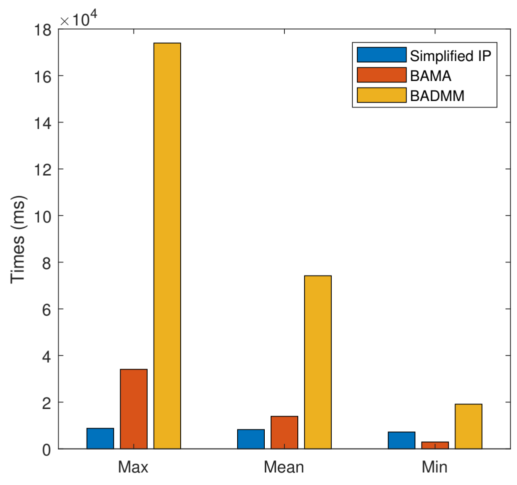


(e) $N = 25$

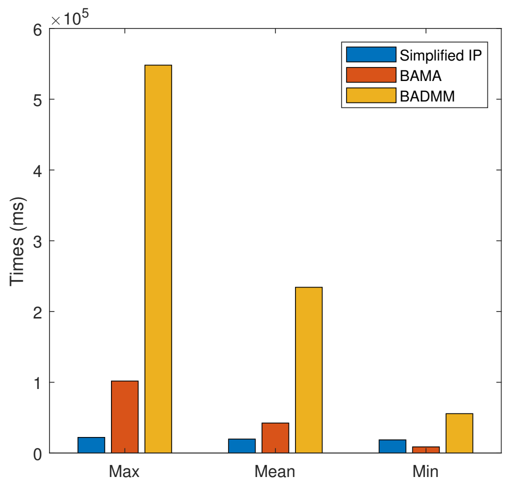
Figure 6.21: Times per iteration for each prediction horizon



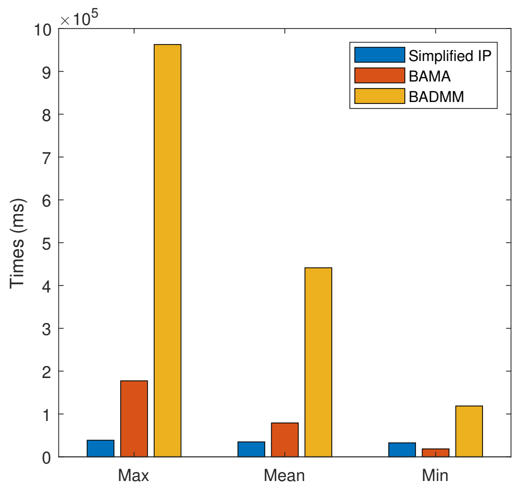
(a) $N = 5$



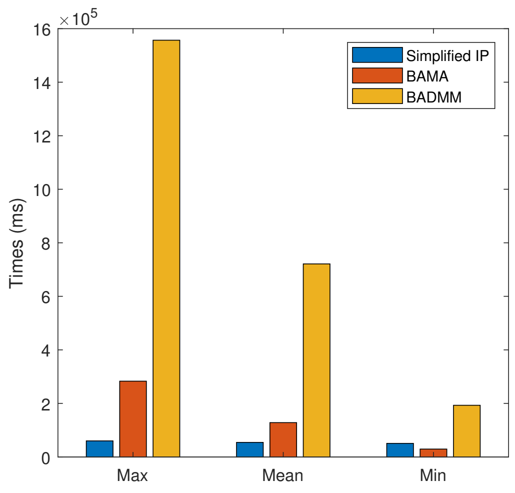
(b) $N = 10$



(c) $N = 15$



(d) $N = 20$



(e) $N = 25$

Figure 6.22: Total times for each prediction horizon

Table 6.8: Resulting times in milliseconds of repeated variables in the closed loop simulations in $N = 5$

Times (ms)					
		BAMA		BADMM	
Variables	**Symbols**	**Min. value**	**Max. value**	**Min. value**	**Max. value**
Control Loop					
Calculation of right-side of equality constraints	f	0.0130	0.0880	0.0130	0.0350
Calculation of second term in quadratic function	h	0.2190	0.2940	0.2200	0.2890
Calculation of right-side of inequality constraints	g	0.3180	0.3860	0.3170	0.4170
Particular solution	z_0	2.4380	2.6940	2.4380	2.7260
Algorithm Loop					
First step (Linear solver)	z	0.1430	6.5810	4.1720	19.4770
Second step	y	0.1290	5.6970	3.1910	14.9550
Third step	τ	0.1120	5.1530	2.1290	9.9990
Calculation of primal and dual residuals	$\|\|r\|\|$	0.1010	4.6700	1.9200	9.0140
	$\|\|s\|\|$	0.1140	5.7680	3.6130	16.4650

Table 6.9: Resulting times in milliseconds of repeated variables in the closed loop simulations in $N = 10$

Times (ms)					
		BAMA		BADMM	
Variables	**Symbols**	**Min. value**	**Max. value**	**Min. value**	**Max. value**
Control Loop					
Calculation of right-side of equality constraints	f	0.0130	0.0420	0.0130	0.0510
Calculation of second term in quadratic function	h	0.7410	0.8750	0.7420	0.9160
Calculation of right-side of inequality constraints	g	1.1040	1.2400	1.1080	1.2830
Particular solution	z_0	14.5190	15.5490	14.5200	15.6220
Algorithm Loop					
First step (Linear solver)	z	5.2350	62.7150	21.2360	193.9000
Second step	y	4.4660	50.7490	15.7190	143.7890
Third step	τ	4.1930	48.1960	10.4140	94.7880
Calculation of primal and dual residuals	$\|\|r\|\|$	3.9390	45.2340	9.6530	88.4750
	$\|\|s\|\|$	4.6170	55.4320	18.6790	167.3190

Table 6.10: Resulting times in milliseconds of repeated variables in the closed loop simulations in $N = 15$

Times (ms)					
		BAMA		BADMM	
Variables	**Symbols**	**Min. value**	**Max. value**	**Min. value**	**Max. value**
Control Loop					
Calculation of right-side of equality constraints	f	0.0140	0.1070	0.0140	0.0400
Calculation of second term in quadratic function	h	1.5710	1.8870	1.5700	1.9130
Calculation of right-side of inequality constraints	g	2.3890	2.6150	2.3980	2.6840
Particular solution	z_0	44.8030	47.9240	44.8030	47.7610
Algorithm Loop					
First step (Linear solver)	z	13.6360	161.6630	52.2110	516.0980
Second step	y	10.9530	125.3620	37.2660	368.7220
Third step	τ	10.5560	121.0820	24.6520	243.9220
Calculation of primal and dual residuals	$\|r\|$	10.1270	116.3820	23.5790	233.6880
	$\|s\|$	12.1940	145.4150	47.0560	456.9130

Table 6.11: Resulting times in milliseconds of repeated variables in the closed loop simulations in $N = 20$

Times (ms)					
		BAMA		BADMM	
Variables	**Symbols**	**Min. value**	**Max. value**	**Min. value**	**Max. value**
Control Loop					
Calculation of right-side of equality constraints	f	0.0150	0.0560	0.0150	0.0520
Calculation of second term in quadratic function	h	2.6880	3.0150	2.6860	3.0780
Calculation of right-side of inequality constraints	g	4.1780	4.5440	4.1910	4.5010
Particular solution	z_0	102.7080	108.5050	102.7110	107.7210
Algorithm Loop					
First step (Linear solver)	z	27.2980	266.4420	108.0320	880.3540
Second step	y	23.2100	220.5850	78.9560	643.6390
Third step	τ	22.6150	215.0540	52.4040	427.6200
Calculation of primal and dual residuals	$\|r\|$	21.9300	208.3760	50.8130	414.1320
	$\|s\|$	24.6380	240.5270	98.6540	788.7120

Based on the results described in Table 6.8 - 6.12 for each prediction horizon, the times calculated have increased especially in the loop of the algorithm.

Table 6.12: Resulting times in milliseconds of repeated variables in the closed loop simulations in $N = 25$

Times (ms)					
		BAMA		BADMM	
Variables	**Symbols**	**Min. value**	**Max. value**	**Min. value**	**Max. value**
Control Loop					
Calculation of right-side of equality constraints	f	0.0150	0.0620	0.0150	0.0560
Calculation of second term in quadratic function	h	4.1020	4.5040	4.1020	4.4900
Calculation of right-side of inequality constraints	g	6.4510	6.9580	6.4740	7.0140
Particular solution	z_0	188.8340	196.9660	188.8460	196.7860
Algorithm Loop					
First step (Linear solver)	z	44.417	432.137	175.752	1421.739
Second step	y	35.1180	337.9500	125.0280	1010.212
Third step	τ	34.3950	331.1830	82.3920	665.8390
Calculation of primal and dual residuals	$\|\|r\|\|$	33.5920	323.3200	80.4340	650.5140
	$\|\|s\|\|$	40.5570	392.9470	159.4520	1282.026

For a prediction horizon $N = 5$, the particular solution z_0 is the most computationally expensive calculation in the control loop. Furthermore, considering the loop of the algorithm, the first update corresponding to the solution of the linear system has the highest time of this comparison; however, the times of the rest of variables in this loop are close to the time of the first update.

With respect to the prediction horizon $N = 10$, the particular solution z_0 still has the highest time of the control loop. This makes sense since for this calculation, it is necessary to solve a linear system more complex as the prediction horizon is bigger. Moreover, for the loop of the algorithm, the first update and the dual residual are the largest times of this comparison.

As regard the prediction horizon $N = 15$, only the particular solution increase its time and it has the highest time of the control loop. Furthermore, the first update and the dual residual are the operations with most computational burden in the loop of the algorithm.

For the prediction horizons $N = 20$ and $N = 25$, the particular solution is clearly the variable which consumes more time in the control loop. Also, all the variables in the loop of the algorithm increase their times and only the linear solver is slightly more computationally expensive.

In summary, taking into account both the smooth and the modified control objective, the second-order methods have the best performance in terms of stability in iterations and behaviour when the parameters of the problems are stricter. This is not the case for the first-order algorithms which iterations increase considerably when the complexity of the system is larger.

In terms of first-order methods, the banded ADMM had demonstrated a worse performance that the banded AMA; however, it is important to take into account that the ADMM algorithm can be used in more problems than the AMA variants since ADMM does not require a strictly convex function.

As regards a future hardware acceleration, it is clear that it must be considered both the control and algorithm loop since the variables inside of these loop are repeated a certain number of times. It is evident that the acceleration should be focused on the loop of the algorithm since the variables are recalculated each iteration. Thus, considering the inefficient performance in terms of iterations of the first-order algorithms when the complexity of a problem is larger, it is necessary to reduce this time per iteration.

As it is summarized in the above section, the control loop could be accelerated considering only the linear solver (first update) or carrying out a pipelining of all the loop of the algorithm. This decision depends on the problem sizes, the horizon selected and the hardware resources.

Finally, in terms of the control loop, it is noted that there is only one operation needs to be accelerated which is the particular solution z_0 which is part of the reduced-Hessian approach to reduce the size of a original QP problem and to construct banded matrices.

Chapter 7

Conclusions

Taking into account the above-obtained results, reduced-Hessian method proves a increase in the performance in terms of time until the convergence due to the reduction of operations in the first update of the algorithm. In addition, this method allows to reduce the size of the original problem by deleting the states and creating a new QP.

Despite the AMA variant based-null space is better than the ADMM, the latter can work with a non-strictly convex problems whereas the main requirement to use AMA is the strongly convexity of the problem evaluated. This is a crucial advantage since ADMM could be used in all types of convex problems; in addition, reformulating the QP by using the reduced-Hessian method, the size and therefore, the complexity of the new problem is less to be solved by the algorithm.

On the other hand, comparing the first-order methods to the second-order algorithms (simplified IP, in this project), the latter ones demonstrate a high robustness along the variation of complexity of the problem evaluated. This is not the same performance for the first-order algorithms that increase considerably their total times when the complexity is intensified due to the increase of the prediction horizon, the number of decision variables, the constraints or the initial state. This behaviour of the second-order methods, in particular of the simplified IP demonstrates the stability of these algorithms throughout the variation of the initial conditions and they are chosen as the best option to solve a problem.

In addition, despite that the second-order methods have better performance in terms of total times, it is difficult to embed them on limited resources platforms and thus, it is still being researched. This problem does not occur for the first-order algorithms their internal updates are relatively easy to parallelize and to embed in comparison to the procedures of the second-order methods.

Repeated variables in the closed loop are identified in the experimental results. According to these results, the first step of the algorithm loop and the particular solution of the control loop have the most computational burden of the closed loop. The algorithm loop is recalculated in each iteration whereas the control loop is recalculated in each sampling time. This analysis is relevant since allows to organize which updates could be accelerated by hardware description. In addition, these results allows to define the maximum sampling time for the application.

Chapter 8

Bibliography

[Berry et al., 1985] Berry, M. W., Heath, M. T., Kaneko, I., Lawo, M., Plemmons, R. J., and Ward, R. C. (1985). An algorithm to compute a sparse basis of the null space. *Numerische Mathematik*, 47(4):483–504.

[Boyd et al., 2010] Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2010). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122.

[Dang et al., 2017] Dang, T. V., Ling, K. V., and Maciejowski, J. (2017). Banded null basis and admm for embedded mpc. *IFAC-PapersOnLine*, 50(1):13170 – 13175. 20th IFAC World Congress.

[Dang et al., 2015] Dang, T. V., Ling, K. V., and Maciejowski, J. M. (2015). Embedded admm-based qp solver for mpc with polytopic constraints. In *2015 European Control Conference (ECC)*, pages 3446–3451.

[Dang, 2018] Dang, V. T. (2018). *Efficient algorithms for embedded optimisation-based control*. PhD thesis.

[del Rio Ruiz and Basterretxea, 2019] del Rio Ruiz, A. and Basterretxea, K. (2019). Towards the automatic implementation of reduced-size and high throughput mpc on fpgas. pages 1768–1773.

[Ghadimi et al., 2015] Ghadimi, E., Teixeira, A., Shames, I., and Johansson, M. (2015). Optimal parameter selection for the alternating direction method of multipliers (admm): Quadratic problems. *IEEE Transactions on Automatic Control*, 60(3):644–658.

[Gilbert and Heath, 1987] Gilbert, J. R. and Heath, M. T. (1987). Computing a sparse basis for the null space. *SIAM Journal on Algebraic Discrete Methods*, 8(3):446–459.

[Goldstein et al., 2014] Goldstein, T., O'Donoghue, B., Setzer, S., and Baraniuk, R. (2014). Fast alternating direction optimization methods. *SIAM J. Imaging Sciences*, 7:1588–1623.

[Goodall and Donoghue, 1993] Goodall, R. M. and Donoghue, B. (1993). Very high sample rate digital filters using the d operator.

[Hartley et al., 2012] Hartley, E. N., Jerez, J. L., Suardi, A., Maciejowski, J. M., Kerrigan, E. C., and Constantinides, G. A. (2012). Predictive control of a boeing 747 aircraft using an fpga. *IFAC Proceedings Volumes*, 45(17):80 – 85. 4th IFAC Conference on Nonlinear Model Predictive Control.

[Jerez et al., 2013] Jerez, J., Goulart, P., Richter, S., Constantinides, G., Kerrigan, E., and Morari, M. (2013). Embedded online optimization for model predictive control at megahertz rates. *IEEE Transactions on Automatic Control*, 59.

[Jerez et al., 2011] Jerez, J. L., Constantinides, G. A., and Kerrigan, E. C. (2011). An fpga implementation of a sparse quadratic programming solver for constrained predictive control. In *Proceedings of the 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '11, page 209–218, New York, NY, USA. Association for Computing Machinery.

[Kaneko et al., 1982] Kaneko, I., Lawo, M., and Thierauf, G. (1982). On computational procedures for the force method. *International Journal for Numerical Methods in Engineering*, 18(10):1469–1495.

[Liu et al., 2014] Liu, J., Peyrl, H., Burg, A. P., and Constantinides, G. A. (2014). Fpga implementation of an interior point method for high-speed model predictive control. *2014 24th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–8.

[Maciejowski, 2002] Maciejowski, J. (2002). *Predictive Control: With Constraints*. Prentice Hall.

[McInerney et al., 2018] McInerney, I., Constantinides, G. A., and Kerrigan, E. C. (2018). A survey of the implementation of linear model predictive control on fpgas. *IFAC-PapersOnLine*, 51(20):381 – 387. 6th IFAC Conference on Nonlinear Model Predictive Control NMPC 2018.

[Parikh and Boyd, 2013] Parikh, N. and Boyd, S. (2013). Proximal algorithms. *Found. Trends Optim.*, 1(3):127–239.

[Pu, 2016] Pu, Y. (2016). Splitting methods for distributed optimization and control. page 117.

[Raghunathan and Di Cairano, 2014] Raghunathan, A. and Di Cairano, S. (2014). Alternating direction method of multipliers for strictly convex quadratic programs: Optimal parameter selection. *Proceedings of the American Control Conference*, pages 4324–4329.

[Raghunathan and Cairano, 2015] Raghunathan, A. U. and Cairano, S. D. (2015). Admm for convex quadratic programs: Linear convergence and infeasibility detection. *arXiv: Optimization and Control*.

[Shukla et al., 2017] Shukla, H. A., Khusainov, B., Kerrigan, E. C., and Jones, C. N. (2017). Software and hardware code generation for predictive control using splitting methods. *IFAC-PapersOnLine*, 50(1):14386 – 14391. 20th IFAC World Congress.

[Stathopoulos et al., 2016] Stathopoulos, G., Shukla, H., Szucs, A., Pu, Y., and Jones, C. N. (2016). Operator splitting methods in control. *Foundations and Trends in Systems and Control*, 3(3):249–362.

[Topcu, 1979] Topcu, A. (1979). *A contribution to the systematic analysis of finite element structures using the force method.* PhD thesis, University of Essen. In German.

Appendix A

Scalability results

Table A.1: Results for $N = 5$

$N = 5$			
Algorithms	**Average Time (ms)**	**Average Iterations**	**Average time per iteration (TPI)**
AMA IND	6.0732	101.2100	0.0600
ADMM IND	0.7075	11.6600	0.0607
DAMA	0.1730	10.0700	0.0172
DADMM	0.3541	20.1200	0.0176
BAMA	0.2384	13.5500	0.0176
BADMM	0.4336	25.5700	0.0170
FDAMA	0.2801	16.2400	0.0172
FDADMM	0.1704	8.8400	0.0193
FBAMA	0.3723	21.8000	0.0171
FBADMM	0.1972	10.4300	0.0189

Table A.2: Results for $N = 10$

$N = 10$			
Algorithms	**Average Time (ms)**	**Average Iterations**	**Average time per iteration (TPI)**
AMA IND	2.5754	118.6700	0.1828
ADMM IND	2.5754	13.9800	0.1842
DAMA	0.7746	12.3100	0.0629
DADMM	1.7145	24.7000	0.0694
BAMA	1.0408	17.4300	0.0597
BADMM	2.2662	33.0700	0.0685
FDAMA	1.3139	19.9500	0.0659
FDADMM	0.5486	9.9800	0.0550
FBAMA	1.7026	27.5400	0.0618
FBADMM	0.6503	12.2800	0.0530

Table A.3: Results for $N = 15$

$N = 15$			
Algorithms	**Average Time (ms)**	**Average Iterations**	**Average time per iteration (TPI)**
AMA IND	45.4834	130.5900	0.3483
ADMM IND	5.3644	14.6900	0.3652
DAMA	1.8568	13.8600	0.1340
DADMM	4.0034	27.7300	0.1444
BAMA	2.5747	20.6000	0.1250
BADMM	5.3466	39.0000	0.1371
FDAMA	2.6821	20.3500	0.1318
FDADMM	1.0722	10.6600	0.1006
FBAMA	3.8741	30.8300	0.1257
FBADMM	1.3045	13.8200	0.0944

Table A.4: Results for $N = 20$

$N = 20$			
Algorithms	**Average Time (ms)**	**Average Iterations**	**Average time per iteration (TPI)**
AMA IND	82.7745	150.7300	0.5492
ADMM IND	9.1205	15.7200	0.5802
DAMA	2.7490	14.4500	0.1902
DADMM	6.2445	29.1700	0.2141
BAMA	3.4358	19.5100	0.1761
BADMM	7.3631	36.9800	0.1991
FDAMA	4.3066	23.0100	0.1872
FDADMM	1.5419	11.6300	0.1326
FBAMA	5.4606	30.9200	0.1766
FBADMM	1.7688	14.0000	0.1263

Table A.5: Results for $N = 25$

$N = 25$			
Algorithms	**Average Time (ms)**	**Average Iterations**	**Average time per iteration (TPI)**
AMA IND	323.7722	149.9800	2.1588
ADMM IND	34.2211	14.9300	2.2921
DAMA	4.1739	14.3600	0.2907
DADMM	8.5983	28.9200	0.2973
BAMA	4.6186	19.8000	0.2333
BADMM	9.6636	37.4900	0.2578
FDAMA	6.2268	22.5200	0.2765
FDADMM	2.2989	11.4000	0.2017
FBAMA	7.0856	31.0600	0.2281
FBADMM	2.3968	14.5600	0.1646