Efficient Real-Time Solutions for Nonlinear Model Predictive Control with Applications

A thesis submitted to The University of Sheffield for the degree of Doctor of Philosophy

Oscar Julian Gonzalez Villarreal

Department of Automatic Control and Systems Engineering

August 2021

Acknowledgements

I would like to begin by thanking CONACyT, Mexico for supporting me throughout both, my M.Sc. and Ph.D. studies since October 2015. This has been nothing but a life time experience and I am extremely grateful that I was given this opportunity. I hope that I will be able to contribute to the development of Science and Technology in my country for years to come.

Secondly, I would like to express my deep and sincere gratitude to my supervisor; Dr. Anthony Rossiter, for his selfless dedication of time and effort towards my development as a researcher in this field, as well as for the countless hours of talks and discussions, not only of academic-related endeavours but of life-time goals and objectives which shaped myself towards becoming a better researcher, and a better person.

I would also like to express my great appreciation to my second supervisor; Dr. Bryn Jones, who helped me develop both my technical and theoretical skills, as well as for supporting my research throughout.

Additionally, I would like to express my complete gratitude to my previous supervisors; Dr. Antonio Favela from my alma-mater ITESM Campus Monterrey, Mexico for introducing me to Control System Theory in a way I will never forget, and for consistently mentoring me with rich discussions about my life goals; and Dr. Hyo-Sang Shin from Cranfield University, UK, for his selfless support and mentoring throughout my M.Sc. studies.

I am particularly grateful for the help provided by Ph.D. Student Juan Guerrero-Fernandez, with whom countless discussions and collaborations resulted in an important publication related to my Ph.D. work, and whose companion, friendship and influence as a dedicated researcher helped me overcome some of my own limitations.

I would also like to give my special thanks to the Technical Staff of the department, in particular: Technical Experimental Officer Craig Bacon, technician Anthony Whelpton, and technician Jack Powell, for their support towards the development of various experiments, and whose companionship and friendship helped me enjoy my work beyond my expectations.

I wish to thank all my students, both from laboratories ACS213/ACS330/ACS6502, as well as from the VTOL Project, in particular: Stefan Stoian, Akinola Alexander Dada, Wayel Kamal Eldeen, Johan K. Mcilwaine, and James A. Farnworth, who allowed me to support them throughout their respective thesis/works which ultimately helped me to become a better teacher, and a better mentor.

I also would like to thank all my family and friends, both from/in UK and Mexico, who helped me enjoy my relatively lonely time in the UK as well as supported me throughout difficult times which ultimately helped me overcome my own demons.

Finally, I would like to thank the two persons in the whole world without whom I wouldn't be here; my parents, for their selfless efforts and support towards my development, not only as a researcher, but as a person. This thesis is dedicated to them.

Thank you mom, and thank you dad for not giving up on me!

Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

A list of the variables and notation used in this thesis is defined below. The definitions and conventions set here will be observed throughout unless otherwise stated. For a list of acronyms, please consult page xvi.

$\bar{\lambda}_{act}$	Set of Lagrange Multipliers related to active-constraints
$\bar{U}$	Nominal/Guessed Input Vector
$\bar{X}$	Nominal/Guessed State Vector
$\bar{X}_u$	Nominal/Guessed Input-related Constraints Vector
$\bar{X}_x$	Nominal/Guessed State-related Constraints Vector
$\bar{Y}$	Nominal/Guessed Output Vector
$\delta\hat{C}$	Predicted Input-Correction Deviation Vector
$\delta\hat{U}$	Predicted Input Deviation Vector
$\delta\hat{X}$	Predicted Predicted State Deviation Vector
$\delta\hat{Y}$	Predicted Output Deviation Vector
δx_0	Initial State Offset
γ	Constraints Vector
$\hat{U}$	Predicted Input Vector
$\hat{X}$	Predicted State Vector
$\hat{Y}$	Predicted Output Vector
λ	Lagrange Multipliers
$\mathbb{N}$	General Input-Parameterisation Matrix
D	State-Offset Multiple-Shooting Vector
E	Hessian

$E_{\mathbb{N}}$	General Input-Parameterised Hessian
F	Input-to-Input Dual Mode Multiple-Shooting Matrix
f	Linear Term/Gradient
$f_{\mathbb{N}}$	General Input-Parameterised Linear Term/Gradient
G	State-to-State Prediction Matrix
H	Input-to-State Prediction Matrix
J	Cost Function
M	Constraints Matrix
$M_{\mathbb{N}}$	General Input-Parameterised Constraints Matrix
N_b	MWB Block Size
N_E	Hessian Size
N_p	Prediction Horizon
N_u	Control Horizon
n_u	Number of Inputs
n_x	Number of States
n_y	Number of Outputs
n_z	Number of decision variables
n_γ	Number of constraints
n_{act}	Number of active-set constraints
N_{b_0}	Virtual Block Position Indiator
N_{b_i}	Inner Block Position Index
P_N	Terminal Weight
P_N^i	Terminal Weight for i_{th} block position
Q	Output or State Weighting Matrix
R	Input Weighting Matrix
S	Input-Offset Dual Mode Multiple-Shooting Vector
W	State-to-Input Dual Mode Multiple-Shooting Matrix

Acronyms

CARIMA Controlled Auto-Regressive Integrated Moving Average. 57

DARE Discrete Algebraic Riccati Equation. 50

EKF Extended Kalman Filter. 185

F-DoF Full-Degrees of Freedom. 83

FPGA Field Programmable Gate Arrays. 2

FTC Fault Tolerant Control. 3

GPC Generalised Predictive Control. 60

IFT Implicit Function Theorem. 24

IVE Initial Value Embedding. 63

MAS Maximum Admissible Set. 51

MCU Micro-Controlling Unit. 3

MWB Moving Window Blocking. 83

NLP Nonlinear Programming Problem. 21

NMPC Nonlinear Model Predictive Control. 2

NMSS Non-Minimal State Space. 57

OCP Optimal Control Problem. 20

ODE Ordinary Differential Equation. 22

OSI OSI. 185

PFC Predictive Functional Control. 3

PWM Pulse Width Modulated. 186

QP Quadratic Program. 19

RK Runge-Kutta. 22

RLS Recursive Least Squares. 73

RTI Real-Time Iteration. 4

SQP Sequential Quadratic Program. 21

UAV Unmanned Aerial Vehicle. 3

USV Unmanned Surface Vehicle. 60

VTOL Vertical Take-Off and Landing. 3

WEC Wave Energy Converter. 60

Abstract

Nonlinear Model Predictive Control is an advanced optimisation methodology widely used for developing optimal Feedback Control Systems that use mathematical models of dynamical systems to predict and optimise their future performance. Its popularity comes from its general ability to handle a wide range of challenges present when developing control systems such as input/output constraints, complex nonlinear dynamics multi-variable systems, dynamic systems with significant delays as well as handling of uncertainty, disturbances and fault-tolerance.

One of the main and most important challenges is the computational burden associated with the optimisation, particularly when attempting to implement the underlying methods in fast/real-time systems. To tackle this, recent research has been focused on developing efficient real-time solutions or strategies that could be used to overcome this problem. In this case, efficiency may come in various different ways from mathematical simplifications, to fast optimisation solvers, special algorithms and hardware, as well as tailored auto-generated coding tool-kits which help to make an efficient overall implementation of these type of approaches.

This thesis addresses this fundamental problem by proposing a wide variety of methods that could serve as alternatives from which the final user can choose from depending on the requirements specific to the application. The proposed approaches focus specifically of developing efficient real-time NMPC methods which have a significantly reduced computational burden whilst preserving desirable properties of standard NMPC such as nominal stability, recursive feasibility guarantees, good performance, as well as adequate numeric conditioning for their use in platforms with reduced numeric precision such as "floats" subject to certain conditions being met.

One of the specific aims of this work is to obtain faster solutions than the popular ACADO toolkit, in particular when using condensing-based NMPC solutions under the Real-Time Iteration Scheme, considered for all practical purposes the state-of-the-art standard real-time solution to which all the approaches will be bench-marked against. Moreover, part of the work of this thesis uses the concept of "auto-generation" for developing similar tool-kits that apply the proposed approaches. To achieve this, the developed tool-kits were supported by the Eigen 3 library which were observed to result in even better computation times than the ACADO toolkit.

Finally, although the work undertaking by this thesis does not look into robust control approaches, the developed methods could be used for improving the performance of the underlying "online" optimisation, eg. by being able to perform additional iterations of the underlying SQP optimisation, as well as be used in common robust frameworks where multi-model systems must be simultaneously optimised in real-time. Thus, future work will look into merging the proposed methods with other existing strategies to give an even wider range of alternatives to the final user.

Chapter 1

Introduction

Nonlinear Model Predictive Control (NMPC) is an advanced optimisation methodology widely used for developing optimal Feedback Control Systems that use mathematical models of dynamical systems to predict and optimise their future performance. Over the recent years it has been gaining an increasing amount of attention and has been the topic of a large amount of research with applications in a wide range of disciplines ranging from chemical [9, 16, 75, 95, 129] to aerospace [57, 81, 89, 93, 98, 107, 137], electronics [156], energy management [90], as well as the multi-disciplinary mechatronics robotics and autonomous systems [104, 113, 135, 142, 157]. Its popularity comes from its general ability to handle a wide range of challenges present when developing control systems such as input/output constraints, complex nonlinear dynamics, multi-variable systems, dynamic systems with significant delays, as well as handling of uncertainty, disturbances and fault-tolerance [26, 42, 68, 71, 81, 82, 87, 132].

Until now, one of the main and most important challenges since its development in the 1960s has been that the underlying optimisation methods typically come with a significant computational burden, particularly when implementing them in an "online" fashion, which makes their application to real-time systems, eg. systems with very fast sampling times, a real challenge. However, given the developments and improvements in the computational power available in general electronics over the last two decades, its application to fast real-time systems is now looking more feasible [66]. A popular alternative to tackle the computational burden are explicit solutions [7, 37, 93, 135] which aim to solve the optimisation in an "offline" manner. Nonetheless, it is known that offline solutions are typically limited to small-state dimensions [37, 135] which makes their implementation in complex systems restricted. Moreover, there may be situations or systems in which the system has a large amount of "external" variability which may not be able to be captured effectively using an "offline" solution. In these scenarios, the optimisation can be shown to benefit dramatically from adjusting the predictions using the latest real-time information. An example of this will be given in the case studies of sections 5.4 and 5.6 from this thesis. As a result, recent research has been focused the development of efficient real-time solutions or strategies that could be used to overcome the limitations of explicit "offline" solutions [37]. In this case efficiency may come in various different ways from mathematical simplifications [21, 152], to fast optimisation solvers [37, 38, 40], tailored coding based on auto-generated tool-kits [66, 73], use of special hardware such as Field Programmable Gate Arrays (FPGA) [135], as well as hot-starting techniques [31, 55], all of which may improve the online performance significantly.

1.1 Motivation

Coming from a background in Unmanned Aerial Vehicles (UAVs) where the systems are typically restricted in terms of computational power, and having applied simple Linear MPC techniques based on the popular Predictive Functional Control (PFC) methods to Quadrotors during M.Sc. studies, one of the first research directions of this Ph.D. was on the topic of Fast NMPC for UAVs, particularly with the objective of focusing in Fault Tolerant Control (FTC) NMPC methods for the Vertical Take-Off and Landing (VTOL) UAV seen in (https://youtu.be/hC0i_kusMKI). These types of vehicles have an inherent redundancy both in its actuators and flight modes which allows for a potential recovery in case where various faults happen in the system. Moreover, the development of research in the area of Fault Tolerant Control for UAVs was considered extremely important and relevant for the safety of the comunity. However, due to various reasons, including real-time limitations in the existing NMPC frameworks and solutions, this direction was only pursued for a brief initial period during the first year of research of this Ph.D. which resulted in the UKACC 2018 publication of an Adaptive Laguerre-based MPC for Attitude Stabilisation of Quadrotor [49] which served as a "fast" auto-tuning method giving excellent performance as seen in (https://youtu.be/RSe35TjjBPI).

At a certain point it was realised that some of the existing state-of-the-art NMPC methods still lacked the computational efficiency to be implemented in low cost Micro-Controlling Units (MCUs) as the ones used by common UAVs such as simple Arduinos, Pixhawk, etc., operating at a reduced frequency of $16 - 160$ MHz., with significantly limited memory as well as lacking proper floating-point units. Indeed, most of the research papers dealing with Fast NMPC solutions obtained the "micro-second" performance [65, 66] in the commodity of a laptop running Intel CPUs @ 3 GHz, which by no means reflect the actual condition in which these type of systems operate. As an example, the work from [107] presented an impressive trajectory optimisation and tracking NMPC framework where an Hexacopter is able to execute precise way-point navigation including going through windows tilted at 30°, at the cost of requiring the use of an expensive on-board laptop running an Intel i7 CPU. This extra weight (or more generally speaking, requirements) could be avoided if simpler or more efficient solutions could achieve similar performance, thus enabling the implementation of advanced solutions using already existing hardware in these systems whilst avoiding the extra weight requirements. In our work, it was found that the standard NMPC solution of the simple Quadrotor Attitude Stabilisation problem of case study from section 4.3 was not computationally feasible when implementing it on the Beaglebone Blue [13] platform; an embedded platform specifically designed for robotics running @ 1 GHz. Although a simplified solution could be obtained by changing some of the parameters of the optimisation and using "inner" control systems as in [154], it was considered relevant to be able to obtain alternative solutions which wouldn't require such type of adjustments and would allow the user to have more control on the desired objectives of the optimisation. This motivated the research direction of this Ph.D. into the development of theoretical frameworks which would allow faster solutions with reduced memory requirements whilst preserving desirable properties such as nominal stability of the optimisation.

Remark 1.1. *This thesis uses the notion of "Nominal Stability" which coincides with the usual notion of convergence or attractivity, ie.* $x_k \to 0$ *for* $k \to \infty$ *as defined in [28, 31]. It does not imply asymptotic stability to the origin in the sense of Lyapunov.*

On the other hand, part of the case studies used in this thesis were based on the popular Inverted Pendulum systems, widely used in academia for their inherent complexity. The interest in this particular type of systems was motivated mainly due to the author's previous knowledge and experience developing Inverted Pendulum systems, eg. as the one seen in (https://youtu.be/_buMNF5MPYE).

These kind of systems (Inverted Pendulums) were well known to be key for the development of rocket guidance and navigation systems back in the 1970s [155], although they were typically analysed or developed only in linear (near equilibrium) conditions, which for all practical purposes satisfied the requirements of common rocket trajectories. Nowadays, researchers seem to have regained interest into more complex systems such as the impressive Triple Inverted Pendulum [45] seen in (https://www.youtube.com/watch?v=cyN-CRNrb3E). More recently, the well known SpaceX company, developing re-usable rockets, have attempted a "flip" manoeuvre as a landing procedure for its "Starship x9" prototype (visible in min 1:44 of https://www.youtube.com/watch?v=_qwLHlVjRyw). What is most interesting is the surprisingly similar dynamic evolution of the aforementioned manoeuvre to that of Nonlinear Inverted Pendulum's trajectories. Indeed, the author of this thesis considers such type systems to have fascinating dynamics and properties, and therefore were considered relevant as case studies for chapters 5, 6 and 8. Consequently, part of the experimental research work of this Ph.D. resulted in the experimental application of the novel NMPC method of chapter 6 into the Double Parallel Inverted Pendulum visible in (https://youtu.be/7E-SXi3YKQo) which formed part of the main contribution of the IET Control Theory and Applications 2020 journal [47].

1.2 Aims and Objectives

Based on the aforementioned motivations, as well as the literature review presented in chapters 2 and 3, the main objective of this Ph.D. thesis is to develop a set of efficient real-time NMPC methods which have a significantly reduced computational burden whilst preserving some of the desirable properties of standard NMPC such as stability, recursive feasibility and good performance. The developed methods could serve as alternatives which fix potential problems present in the standard NMPC solutions. The user can then select from the proposed methods depending on the requirements of the application.

One of the specific aims of this work is to obtain faster solutions than the ACADO toolkit [66], in particular when using condensing-based solutions under the Real-Time Iteration (RTI) Scheme, considered for all practical purposes the state-of-the-art standard solution as discussed in chapter 3. Moreover, part of the work of this thesis looks into the concept of "auto-generation" for developing tool-kits which apply the proposed approaches. In our particular case, the developed toolkits were based on the Eigen 3 library [15] which resulted in better computation times than the ACADO toolkit.

On the other hand, the work undertaken by this thesis focuses specifically on developing methods that allow the reduction of the computational burden, as well as preserve the desired properties for the optimisation, and does not look into other implications such as robust control or explicit solutions which are typically considered "offline" tasks that can be tackled separately. The developed approaches could be used for improving the performance of the underlying "online" optimisation methods, eg. by being able to perform additional iterations, and could also be relevant in the context of robust control specifically when used for solving multi-model optimisations [20, 136] frameworks in real-time.

Finally, although there was a preference for Inverted Pendulum systems, the thesis does not focus on a specific system for the selected case studies, and instead will use various types of systems to discuss the relevant properties, advantages or disadvantages of the developed methods in various contexts.

1.3 Thesis Layout and Outcome

This thesis is organised as follows: Chapter 2 presents a comprehensive literature review regarding the main topic of efficient real-time solutions for NMPC along with various topics related to existing variants and applications of NMPC. Chapter 3 introduces a "detailed" technical background, provided as part of the literature review process including all the mathematical notation, algorithms and general methodologies to be used throughout this thesis. Chapter 4 introduces two optimisation frameworks for obtaining Input-Parameterised NMPC solutions along with a set of algorithms which allow an efficient implementation of the proposed approaches using the RTI Scheme, and a set of 3 case studies which demonstrate its relevant properties, advantages and disadvantages. Chapter 5 proposes the Shifting Strategy; a key contribution of this thesis which resulted in the publications of [48, 50, 59]. The developed method allows a significant reduction of the computational burden of the optimisation using an efficient and systematic methodology that could be relevant for implementation in real-time control systems. Moreover, the chapter comes along with an extension of the $O(N)/O(N^2)$ algorithms from Ph.D. thesis [8], which are key for obtaining an efficient implementation, and includes a set of algorithms required for its implementation using the RTI framework, thus forming one of the main contributions of this Ph.D. Moreover, the chapter contains a set of 4 case studies which demonstrate the various properties, advantages and disadvantages of the proposed approach. On the other hand, chapter 6 proposes a Closed-Loop "generic" pre-stabilisation methodology for obtaining a novel Dual-Mode NMPC approach that solves the problem of numeric conditioning present in the standard condensing-based NMPC frameworks. The proposed approach comes along with an extension the $O(N)/O(N^2)$ algorithms from Ph.D. thesis [8], which are key for obtaining an efficient implementation of the overall approach, and includes a set of algorithms required for implementing the proposed approach using the RTI frameworks, forming one of the secondary contributions of this Ph.D. The chapter uses 4 case studies to demonstrate the various properties, advantages and disadvantages of the proposed approach, including the aforementioned experimental work on the Double Inverted Pendulum. Afterwards, chapter 7 presents an infinite horizon solution supporting the method introduced in chapter 5 which allows one to obtain a rigorous nominal stability guarantee. The chapter comes along with an algorithm for its implementation, as well as 1 case study which is used to demonstrate the validity of the proposed approach. Finally, chapter 8 introduces "The Combined Approach"; a method combining the methods of chapters 4, 5, 6 and 7 which inherits the relevant advantages (and in some cases disadvantages) of the methods proposed in all the other chapters into one final "combined" approach, forming the final contribution of this thesis. The chapter comes along with further extensions of the $O(N)/O(N^2)$ algorithms from Ph.D. thesis [8], as well as 2 case studies which demonstrate the expected properties, advantages and disadvantages inherited from the other methods. The thesis ends with a summary presented in chapter 9 along with conclusions detailing the relevance of the research work undertaken by this Ph.D. thesis, as well as future work plans which may serve as possible extensions to the proposed approaches.

The work of this thesis resulted in the following publications and submissions:

- A 2018 IFAC NMPC Conference (Abstract-only) Paper [48]: "A Time-Varying Shifting Strategy for Block Based MPC Solutions using a RTI Scheme", related to the contents of chapter 5.
- A 2018 UKACC Conference Paper [49]: "Laguerre-based Adaptive MPC for Attitude Stabilisation of Quadrotor", related to the contents of chapter 4 experimentally applied to the physical Quadrotor visible in (https://youtu.be/RSe35TjjBPI).
- A 2020 IET Control Theory and Applications journal paper [50]: "Shifting Strategy for Efficient Block-based Non-linear Model Predictive Control using Real-Time Iterations", related to the contents of chapter 5.
- A 2020 IET Control Theory and Applications journal paper [47]: "Fast Hybrid Dual-Mode NMPC for a Parallel Double Inverted Pendulum with Experimental Validation", related to the contents of chapter 6 based on the experimental application of the proposed approach visible in (https://youtu.be/7E-SXi3YKQo).
- A 2020 IFAC World Congress publication [59]: "Model Predictive Control for Wave Energy Converters: A Moving Window Blocking Approach", related to the contents of chapter 5.
- A 2020 IFAC World Congress publication [100]: "Towards Control of Autonomous Surface Vehicles in Rough Seas", related to the contents of chapter 4.
- A 2020 IEEE Transactions on Automatic Control submission: "Dual Mode Stable Prediction Models for Numerically Robust Fast Nonlinear Model Predictive Control using Real-Time Iterations", related to the contents of chapter 6.

All of the aforementioned articles are provided in appendix D of this thesis.

1.4 Notation

Finally, to clarify the use of various expressions used throughout the thesis, we provide the following notation section which enlists the most relevant notation. Other notation specific to the method/chapter will be introduced were relevant. Moreover, specific nomenclature and acronyms used throughout the thesis are given in the "Nomenclature" and "Acronyms" sections provided in the preface of the thesis.

- $\hat{x}_{k+1}$: The $\hat{x}$ notation means "predicted value", and is used widely by the NMPC prediction models. As an example, $\hat{x}_{k+1}$ is the predicted value of x at future time step $k+1$.
- $\bar{x}_{k+1}$: The $\bar{x}$ notation means "nominal" value. Used for the nominal values/guesses required for the predicted trajectories of NMPC.
- $\hat{x}_{k+1|k}$: Predicted value of $\hat{x}_{k+1}$, calculated/predicted/obtained at time step k. This notation is only used strictly when necessary.

- $blkdiag([x_1, x_2, \cdots, x_N])$: Block-diagonal matrix formed with the values of $[x_1, x_2, \cdots, x_N]$.
- $diag([x_1, x_2, \cdots, x_N])$: Diagonal matrix formed with the values of $[x_1, x_2, \cdots, x_N]$.
- $\dot{x}/\ddot{x}$: First and Second derivatives of x w.r.t. time.
- $\tilde{x}^{[i]}$: Used to mathematically represent the value of a recursion on the i_{th} iteration, particularly used for definition of algorithms, eg. $\tilde{x}^{[1]} = A^T\tilde{x}^{[0]}$ being the value of $\tilde{x}^{[1]}$ obtained recursively from the previous value $\tilde{x}^{[0]}$.
- x^+: Alternative way of defining recursions, eg. as in the Newton-step $x^+ = x^- + \Delta x$.
- $\hat{x}^*_{k+1}$: Optimal predicted value of x_{k+1}
- $||x||$: Euclidean norm of x
- $||x||^2_Q$: Weighted squares of x, ie. $||x||^2_Q = x^TQx$
- $(x)_{k,j}$: k_{th} row/j_{th} column element of (x).
- $\hat{u}^i_k$: i_{th} input at time-step k. Used particularly in chapter 4. Also applies to weights, eg. r^i_k being weight of i_{th} input at time-step k. And applies to parameterisation matrix, eg. $\mathbb{N}^i_k$ being the parameterisation of the i_{th} input at time-step k. In some cases is used without specifying time step, eg. R^i being the weights of the i_{th} input.
- $\lceil x \rceil$: x rounded to infinity (ceil operation in Matlab). Used particularly in chapter 5.
- $\forall i = [1 \rightarrow N_p]$: For $i = [1, 2, 3, \cdots, N_p]$

Chapter 2

Literature Review

Given the complexity of some of the methods required by standard Nonlinear Model Predictive Control (NMPC) methodologies, it was decided to separate the revision of the available literature into non-technical and technical chapters. This will allow the reader to capture both the general picture, as well as the specific details required to implement the approaches proposed in this thesis. Moreover, each chapter of the contributions of this thesis contains its own introduction and discussion around the motivations behind the proposed methods, along with discussions of relevant research works specifically related to the contents of the chapter. As a result, some of the contents presented throughout both technical and non-technical literature review chapters will be inevitably repeated or referred to in later chapters, allowing to improve the recall experience for the reader.

Thus, the thesis will begin with a brief non-technical discussion of general NMPC methods provided in this chapter, followed by a rather detailed, more technical background provided in the following chapter (chapter 3), both of which will provide foundation for all the proposed methods of this thesis. The chapter will focus predominantly on existing concepts and methods related (or relevant) to efficient real-time NMPC, as well as provide overview of the current state-of-the-art approaches in the field. It is assumed here that the reader is familiar with topics such as state space modelling, Gauss-Newton methods, optimisation, as well as general control theory.

2.1 Nonlinear Model Predictive Control

As discussed earlier, Nonlinear Model Predictive Control (NMPC) is an advanced optimisation-based control methodology that repeatedly uses an inner model of the system to predict and optimise its future behaviour with respect to a set of objectives, costs, constraints and/or functions specified by the user [19, 21, 55, 146, 154]. Once the optimisation is solved, the decision variables, ie. the first set of controls or inputs of the system are implemented and the optimisation is solved again in the next sampling time. This is typically referred to as the "Receding Horizon" strategy [12, 99]. Some of the main advantages it has are its ability to explicitly handle constrained multi-variable systems with delays, open loop unstable and non-minimum phase nonlinear dynamics. This makes it of great interest for industry applications where a linear controller would perform poorly, eg. when the system has highly nonlinear dynamics in the vicinity of the operating point or when the system is required to operate in a wider range of operating points and/or subject to sudden changes/disturbances that can

drive the system far away from its operation point and close to their boundaries leading to significant performance degradation [12, 70, 81, 154]. However, given its non-convex optimisation-based nature, it comes with a increased computational cost where at every sampling time the system must solve a non-linear non-convex optimisation that can result in large computation times, especially on systems with reduced computational resources [154]. Because of this, it was originally applied mostly in systems with slow dynamics such as petrochemical plants where sampling times of minutes or even hours were used [12, 129, 154]. Nevertheless, due to the recent progress made in the computation power of embedded platforms, algorithms and software implementations, applications with short sampling times such as in manufacturing, aerospace systems, robotics and autonomous vehicles are now looking more feasible [81, 154]. As an example, works presented in [25, 55, 65, 66, 84, 107, 114, 115, 153, 154] report applications where timings in the mili- and micro timescales were possible.

Although these works indicate that NMPC can be now be implemented in the kilo-Hertz frequency ranges [65, 84], its implementation has been mostly investigated in systems with relatively large computational resources such as laptops or PC with fast CPUs and increased RAM memory size (see [1, 55, 61, 65, 73, 83] for a few examples). As an example, the addition of a small laptop to a medium scale aerial robotic system was presented in [74, 107]. These type of hardware differ from embedded platforms mainly in the support of vectorised instructions allowing parallel computing of specific operations [154], CPU frequency range and support of multiple-cores. As a result, its implementation on platforms with reduced computational resources such as the popular Arduino remains to be a challenge. Moreover, the use of open-source hardware, already available in this type of fast real-time systems is naturally a simpler option. This motivates the assessment and further development of these control techniques in available off-the-shelf hardware such as the recently developed Beaglebone Blue [13], the Xilinx Zynq [154], the famous Raspberry PI, as well as the popular Field Programmable Gate Arrays (FPGA) [101, 135] which would allow its rapid deployment and integration in industry applications. However, given their even more reduced computational resources, efficient solutions must be employed to guarantee real-time feasibility of the optimisation [127, 154].

2.2 Modelling

Given that NMPC relies on a mathematical model to predict the future behaviour of the system, modelling is an extremely important matter. According to [19, 70] there are three main types of non-linear models that have been employed in the NMPC framework. The first one is based on the fundamental models which are usually obtained by applying first-principles theory such as mass, energy and momentum balances resulting in ordinary differential equations (ODE) some times coupled through algebraic equations [70]. Within these types of models, authors from [114, 115] emphasise the possibility of representing most systems with a frequently occurring model structure containing one or more of the three subsystems in the specific order detailed in their work, namely linear input systems, nonlinear systems and linear output systems. The second type of non-linear models are those using input-output measurements attempting to replicate the dynamics of the system through some arbitrary mathematical representation, typically referred to as empirical models [19, 70]. Many types of empirical models exist in the literature such as polynomial Auto-Regressive Moving Average (ARMA) polynomials, Hammerstein models, Volterra models, Weiner models [19, 70] , Non-linear

Auto-Regressive Moving Average with exogenous inputs (NARMAX) models [52, 157], artificial neural networks [52], fuzzy logic [132] and machine learning techniques such as support vector machines [70]. Finally, the last type of non-linear models are called hybrid non-linear models which conceptually are a combination of both fundamental and empirical models [70], although in some cases they can be used to represent systems subject to abrupt dynamical changes where only discrete variables are used [138].

Regardless of the type of model to be used, the same principle applies to all the NMPC approaches where the models are used to predict and optimise the system's future behaviour. Moreover, the resulting models are commonly handled in discrete-time given the use of digital components for developing control systems, and will typically be arranged in one of two main forms: state-space models or Z-transfer function models, which in most cases can be used interchangeably although modern control systems nowadays prefer the use of state-space models given their ability to represent multi-variable systems in a relatively straight forward manner. In any case, a discrete transfer function model can always be exactly represented by state space model. A relatively simple approach to achieve this is to use the so called Non-Minimal State Space (NMSS) representations described in chapter 9 of [146] where the system is augmented as required with the states representing previous inputs and outputs based on the standard difference equations that arise from Z-transfer function models. Lastly, in the case of non-linear models, the resulting models will usually be required to be linearised to provide first-order sensitivity information for the optimisation. This can typically be done in a few ways, as described in [28] with two of the most common being: Internal Algorithmic Differentiation, where the optimisation simply differentiates the equations of the simulation (in the case of continuous systems, represented by the integrator method used); and External Numerical Differentiation (END), where the optimisation just treats the model as a black box and uses finite differences.

2.3 State and Parameter Estimation

Although there exist applications in industry that have the advantage of full state feedback or vibration free measurement allowing for explicit derivation, e.g. ([104]), it is often encountered that some form of state estimation is required to achieve good performance [19, 24, 67, 70, 136]. Indeed, in [19] is suggested that the use of a properly tuned observer is crucial for the performance enhancement of Generalized Predictive Control (GPC) schemes. In the case of nonlinear systems, many estimation strategies are available such Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF), Nonlinear Moving Horizon Estimation (NMHE) and Particle Filters (PF) [67, 70, 75, 136]. An interesting approach is presented in [136] were the dynamics of the EFK are embedded into the optimization accounting for future predicted innovations and correction/propagation phases of the EKF. Although the approach seems reasonable, the inherent amount of computations could increase exponentially given the matrix multiplications required by the propagation and correction phases of the co-variance matrix, thus making its implementation in real-time of great difficulty. A more reasonable approach in this case is the NMHE presented in [25, 54, 67] where the estimation and prediction are dealt separately, with the estimation algorithm takes into account the last N samplings to estimate the current state (x_k) starting from an estimated state N samples ago (x_{k-N}). Finally, authors from [75, 76] make use of multiple EFKs, ie. a bank of EKFs, for fault detection and isolation which could be used for achieving Fault-Tolerant Control when using NMPC methods.

On the other hand, the use of parameter estimation techniques is desirable in most systems given the underlying uncertainty of the mathematical models used, eg. for simulations [70, 80, 137]. Work presented in [80] makes use of a Recursive Least Squares (RLS) algorithm to estimate the parameters of an UAV coupled with an NMPC for attitude control. Similarly, authors from [132] make use of the forgetting factor RLS for estimating the parameters of a so called Takagi-Sugeno fuzzy model and implement an Adaptive Predictive Functional Control (APFC) for an average cracking outlet temperature of the ehtylene cracking furnace. The work in [70] utilises a Runge-Kutta based EKF for both state and parameter estimation together with a Runge-Kutta based NMPC formulation that was tested on two examples with surprisingly good results.

Both of these topics motivate the use of these kind of methods to improve the performance of the system as well as possibly achieve a Fault-Tolerant Control as in [75, 76]. Indeed, the RLS algorithm was used for parameter estimation as an "adaptation mechanism" in the publication of the UKACC 2018 conference paper [49], discussed later in chapter 4, and was used together with an EKF for both, online system identification and state estimation respectively in the IET 2020 journal publication [47].

2.4 Robustness, Sensitivity and Offset-Free Control

Another important property to consider when designing a NMPC is its robustness and sensitivity to noise, disturbances and model errors. The application of robust MPC approaches typically leads to models with various descriptions of the uncertainty such as multi-model systems [20, 136], as well as systems with bounded input-disturbances [20]. Standard robust MPC designs have been originally based on min-max approaches where the worst case cost of the prediction is optimised enforcing the fulfilment of constraints for all case of uncertainties [14, 20, 136]. Alternatively, tube based approaches use two controllers, a nominal controller and an ancillary controller where the task of the latter is to ensure the path of the system stays close to the nominal path [136]. An interesting approach is presented in [136] where multi-stage NMPC is designed optimising all the possible scenarios of the predicted evolution of the system. Each scenario is then given a probability based on an Interactive Multiple Models Filter (IMMF) and the respective probability is included into the optimisation as a weighting parameter. On the other hand, given robustness is more strongly analysed in the frequency domain, work in [128] presents a methodology for tuning the parameters (N_1, N_2, ρ, α) of linear GPC, thus allowing frequency shaping of the resulting closed loop solutions. Similarly, T-filter polynomials have been suggested in [19, 121, 122] for increasing the robustness of the closed loop performance. An important characteristic of this method is that it allows a two-degree of freedom design where the tracking and robustness of the system can be dealt with separately. Finally, the use of (Δ) integrated models for offset-free control which account for potential disturbances has been shown to result in rather sensitive solutions that typically require the use of the aforementioned T-filter polynomials to obtain good performance [122]. Based on this, authors from [69] concluded that the use of disturbance estimation methods for offset-free control provides the best performance when compared to (Δ) models which have been shown to be extremely sensitive, whilst also allowing a two-degree of freedom design for disturbance rejection and trajectory tracking. This will be discussed further in chapter 3, section 3.5 where various types of offset-free control methodologies will be introduced.

2.5 Stability and Recursive Feasibility

One of the most important things in NMPC is the stability of the underlying optimisation [19, 121]. This topic will be discussed in detail later in chapter 3, section 3.4. However, to give a general idea, [19] identified 4 main types of solutions which guarantee stability, namely: (i) infinite horizon, where the objective can then be considered a Lyapunov function, providing nominal stability; (ii) zero-terminal constraints, which enforce the state to be at a specific points at the end of a finite horizon; (iii) dual modes, where the system has an embedded optimal control law, typically a Linear Quadratic Regulator (LQR) that provides optimallity and a second mode deals mainly with constraint handling [79]; and (iv) quasi-infinite horizon which uses the idea of terminal region to add a terminal cost to the optimisation approximating the infinite horizon. Regarding dual modes, authors from [105] designed a sub-optimal dual mode NMPC based on dynamic inversion for a milling circuit machine. Similarly, work presented in [88] uses a nonlinear dynamic inversion (NDI) dual-mode NMPC for initialising the optimisation. Although dual modes represent the ideal solution given their ability to pre-stabilise the system, the most popular are those based on terminal costs, also known as the so called Mayer terms [82]. However, terminal costs add complexity into certain calculations related to the optimisation, thus if possible they should be omitted as they are often not even needed when the NMPC is carefully designed [87, 145]. A demonstration of this will be given later in example 3.2 in the following chapter.

On the other hand, recursive feasibility is another important property of the system that must be taken into account when designing a NMPC. Moreover, this property is the foundation of the Initial Value Embedding (IVE) strategy from the Real Time Iteration (RTI) Scheme; one of the main methods used by this thesis (see section 2.7). Thus, making it of extreme importance. Hence, the property will also be discussed in detail later in chapter 3, section 3.4. According to [94], an NMPC approach is recursively feasible **if and only if** for a given initially feasible solution, all subsequent solutions remain feasible for all time. In this work, they propose a methodology for invalidating the recursive feasibility of a MPC controller, ie. detect that it has problematic states where recursive feasibility is lost, and for finding certificates of guaranteed recursive feasibility both in the nominal and disturbed case allowing to even calculate the range of disturbances for which the system will preserve recursive feasibility.

2.6 Optimisation Methods

So far, no regard to the optimisation methods used for NMPC has been addressed and only some of the important aspects and concepts to consider as part of the design of NMPC systems such as modelling, state and parameter estimation, robustness, sensitivity, offset-free control, stability and recursive feasibility were discussed, introducing key ideas for both linear and nonlinear MPCs.

Within most of the available methods for NMPC optimisation, the same fundamental problem is present; the inability to analytically express the future in terms of the decision variables. With the exception of few types of simple nonlinear systems, eg. where a "virtual" variable can be used [19] or where the future can be expressed as expanded polynomials [23], the main challenge of NMPC when compared to MPC is to formulate the unknown future in terms of decision variables. This problem is unique to NMPC methodologies as linear MPC models have known exact representations.

Based on this, one of the most common techniques used in the literature are shooting methods where the future decision variables are guessed, the system is linearised along the resulting nominal/guessed trajectories and the optimisation improves the guess at every iteration using, eg. using the Gauss-Newton method [115]. In general, this leads to the solution of a series of linearised Quadratic Programs (QP), typically referred to as the Sequential Quadratic Programming (SQP) approach which is used to obtain the solution of the underlying optimisation problems formulated by the NMPC [104, 115]. If the system is represented in continuous time, a direct discretisation method can be used to reduce the intractable infinite dimension Optimal Control Problem (OCP), to a tractable finite dimension Nonlinear Programming (NLP) problem [65, 115]. We will discuss this in detail in chapter 3, section 3.1.2, where single/multiple shooting "discretisation" schemes will be introduced and thoroughly discussed.

Once a discretisation method is selected (if necessary), the optimisation can then be done using simultaneous or sequential approaches as discussed in [57, 115] where in the simultaneous approach the optimisation deals with both simulation and optimisation at the same time, and in the sequential approach, these tasks are carried out separately [115]. The main advantage of sequential approaches is that they implement the so called **condensing** approach to eliminate the state variables from the optimisation allowing faster computations for small to medium scale systems [145]. In contrast, simultaneous approaches keep the state variables for the optimisation hence increasing the number of decision variables whilst at the same time allowing the use of QP solvers that can exploit the sparse structure of the underlying QP leading to faster computation times for medium to large scale optimisation systems [145]. It is shown in [116, 145] that sparse methods can present linear performance w.r.t. the prediction horizon which makes them ideal for optimisations where large prediction horizons are required although they can still present computational challenges when the number of states and inputs are large [116], whereas condensing methods present cubic performance typically giving better performance at smaller prediction horizons. For this reason, the condensing approach was selected as part of the research topics for this Ph.D. thesis given the interest in relatively small systems and optimisations, as well as the method being a commonly used step on standard Linear MPC theory. Moreover, the thesis focuses on methods which rely precisely on the reduction of the number of decision variables of the optimisation, thus decreasing them even further than that of the standard condensing approach, making this step ideal for the developed approached. On the other hand, a potential advantage of the simultaneous approach is that they have been shown to perform better when dealing with unstable systems [57]. However this problem can now be tackled with the method proposed in chapter 6 of this thesis which allows the use of sequential approaches for a wide range of unstable nonlinear systems. Lastly, it is emphasised that the QP solver used to solve the optimisation is of great importance and therefore efficient methods and QP solvers will be discussed in the following sections. If unconstrained solutions are acceptable (eg. [58, 69, 92, 107, 128]), a classic way of solving the optimisation is by solving the Discrete Algebraic Riccati Equation as suggested in [107].

Other methods such as the Linear Parameter Varying (LPV) modelling approach presented in [11, 12, 23, 60] use the same shooting concept of guessing the future trajectories with the main difference being the models of the state matrices used in the optimisation. Although they prove to give efficient and good performance [11, 23] it has been shown in [111] that this type of representation does not guarantee the convergence to the optimum and could even lead to a phenomena called cycling where the solution jumps back and forth forever without making progress towards the optimum [72].

Among alternative methods we have the previously introduced "virtual variable" from [19]. In this case, the inputs of the system are represented with an "alternative/virtual" variable that allows the use of fully linear optimisation methodologies, and the variable is recovered afterwards using a nonlinear mapping. This concept is widely used for "control allocation" [72]; an optimisation method commonly used by aerospace, maritime and autonomous vehicles applications to represent the so called "virtual forces and moments" where the idea is to consider "high-level" control signals that represent an "overall" force or moment which can be achieved with the use of the inner actuators. A classic example of this are quad-rotors UAVs which although they typically hold a nonlinear relationship with the body, can usually be handled via simple control allocation techniques which allow a virtual linearisation of the system. This method was used to define the linear models for the UKACC 2018 publication [49], as well as for the Obstacle Avoidance problem of chapter 5, section 5.6. The main disadvantage of this method is that it forces the optimisation to be performed on the virtual variable, hence leading to a possible forced change in the index of optimisation (eg. see [154]). This can be relevant if the underlying optimisation is subject to "economic" costs where the system is attempting to minimise a given variable, eg. power or energy usage [56, 118]. Thus, in these cases, if using virtual variables modifies the definition of "power" it would not give an optimal solution unless some obvious relation between the minimum of the virtual variable and the real variable can be established. Naturally, if this approach is used, the constraints will have to be modified or considered in some relation with the virtual variable.

A similar alternative, which was one of the first methods originally developed for Dynamic Matrix Control (DMC) is the "extended linear model" discussed in [19] which basically attempts to capture the non-linearities of the system using an output-mismatch. This however prevents the system from recovering the actual optimum given the assumed independence of the output-mismatch non-linearity w.r.t to the decision variables.

Finally, a completely different set of approaches are those based on heuristic or algorithmic solutions of the optimisation. In [102] a Particle Swarm Optimization (PSO) algorithm is used to solve the optimisation where a set of randomly generated solutions named particles is given a random position and velocity and the algorithm iteratively attracts all the particles towards the best solution. A similar approach is used by [138] for the solution of a hybrid system with discrete inputs only. Work in [51] presents an accelerated leapfrogging optimisation algorithm that has better global optimum recovery than other methods. Although some of these methods present reasonable and simple approaches, in most cases they represent higher computational load or worse performance than those given, for example by efficient solutions which will be addressed in the following section.

2.7 Efficient Real-Time Solutions

The solution of NMPC problems is generally speaking a non-trivial task which can result in a significant computational burden if a naive approach is used [145]. Because of this, efficient solutions are currently being explored to tackle this problem, making it one of the most popular topics in the area of NMPC [21, 23, 38, 55, 65, 66, 83, 84, 114, 127, 131, 140, 145, 149, 152]. Efficiency may come in many different ways, from the way the optimisation is programmed [66, 145**?**] and the algorithms that are used [8], to the optimisation method that is used (eg. sequential or simultaneous [145]), the optimisation

solver that is used [37, 38, 53, 84, 140], the way the optimisation is initisalised [55] and the way the optimisation is formulated [21, 92, 131, 146, 152]. On the other hand, the implementation of the approach will generally be limited by the available hardware and its computational resources, such as memory, operating frequency, cache speeds, bus speeds, specialised floating-point processing units, and so on. However, in this section we will focus only on the underlying methods, with the understanding that some methods will be more applicable or "relevant" than others for a given hardware setup, as well as for a given application.

Given the relation between the optimisation in time, eg. between two subsequent samples, a-priori information can be used to warm-start the solution [55]. One of the most popular methods to do this is the Real Time Iteration (RTI) scheme, originally developed in [32]. Because of its popularity and underlying efficiency, this method was selected as a key part of this thesis. The method is well documented in [55] as well as in section 3.2 of the following chapter. To give a brief overview, the method is based on three main tasks, namely: (i) the use of an Initial Value Embedding (IVE) scheme where the solution of the optimisation at the previous sampling is used to warm-start by using a shifting strategy; (ii) the division of the computations into a preparation and feedback phase to avoid having feedback delays where the preparation phase uses an estimated state to "prepare" a QP before the next sampling is available, and the feedback phase quickly delivers an estimate once the system is sampled; and (iii) the solution is obtained based on a single linearized QP leading to approximate solutions that have been shown to give a reasonable estimate of the fully converged solution [55]. Additionally, work done in [145] indicates that up to 75% of the computations can be saved by using efficiently coded libraries such as the ACADO toolkit [66], especially when condensing approaches are used during preparation phases of the QP solver leading to timescales in the range of micro-mili seconds [65, 115, 154]. Of particular interest are the $O(N)/O(N^2)$ algorithms from Ph.D. thesis [8] which will be introduced later in chapter 3, section 3.3.1, and will be used extensively throughout the thesis to propose extensions implementing the proposed approaches of chapters 5, 6 and 8.

Among efficient methods for the RTI scheme, the Inexact NMPC Scheme from [152] proposed using incorrect/approximated derivative information allowing complete reduction of the computations related to the preparation phases, leading to efficient solutions with stability guarantees for non-negligible regions of the state space. In [21, 149], a so called Adjoint-based numerical method (adj-RTI) is proposed using approximated constraint Jacobians and modified gradients which are less expensive than the general ones [21]. A multi-level method for the RTI (ml-RTI) is discussed in [21] where the computations are divided into several components running at different sampling rates. In this method, the sensitivities are updated every $m > 1$ sampling times while the QP solver uses inexact derivatives until the next sensitivity update is done. Similarly, the recently proposed Curvature-like Measure of Nonlinearity for the RTI from [21] uses the so called Curvature Measure of Nonlinearity (CMoN) to trigger the sensitivity updates in specific points of the predicted trajectory, and sensitivities from previous sampling are used as long as they are considered reliable. Although this method is surprisingly intuitive and the results indicate a method 10 times faster than the original RTI, it is restricted to simultaneous optimisation give that it is exploiting the diagonal structure of the prepared QP to update individual sensitivities. Lastly, another recently proposed method which uses a stabilising partially tightened strategy that replaces the constraints in future stage costs with logarithmic barriers such as those used in interior point methods and a Ricatti-like Recursion is used to approximate the

terminal cost in the early stage of the optimisation is presented in [153]. Although the method is shown to perform up to 20 times faster, it is once again restricted to simultaneous approaches where the structure of the QP can be exploited, thus having a slightly different focus to the one pursued by the work undertaken by this thesis. Nonetheless, some of the proposed methods of this thesis, particularly that of the Shifting Strategy of chapter 5 and the Pre-stabilisation method of chapter 6 are based on generic concepts which could potentially be merged with the aforementioned methods.

Although not very popular, some of the alternative efficient methods are the Continuation/Generalized Minimal Residual (C/GMRES) and Advanced-Step NMPC, discussed in [36, 131]. The benefit of the C/GMRES method relies on avoiding the calculation of Hessian inverses by taking the advantage of the numerical continuation method presented in [131] that view the optimisation as a dynamical system governed by a differential equation where the user can specify some desired stable dynamics. On the other hand, the benefits of the Advanced-Step (or in general Advanced-multi-step) method presented in [36] is that it focuses on solving a problem one or multiple steps ahead, thus solving the optimisation multiple times in advance and quickly delivering feedback when measurements are available. Although, the authors don't relate this to the RTI scheme, it is obviously a similar approach. Indeed, a recent work from [108] proposed an Advanced Step RTI framework which combines the ideas of the RTI Scheme with those of the Advanced Step NMPC as well as with the aforementioned Multi-level iteration method discussed in [21]. However, it is questionable whether solving the optimisation multiple times in advance would lead to rather slow feedback against fast impulse disturbances, and could lead to potentially unstable optimisations due to the use of inexact sensitivities.

Other types of efficient methods are those that aim to reduce the number optimisation variables such as Laguerre and Move Blocking [18, 79, 92, 130, 135, 146], hereafter referred as input-parameterised solutions both of which formed the main motivations behind the contents of chapters 4 and 5. In the case of Laguerre-based input-parameterisations, various applications were found for linear MPC in [79, 92, 146] whereas only one instance was found for NMPC in [33], with no regard to the computational or algorithmic aspects of it. On the other hand, although it has been mainly investigated and properly referenced for linear MPC, move blocking is rather common among NMPC schemes (eg. [67, 85, 86]). However, an important problem of all these works is that they disregard the fundamental recursive feasibility problem that this method presents, in some cases attempting to compensate with the use of additional slack variables [86]. To solve this problem, work from [18] proposed a "Moving-Window-Blocking (MWB)" strategy that yields stability and recursive feasibility guarantees, although in the author's opinion the scheme has some deficiencies given that it does not embed a "constant" performance along the horizon because of the nature of the admissible set of matrices blocking matrices it uses (eg. see matrices that result from the use of algorithm 1), and the fact that it uses a time-varying prediction horizon. Other works such as [46] proposed a least-restrictive approach where inputs and state constraints where "minimally increased" or "relaxed" from the actual state/input constraints such that the move-blocking MPC problem was feasible. However, this method lacked any a-priori stability guarantees as the ones offered by the work from [18]. Based on this, chapter 4 focused on obtaining a general input-parameterisation framework which would allow the user to embed any desired parameterisation using the proposed algorithms. Additionally, chapter 5 proposed a novel time-varying Shifting Strategy supported by the aforementioned MWB Scheme which is shown to lead to significant computational gains whilst preserving stability and recursive feasibility of the optimisation.

Finally, a quite popular topic, particularly for linear MPC are those of explicit solutions where the optimisation solves a multi-parametric QP and the solution is converted to a look-up table where efficient algorithms such as the binary search tree can be used [135]. However, this is usually not a straightforward approach for NMPC and it is usually limited to small-state dimensions [37]. Nevertheless, work in [135] proposes a method where the NMPC is fully-solved offline generating a grid of 25,000 solutions and the solution is then approximated by using interpolating polynomials between the solutions. A similar approach is presented in [52].

2.8 Auto-generation Tool-kits and QP Solvers

As mentioned earlier, the use of appropriate (possibly auto-generated) algorithms for the solution of NMPC problems can lead significant computation savings [145] which has led to the development of a variety of tool-kits. In [66], a tutorial-style article for using the popular ACADO toolkit with Matlab is presented. This toolkit has the advantages of supporting a wide number of options from single/multiple shooting to Runge-Kutta Integrators, implicit integration schemes as well as a number of different QP solvers embedded into the toolkit. The tool-kit comes along with a code-generation tool, designed specifically to export optimised C-code tailored to the optimisation setup specified by the user. Among the available QP solvers, QPOASES [38] is one of the most popular ones giving its efficiency for small-medium scale optimisations [145]. Other QP solvers such as FORCES, qpDUNES, HPMPC and CVXGEN which are sparse interior-point based methods that allow exploitation of the problem's structure, have also been shown to give excellent results [55, 115, 145]. Given the popularity and impressive performance of this ACADO tool-kit, one of the main objectives of this thesis was to achieve better performance by using tailored auto-generation routines based on the Eigen 3 library. The resulting codes were benchmarked against the ACADO using QPOASES given its efficiency for small-medium scale optimisations, as well as its ability to solve dense QP programs.

Although not so popular, other tool-kits exist such as VIATOC, CasADi and GRAMPC, as well as the more recently proposed ACADOS [73, 143] which were not considered for comparisons.

2.9 Summary

This chapter presented a brief initial discussion around the main topics of NMPC, in particular those related to efficient real-time NMPC solutions which are relevant to the contents of this thesis. The chapter introduced important concepts and ideas related to various modelling approaches, state and parameter estimation techniques, common methods for achieving robustness, as well as sensitivity, offset-free control, stability and recursive feasibility of the NMPC optimisation methodologies. Moreover, it included a discussion around some of the state-of-the-art efficient real-time solutions for the popular Real-Time Iteration (RTI) Scheme which was found to be one of the most popular, intuitive and efficient, thus making it one of the main methodologies to be used throughout this thesis. Finally, the chapter ended with a brief discussion around the available tool-kits which allow a straightforward implementation of various NMPC approaches, from which the ACADO tool-kit was selected based on its popularity, support, reliability and impressive performance.

We remind the reader that given the decision to divide the literature review into both, non-technical and technical chapters, some of the concepts and ideas will be inevitably repeated in the following chapter where the overall methodologies will be discussed in significantly more detail to provide a clear set of notation and methods that will support the rest of the thesis.

Chapter 3

Technical Background

Having reviewed the available literature in the general topic of Nonlinear Model Predictive Control (NMPC), it was found that the notation used varied quite significantly from author to author which ultimately can lead to problems of interpretation and generalisation of the procedures involved. Moreover, given the lack of space available in regular papers, the introduction or discussion of basic/standard NMPC methodologies or algorithms is often found in a rather compressed or "too mathematical" format as it is an "obvious" or "well known" methodology, and often skips important details or concepts that provide key insights which could allow a more solid understanding of the basic principles and methodologies at hand. Both of this problems combined can prove the simplest understanding of the basic methods to be quite a challenging feat for anyone unfamiliar with the topic, or anyone who doesn't have a Degree in Mathematics. Therefore, this chapter is written as part of a "detailed" literature review with the purpose of establishing the foundational set of notation and methodologies to be used throughout the thesis in the hopes that it could serve as an entry point for the new student, as well as to establish a clear connection with Linear MPC and allow the contents of this thesis to be self-contained. The thesis follows the notation used in [146] for Linear MPC (now extended to NMPC) closely, in particular the use of variables such as (E, M, γ) for the Hessian, Constraints Matrix, and constraints vectors, respectively, of the general Quadratic Program (QP).

The chapter is organised as follows: Section 3.1 presents the derivation of the condensing-based multiple/single shooting NMPC methodology, and is divided into 3 main subsections which describe precise topics required to successfully understand and implement the general approach. Within this section, a couple of interesting theorems are given along the way which will be referred to in later chapters. Section 3.2 introduces the Real-Time Iteration (RTI) scheme which serves as the core method used by this thesis to achieve real-time performance, and ultimately represents the "state of the art" method to which the contributions of future chapters will be bench-marked against. Section 3.3 presents a set of algorithms, including a re-derivation of the "state of the art" $O(N^2)$ and $O(N)$ from Ph.D. thesis [8], all of which are ultimately used by the ACADO toolkit to implement the RTI Scheme. Section 3.4 discusses the general convergence, as well as nominal stability and recursive feasibility properties and methodologies used for NMPC. Section 3.5 contains a discussion around 3 of the main available methodologies for offset-free control in the literature for Linear MPC, and their possible implications when using them in a NMPC framework. Finally, the chapter ends with a brief summary of the contents in section 3.6.

3.1 Direct Optimal Control: Condensing-based Multiple/Single Shooting Nonlinear Model Predictive Control

Coming from a background on Linear MPC where the systems are typically handled in discrete time, and the decision variables of the optimisation are typically the future input trajectories as in [121, 122, 146], the natural step from this was to look into the so-called topic of "direct multiple shooting for nonlinear model predictive control" based on the condensing approach which essentially pose discrete-time Optimal Control Problems (OCPs) [83] that perform a "condensing" step to eliminate the states from the decision variables of the optimisation (as performed in Linear MPC, though not referred to this step as such), resulting in a small but dense QPs which have been shown to give better results for small to medium sized problems [145]. Thus, in this section we will introduce the required theory and methodologies to implement this approach.

Let us begin by formulating the basic Optimal Control Problem of interest given by:

$$\min_{\hat{X},\hat{U}} \quad J = \frac{1}{2}\left(X_r - \hat{X}\right)^T Q \left(X_r - \hat{X}\right) + \frac{1}{2}\left(\hat{U} - U_r\right)^T R \left(\hat{U} - U_r\right) \tag{3.1a}$$

$$s.t. \quad \hat{x}_{k+i} = f(x_{k+i-1}, u_{k+i-1}) \quad \forall i = [1 \rightarrow N_p] \tag{3.1b}$$

$$\hat{x}_k = x_0 \tag{3.1c}$$

$$X_{min} \leq \hat{X} \leq X_{max} \tag{3.1d}$$

$$U_{min} \leq \hat{U} \leq U_{max} \tag{3.1e}$$

where N_p is typically referred as the "prediction horizon"; $\hat{X} = [\hat{x}_{k+1}^T, \hat{x}_{k+2}^T, \cdots, \hat{x}_{k+N_p}^T]^T \in \mathbb{R}^{N_p n_x}$ is a column-vector of N_p "predicted states" of size n_x; $\hat{U} = [\hat{u}_k^T, \hat{u}_{k+1}^T, \cdots, \hat{u}_{k+N_p-1}^T]^T \in \mathbb{R}^{N_p n_u}$ is a column-vector of N_p "predicted inputs" of size n_u; $X_r = [x_{r_{k+1}}^T, x_{r_{k+2}}^T, \cdots, x_{r_{k+N_p}}^T]^T \in \mathbb{R}^{N_p n_x}$ is a column-vector of N_p "state references" of size n_x; $\hat{U}_r = [\hat{u}_{r_k}^T, \hat{u}_{r_{k+1}}^T, \cdots, \hat{u}_{r_{k+N_p-1}}^T]^T \in \mathbb{R}^{N_p n_u}$ is a column-vector of N_p "input references" of size n_u; $Q = blkdiag([q_{k+1}, q_{k+2}, \cdots, q_{k+N_p}]) \in \mathbb{R}^{N_p n_x \times N_p n_x}$ is a positive-semi-definite "state error weighting matrix", typically selected as a block diagonal matrix; $R = diag([r_k, r_{k+2}, \cdots, r_{k+N_p-1}]) \in \mathbb{R}^{N_p n_u \times N_p n_u}$ is a positive-definite "input error weighting matrix", typically selected as a diagonal matrix; equation (3.1b) are the state dynamics describing the state evolution from one step to the next, considered for all practical purposes controllable, observable and twice-continuously-differentiable; equation (3.1c) is the initial condition; equation (3.1d) are the state constraints of the optimisation, with $X_{max}/X_{min} \in \mathbb{R}^{N_p n_x}$ being column-vectors containing the limits of the states as appropriate; and equation (3.1d) are the input constraints of the optimisation, with $U_{max}/U_{min} \in \mathbb{R}^{N_p n_u}$ being column-vectors containing the limits of the inputs.

Remark 3.1. *Note that in some cases, only certain states or inputs may be required to be constrained which can be done by "selecting" the respective rows of the state prediction models discussed in subsection 3.1.2, in particular model (3.17a). We will discuss this further in subsection 3.1.3.*

Remark 3.2. *The special case where the state dynamics (3.1b) are expressed in continuous-time instead of discrete-time will be discussed in subsection 3.1.1 where a set of common discretisation by integration methodologies will be introduced.*

In addition to the basic cost function (3.1), it is often required to optimise a set of "outputs" of the system which may be different from the states themselves, and may even involve non-linear relationships with the states. Such an optimisation can be obtained simply by reformulating the cost function (3.1) to include the output function (3.2c), as well as the output constraints (3.2f), and the output errors cost (3.2a), resulting in the general output cost function (3.2) given by:

$$\min_{\hat{X}\hat{U}\hat{Y}} J = \frac{1}{2}\left[\left(X_r - \hat{X}\right)^T Q \left(X_r - \hat{X}\right) + \left(\hat{U} - U_r\right)^T R \left(\hat{U} - U_r\right) + \left(Y_r - \hat{Y}\right)^T Q_y \left(Y_r - \hat{Y}\right)\right] \quad (3.2a)$$

$$s.t. \quad \hat{x}_{k+i} = f(x_{k+i-1}, u_{k+i-1}) \quad \forall i = [1 \rightarrow N_p] \quad (3.2b)$$

$$\hat{y}_{k+i} = g(x_{k+i}) \quad (3.2c)$$

$$\hat{x}_k = x_0 \quad (3.2d)$$

$$X_{min} \leq \hat{X} \leq X_{max} \quad (3.2e)$$

$$Y_{min} \leq \hat{Y} \leq Y_{max} \quad (3.2f)$$

$$U_{min} \leq \hat{U} \leq U_{max} \quad (3.2g)$$

where $\hat{Y} = [\hat{y}_{k+1}^T, \hat{y}_{k+2}^T, \cdots, \hat{y}_{k+N_p}^T]^T \in \mathbb{R}^{N_p n_y}$ is a column-vector of N_p "predicted outputs" of size n_y; $Y_r = [y_{r_{k+1}}^T, y_{r_{k+2}}^T, \cdots, y_{r_{k+N_p}}^T]^T \in \mathbb{R}^{N_p n_y}$ is a column-vector of N_p "output references" of size n_y; $Q_y = blkdiag([q_{y_{k+1}}, q_{y_{k+2}}, \cdots, q_{y_{k+N_p}}]) > 0 \in \mathbb{R}^{N_p n_y \times N_p n_y}$ is a positive-definite "output error weighting matrix", typically selected as a block diagonal matrix; and Y_{max}/Y_{min} are column-vectors containing the limits of the output.

Remark 3.3. *Once again, it may be the case that only certain outputs are required to be constrained which can be done by selecting the respective rows of the output prediction models presented in subsection 3.1.2, in particular model (3.21).*

We make this distinction between "state-only" and "state-and-output" cost functions as it forms a special case when using the multiple-shooting framework, in particular due to the "expansion step" which will be discussed in subsection 3.1.3.

Moreover, to ensure nominal stability of the optimisation, this general Optimal Control Problem often includes the so called "terminal conditions" which typically come in the form of some terminal weight [122], for example by having a special q_{k+N_p} matrix, or a "zero-terminal" constraint which comes in the form of an "equality constraint" such as $\hat{x}_{k+N_p} = O$. These type of conditions will be discussed further in section 3.4, and can be added to this optimisation with very small modifications.

Any of the above optimisations represent a Non-Convex Nonlinear Programming Problem (NLP) or optimisation which is typically very hard to solve. One of the most competitive approaches to solve them are Sequential Quadratic Program (SQP) methods where the cost is linearized at a given trajectory, resulting in a linearized Convex Quadratic Program (QP) which can be used to find an optimal search direction, typically based in the Gauss-Newton method, that eventually converges to the local-optimal. Notice the linearisation of the trajectory is only defined after a given input/state pairs have been applied through the state dynamics (3.1b), and in the case of "state-and-output" problems, the output function (3.2c). A popular approach to achieve this are the so called "shooting methods" which use an "initially guessed" **nominal input trajectory** $\bar{U} = [\bar{u}_k^T, \bar{u}_{k+1}^T, \cdots, \bar{u}_{k+N_p-1}^T]^T \in \mathbb{R}^{N_p n_u}$, **nominal state trajectory** $\bar{X} = [\bar{x}_{k+1}^T, \bar{x}_{k+2}^T, \cdots, \bar{x}_{k+N_p}^T]^T \in \mathbb{R}^{N_p n_x}$, and in the case of "state-and-

output" cost functions, **nominal output trajectory** $\bar{Y} = [\bar{y}_{k+1}^T, \bar{y}_{k+2}^T \cdots, \bar{y}_{k+N_p}^T]^T$, to linearize the OCP along the trajectory and obtain the required sensitivity matrices for the optimisation. In order to achieve this using the condensed multiple/single shooting approach, there are three main steps or "tasks" at hand which must be understood properly which are:

1. Simulation and Linearisation
2. Prediction Modeling
3. Condensing-based Optimisation

These tasks will be described in detail in the following subsections.

3.1.1 Direct Optimal Control: Discretisation by Integration

One of the key tasks to be able to implement NMPC is to be able to "Simulate And Linearise" the system, in particular the operations related to the system dynamics (3.1b) which will allow the use of the prediction models (3.17a) and (3.21) discussed in the following subsection 3.1.2. In the case the system evolution is already described by nonlinear discrete-time equations, then the steps described in this subsection are not particularly required and one can simply proceed to establish the prediction models directly from them. However, it is rather common to find that the representation of physical systems are based on first principles which result in Ordinary Differential Equations (ODEs), thus describing the system evolution in continuous time. As the optimisation introduced previously is formulated in discrete-time, we will require to translate the continuous-time models into discrete-time, something which is sometimes referred to as a "Direct Optimal Control" approach that follows a "discretise-then-optimise" philosophy [8].

To tackle this, a common approach is to use the concept of "discretisation by integration" as discussed in [55]. Indeed, it is well known that solution to ODEs can be approximated (or simulated) to arbitrary accuracy by using integrator methods such as Explicit Euler or Explicit 4th Order Runge-Kutta (RK), as well as more general Implicit RK methods, and so on. In addition to the simulated values, the prediction models discussed in the following section also require the linearisation of the system dynamics (sometimes called the sensitivity matrices) [55], in particular matrices A_k, B_k. Therefore, an efficient method to achieve a discrete-time model is to translate the continuous-time dynamics into discrete-time by using the principle of internal numerical differentiation which obtains the linearisation matrices by differentiation of the integrator method itself [55], thus giving both, the simulation and linearisation parts required by the optimisation resulting in a "discretise-then-linearise" approach [55]. It is important to note that although the integrator method is an approximation to the real system evolution, the linearisation that results from its differentiation gives the exact derivatives of the method (to machine precision).

To illustrate the basic idea behind this overall methodology, let us describe its application by deriving the resulting algorithm that arises for the Explicit Euler integration scheme which can be found in algorithm 4 of [55].

Consider a general ODE describing some nonlinear dynamics of the form:

$$\dot{x} = f(x, u) \tag{3.3}$$

By applying a single Explicit Euler step on the segment $[k \to k+1]$ (or equivalently represented in time-domain $[0 \to T]$), and considering an initial state x_0 and a piece-wise constant input u_0 over the interval, we obtain the following:

$$x_T = x_0 + Tf(x_0, u_0) \tag{3.4}$$

In this simple case, the partial derivatives w.r.t the state and input are given by:

$$A_0 = I + T\frac{\partial f(x_0, u_0)}{\partial x_0} \quad B_0 = T\frac{\partial f(x_0, u_0)}{\partial u_0} \tag{3.5}$$

and directly represent the "total" derivatives A_k, B_k required at each step of the prediction models.

To visualise the pattern that emerges for a higher number of inner steps, consider now the case of dividing the segment in 3 steps (ie. $[0 \to \frac{T}{3} \to \frac{2T}{3} \to T]$).

The simulations of the system at each of the inner steps would simply be given by:

$$x_{\frac{T}{3}} = x_0 + \frac{T}{3}f(x_0, u_0) \tag{3.6a}$$

$$x_{\frac{2T}{3}} = x_{\frac{T}{3}} + \frac{T}{3}f(x_{\frac{T}{3}}, u_0) \tag{3.6b}$$

$$x_T = x_{\frac{2T}{3}} + \frac{T}{3}f(x_{\frac{2T}{3}}, u_0) \tag{3.6c}$$

Likewise, the resulting linearisation matrices at each of this inner steps are then given by:

$$A_0 = I + \frac{T}{3}\frac{\partial f(x_0, u_0)}{\partial x_0} \qquad B_0 = \frac{T}{3}\frac{\partial f(x_0, u_0)}{\partial u_0} \tag{3.7a}$$

$$A_{\frac{T}{3}} = I + \frac{T}{3}\frac{\partial f(x_{\frac{T}{3}}, u_0)}{\partial x_0} \qquad B_{\frac{T}{3}} = \frac{T}{3}\frac{\partial f(x_{\frac{T}{3}}, u_0)}{\partial u_0} \tag{3.7b}$$

$$A_{\frac{2T}{3}} = I + \frac{T}{3}\frac{\partial f(x_{\frac{2T}{3}}, u_0)}{\partial x_0} \qquad B_{\frac{2T}{3}} = \frac{T}{3}\frac{\partial f(x_{\frac{2T}{3}}, u_0)}{\partial u_0} \tag{3.7c}$$

Each of this matrices only represent the linearisation of "a third of the total step" (ie. $\frac{T}{3}$). In order to obtain the "total" linearisation matrix we must first understand the linearisation models that they represent.

Following a Taylor linearisation, the linearised models given by these matrices are given by:

$$\delta x_{\frac{T}{3}} = A_0\delta x_0 + B_0\delta u_0 \tag{3.8a}$$

$$\delta x_{\frac{2T}{3}} = A_{\frac{T}{3}}\delta x_{\frac{T}{3}} + B_{\frac{T}{3}}\delta u_0 \tag{3.8b}$$

$$\delta x_T = A_{\frac{2T}{3}}\delta x_{\frac{2T}{3}} + B_{\frac{2T}{3}}\delta u_0 \tag{3.8c}$$

where $\delta x_k \ \forall k = [0, \frac{T}{3}, \frac{2T}{3}, T]$ and δu_0 are the state and input deviations, respectively, from the points at which the linearisation was calculated.

As it can be seen, these models relate each of the intermediate steps with its immediate following step. This is still not useful given that the interest of this procedure is to understand the relationship between the initial step state deviation (δx_0) and input deviation (δu_0), with the final step state

deviation (δx_T). Moreover, it is important to understand that the optimisation will not "see" the inner steps, and will only optimise the points at the end of the segment of the integration method.

With the purpose of deriving the steps required by the Explicit Euler algorithm, let us assume that the initial value of the "total" linearisation matrices ($A_k^{[0]}, B_k^{[0]}$) is given by the matrices at the initial step, ie. are given by:

$$\delta x_{\frac{T}{3}} = \underbrace{A_0}_{A_k^{[0]}} \delta x_0 + \underbrace{B_0}_{B_k^{[0]}} \delta u_0 \tag{3.9}$$

By substituting equation (3.8a) into (3.8b) we obtain an intermediate linearised model up to the 2^{nd} inner step ($\frac{2T}{3}$).

$$\delta x_{\frac{2T}{3}} = \underbrace{A_{\frac{T}{3}} A_0}_{A_k} \delta x_0 + \underbrace{(A_{\frac{T}{3}} B_0 + B_{\frac{T}{3}})}_{B_k} \delta u_0 \tag{3.10}$$

or expressed in terms of the initial matrices of equation (3.9):

$$\delta x_{\frac{2T}{3}} = \underbrace{A_{\frac{T}{3}} A_k^{[0]}}_{A_k^{[1]}} \delta x_0 + \underbrace{(A_{\frac{T}{3}} B_k^{[0]} + B_{\frac{T}{3}})}_{B_k^{[1]}} \delta u_0 \tag{3.11}$$

In other words, the new "total" linearisation matrices are given by $A_k^{[1]} = A_{\frac{T}{3}} A_k^{[0]}$, and $B_k^{[1]} = A_{\frac{T}{3}} B_k^{[0]} + B_{\frac{T}{3}}$) where the terms ($A_{\frac{T}{3}}$ and $B_{\frac{T}{3}}$) represent the linearisation matrices at the "current step" of the algorithm.

Moving further along the simulated steps, the last step (T) can be obtained by substituting (3.11) in equation (3.8c), resulting in:

$$\delta x_T = \underbrace{A_{\frac{2T}{3}} A_k^{[1]}}_{A_k^{[2]}} \delta x_0 + \underbrace{(A_{\frac{2T}{3}} B_k^{[1]} + B_{\frac{2T}{3}})}_{B_k^{[2]}} \delta u_0 \tag{3.12}$$

Thus, the resulting pattern can be clearly seen where the new "total" linearisation matrices is a recursive function of the previous "total" linearisation matrices and the linearisation matrices at the "current step". The resulting algorithm is given in algorithm 3.1.

As this thesis is not focused on the topic of integrators, this basic algorithm was used for simulation of most of the non-linear systems and case studies presented throughout the thesis.

A similar procedure can be followed to derive the Explicit 4th Order Runge Kutta simulation and linearisation algorithm 5 provided in [55], or any other integration method for that matter. In the specific case of Implicit integration methods, the most efficient way is to implement the Implicit Function Theorem (IFT) as discussed in [114].

Algorithm 3.1: Explicit Euler Algorithm

Data: x_k, u_k, T, N_s

begin

- $T = T/N_s$; // Divide sampling time into N_s segments
- $x_{k+1} = x_k$; // Initialize output state
- $A_k = I, B_k = O$; // Initialize linearisation matrices
- **for** $i = 1$ **to** N_s **do**
  - $B_k = (I + T\frac{\partial f(x_{k+1},u_k)}{\partial x_{k+1}})B_k + T\frac{\partial f(x_{k+1},u_k)}{\partial u_k}$; // Propagate input-to-state matrix
  - $A_k = (I + T\frac{\partial f(x_{k+1},u_k)}{\partial x_{k+1}})A_k$; // Propagate state-transition matrix
  - $x_{k+1} = x_{k+1} + Tf(x_{k+1}, u_k)$; // Simulate system dynamics
- **end**

end

Result: x_{k+1}, A_k, B_k

3.1.2 The Multiple/Single Shooting Prediction Models

As discussed earlier, shooting methods are a popular approach which use a guessed trajectory for the nominal input ($\bar{U}$), the nominal state ($\bar{X}$), and nominal output ($\bar{Y}$) over which the QP is linearised, thus allowing the formulation of a linearised convex Quadratic Program. Although the topic of Multiple Shooting is "well known", there is a rather common miss-conception around many authors that Single/Multiple Shooting methods are "discretisation" methods that convert an infinite dimension continuous OCP into a finite dimension Nonlinear Programming Problem (NLP) in which a "grid" of points, commonly called the "shooting points", is formed [21, 30, 57]. However, the author of this thesis considers that both of this methodologies (Single/Multiple Shooting) are not particularly dealing with converting a "continuous-time" problem into a "discrete-time" problem as this task is done by the "discretisation by integration" process itself presented in the previous subsection, and are instead "prediction models linearisation" philosophies. Indeed, if one can apply the single or multiple shooting linearisation schemes to already nonlinear discrete systems, then the whole "discretisation" claim is not a "distinctive" part of the underlying process, and in this subsection we will present both of this scenarios to explain the resulting linearised prediction models in detail.

Single Shooting

The single shooting scenario is perhaps the most intuitive form of linearisation in which the system dynamics (3.1b) are simulated and linearised forward by using the nominal initial state ($\bar{x}_0$) and nominal input trajectory ($\bar{U}$) [115] to obtain the nominal state trajectory ($\bar{X}$), and the resulting state deviation prediction models.

Using a simple Taylor Series expansion, this results in a single shooting linearised prediction model of the form:

$$\begin{aligned}\hat{x}_{k+1} &= \bar{x}_{k+1} + \delta\hat{x}_{k+1} \\ &= \underbrace{f(\bar{x}_k, \bar{u}_k)}_{\bar{x}_{k+1}} + \underbrace{A_k\delta\hat{x}_k + B_k\delta\hat{u}_k}_{\delta\hat{x}_{k+1}}\end{aligned} \tag{3.13}$$

where $\delta\hat{x}_k = \hat{x}_k - \bar{x}_k$ and $\delta\hat{u}_k = \hat{u}_k - \bar{u}_k$ are the deviations of the state and input from their linearised points $(\bar{x}_k, \bar{u}_k)$, respectively, and A_k, B_k are the partial derivatives w.r.t to the nominal state and input, respectively, defined as:

$$A_k = \left.\frac{\partial f(\hat{x}_k, \hat{u}_k)}{\partial \hat{x}_k}\right|_{\substack{\hat{x}_k=\bar{x}_k \\ \hat{u}_k=\bar{u}_k}} \quad B_k = \left.\frac{\partial f(\hat{x}_k, \hat{u}_k)}{\partial \hat{u}_k}\right|_{\substack{\hat{x}_k=\bar{x}_k \\ \hat{u}_k=\bar{u}_k}} \tag{3.14}$$

Notice in this specific scenario, the value of $\bar{x}_{k+1}$ is selected as the actual result from the simulation with $\bar{x}_k$ and $\bar{u}_k$, ie. $\bar{x}_{k+1} = f(\bar{x}_k, \bar{u}_k)$, which is then subsequently used to simulate the value of $\bar{x}_{k+2}, \bar{x}_{k+3}, \cdots$, etc. This particular "assumption" or "strategy" is THE absolute most distinctive feature of single shooting (when compared with multiple shooting), in which the system is linearised following a philosophy of "the ACTUAL simulated value PLUS a deviation". As single shooting is just a special case of multiple shooting, let us now look into its methodology to obtain the final prediction models to be used throughout this thesis.

Multiple Shooting

A powerful extension of the single shooting modeling methodology is the multiple shooting approach where rather than representing the whole trajectory as a "single simulation", the trajectory is divided into smaller segments with each segment represented by its own simulation [77], as depicted in figure 3.1. This results in 2 very specific differences between the multiple shooting and single shooting approaches, namely:

1. A different linearised prediction model.

2. An expansion step.

In contrast to single shooting which only requires an initial state $(\bar{x}_0)$ and a nominal input trajectory $(\bar{U})$ to generate the predictions, the multiple shooting philosophy follows a different approach in which the user can linearise the system, not only around the nominal input trajectory $(\bar{U})$, but also around a possibly desired nominal state trajectory $(\bar{X})$, which is not necessarily the one resulting from the actual simulation as in the single shooting approach. This ultimately offers stronger flexibility in the problem initialisation, and also presents improved convergence properties, especially in the case of unstable systems [115]. We will discuss this further in the following subsection 3.1.3 where we will explain this in more detail, in particular, how it owes the improved convergence properties to the "expansion step" required for the next iteration of the SQP procedure.

Because the linearisation is now done on an arbitrary trajectory over the nominal state $(\bar{X})$, the method uses offsets $(\bar{d}_k)$ to compensate, and ensures the continuity of the overall trajectory by propagating this offsets forward as depicted in figure 3.1 by the red and blue arrows, respectively, allowing the optimisation to "understand" the effects of a given segment in future segments. As it can be seen from this figure, there may be some "inner" points, such as the ones that form part of the Explicit Euler integration scheme dicussed in the previous subsection, which are not "seen" by the optimisation. To understand this further, let us describe this by using the underlying mathematical models.

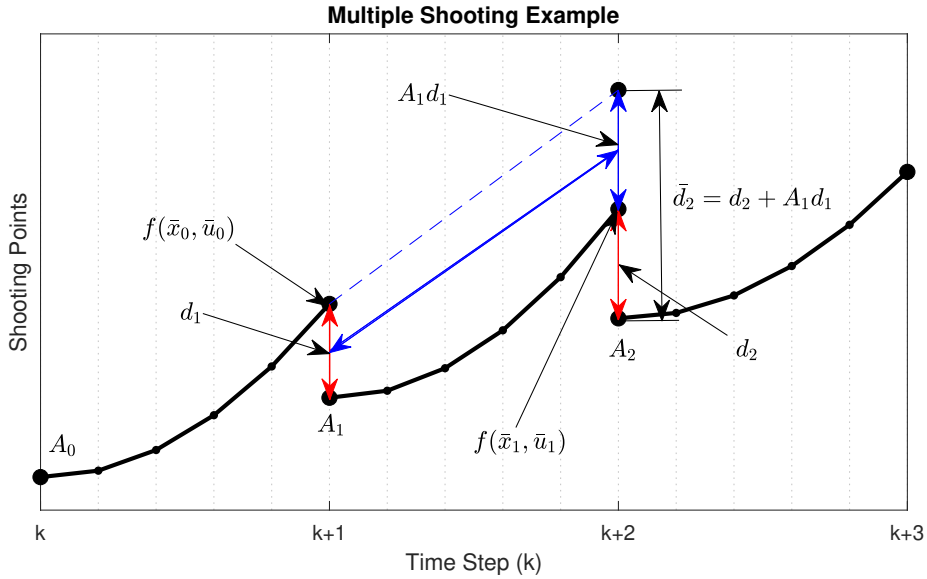


Figure 3.1: Example of the Multiple Shooting Modeling Approach

The resulting linearised prediction models that arise from this scenario are given by the following:

$$\hat{x}_{k+1} = \bar{x}_{k+1} + \delta\hat{x}_{k+1} = \bar{x}_{k+1} + \underbrace{A_k\delta\hat{x}_k + B_k\delta\hat{u}_k + \boxed{d_{k+1}}}_{\delta\hat{x}_{k+1}} \tag{3.15a}$$

$$d_{k+1} = f(\bar{x}_k, \bar{u}_k) - \bar{x}_{k+1} \tag{3.15b}$$

where the d_{k+1} term signaled by the red box is the "distinctive" part when compared to the single shooting model (3.13). As it can be seen this term contains the difference between the nominal simulated value $f(\bar{x}_k, \bar{u}_k)$ and the nominal value ($\bar{x}_{k+1}$), depicted by the red arrows in figure 3.1. The reader might argue that the simple summation of $\bar{x}_{k+1} + d_{k+1} = f(\bar{x}_k, \bar{u}_k)$, which makes the whole modeling exactly the same as in the single shooting approach, and for the very first prediction step, it is! The difference however relies in how this offset (d_{k+1}), which forms part of the predicted state deviation ($\delta\hat{x}_{k+1}$) sometimes referred as the continuity condition [28, 85], propagates into future values of the predictions, depicted by the blue arrows in figure 3.1. This can be seen more clearly by forming the model in the following step.

By following the same linearisation procedure to formulate the prediction model at next step $(k+2)$, and substituting the value of the predicted step deviation ($\delta\hat{x}_{k+1}$) in the predictions, the resulting models is given by:

$$\begin{aligned}
\hat{x}_{k+2} &= \bar{x}_{k+2} + \delta\hat{x}_{k+2} \\
&= \bar{x}_{k+2} + A_{k+1}\delta\hat{x}_{k+1} + B_{k+1}\delta\hat{u}_{k+1} + d_{k+2} \\
&= \bar{x}_{k+2} + A_{k+1}\underbrace{(A_k\delta\hat{x}_k + A_{k+1}B_k\delta\hat{u}_k + d_{k+1})}_{\delta\hat{x}_{k+1}} + B_{k+1}\delta\hat{u}_{k+1} + d_{k+2} \\
&= \bar{x}_{k+2} + A_{k+1}A_k\delta\hat{x}_k + A_{k+1}B_k\delta\hat{u}_k + B_{k+1}\delta\hat{u}_{k+1} + \underbrace{\boxed{d_{k+2} + A_{k+1}d_{k+1}}}_{\bar{d}_{k+2}}
\end{aligned} \tag{3.16}$$

where the new term ($\bar{d}_{k+2} = d_{k+2} + A_{k+1}d_{k+1}$), signaled by the red box, actually contains the previous term propagated through the linearised prediction matrix (A_{k+1}), as depicted in the "total" offset ($\bar{d}_{k+2}$) signaled by the black arrow limits in the step $k+2$ of figure 3.1.

By propagating this linearised models forward N_p steps and grouping the terms in matrix/vector formats, the resulting linearised prediction models can be found to be given by:

$$\hat{X} = \bar{X} + \delta\hat{X} = \bar{X} + \underbrace{D + G\delta x_0 + H\delta\hat{U}}_{\delta\hat{X}} \tag{3.17a}$$

$$\hat{U} = \bar{U} + \delta\hat{U} \tag{3.17b}$$

where $\delta x_0 = x_0 - \bar{x}_0$ is an initial condition mismatch which forms an important part of the RTI Scheme introduced in section 3.2, $D \in \mathbb{R}^{N_p n_x}$, $G \in \mathbb{R}^{N_p n_x \times n_x}$, $H \in \mathbb{R}^{N_p n_x \times N_p n_u}$ are defined as:

$$D = \begin{bmatrix} \bar{d}_1 \\ \bar{d}_2 \\ \vdots \\ \bar{d}_{N_p} \end{bmatrix} \qquad G = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_{N_p} \end{bmatrix} \qquad H = \begin{bmatrix} h_{1,1} & 0 & \cdots & 0 \\ h_{2,1} & h_{2,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ h_{N_p,1} & h_{N_p,2} & \dots & h_{N_p,N_p} \end{bmatrix} \tag{3.18a}$$

and with a slight abuse of notation by dropping the $k+i$ notation (eg. $A_k \to A_0, A_{k+1} \to A_1$,etc.), the inner matrices/vectors are defined through the following recursions as:

$$\bar{d}_k = \begin{cases} d_k & k = 1 \\ d_k + A_{k-1}\bar{d}_{k-1} & k > 1 \end{cases} \tag{3.19a}$$

$$g_k = \begin{cases} A_{k-1} & k = 1 \\ A_{k-1}g_{k-1} & k > 1 \end{cases} \tag{3.19b}$$

$$h_{k,j} = \begin{cases} B_{k-1} & k = j \\ A_{k-1}h_{k-1,j} & k > j \end{cases} \tag{3.19c}$$

$$\forall k = [1 \to N_p], \quad j = [1 \to N_p]$$

Notice the case of single shooting is included in this general model as if one chooses to simulate and linearise over the actual state trajectory, eg. $\bar{x}_{k+1} = f(\bar{x}_k, \bar{u}_k)$, the $D = \mathbb{O}$ term simply becomes zero, and $\bar{X}$ becomes the actual state trajectory.

In the specific case that an "output" linearised prediction model is required for nonlinear outputs of the form (3.2c) to be used in an "state-and-output" cost function such as (3.2), a similar procedure can be followed by observing that the core linearised prediction model at each step is given by:

$$\begin{aligned}\hat{y}_{k+1} &= \bar{y}_{k+1} + C_{k+1}\delta\hat{x}_{k+1} + d_{y_{k+1}} \\ &= \bar{y}_{k+1} + C_{k+1}(A_k\delta\hat{x}_k + B_k\delta\hat{u}_k + d_{k+1}) + d_{y_{k+1}} \qquad (3.20a)\\ d_{y_{k+1}} &= g(\bar{x}_{k+1}) - \bar{y}_{k+1} \qquad (3.20b)\end{aligned}$$

where C_{k+1} is the partial derivative of the output function (3.2c) given by:

$$C_{k+1} = \left.\frac{\partial g(\hat{x}_{k+1})}{\partial x_{k+1}}\right|_{\hat{x}_{k+1}=\bar{x}_{k+1}} \qquad (3.20c)$$

Thus, this results in a "total" general output linearised prediction model of the form:

$$\hat{Y} = \bar{Y} + \delta\hat{Y} = \bar{Y} + \underbrace{D_y + G_y + H_y\delta\hat{U}}_{\delta\hat{Y}} \qquad (3.21)$$

where $D_y \in \mathbb{R}^{N_p n_y}$, $G_y \in \mathbb{R}^{N_p n_y \times n_x}$, $H_y \in \mathbb{R}^{N_p n_y \times N_p n_u}$ are defined as:

$$D_y = \begin{bmatrix} \bar{d}_{y_1} \\ \bar{d}_{y_2} \\ \vdots \\ \bar{d}_{y_{N_p}} \end{bmatrix} \qquad G_y = \begin{bmatrix} g_{y_1} \\ g_{y_2} \\ \vdots \\ g_{y_{N_p}} \end{bmatrix} \qquad H_y = \begin{bmatrix} h_{y_{1,1}} & 0 & \cdots & 0 \\ h_{y_{2,1}} & h_{y_{2,2}} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ h_{y_{N_p,1}} & h_{y_{N_p,2}} & \dots & h_{y_{N_p,N_p}} \end{bmatrix} \qquad (3.22)$$

with the inner matrices/vectors are defined in terms of the "original" matrices values (3.19) by:

$$\begin{aligned}\bar{d}_{y_k} &= d_{y_k} + C_k\bar{d}_k \qquad (3.23a)\\ g_{y_k} &= C_k g_k \qquad (3.23b)\\ h_{y_{k,j}} &= C_k h_{k,j} \qquad (3.23c)\\ &\forall k = [1 \to N_p], \quad j = [1 \to N_p]\end{aligned}$$

It is important to clarify that both state and output models are required for the proper implementation of the multiple shooting scheme as both of them must be used in the "expansion step".

Multiple or Single Shooting for Linear MPC

An unfamiliar reader might be wondering how do all these models relate to the standard Linear MPC approaches. Thus, in order to establish a clear relationship between the general nonlinear methodologies discussed throughout this thesis and common linear MPC methodologies, we present the following theorem relating the equivalence of single/multiple shooting models to avoid confusion regarding the expectation about single/multiple shooting methods for linear systems. This comes to say that it is irrelevant to talk about single or multiple shooting if the system dynamics or output functions are linear, and one can instead just refer to standard MPC theory instead for this case.

Theorem 3.1. *Equality of Single/Multiple Shooting for Linear Systems*

Both single and multiple shooting prediction models result in the exact same predictions for a linear system which consequently results in the exact same optimisation.

Proof. For a linear system, we have the following predictions for the single shooting model,

$$\begin{aligned}\hat{x}_{k+1} &= \underbrace{\bar{x}_{k+1}}_{f(\bar{x}_k,\bar{u}_k)} + A\underbrace{(x_k - \bar{x}_k)}_{\delta x_k} + B\underbrace{(u_k - \bar{u}_k)}_{\delta x_k} \\ &= \underbrace{A\bar{x}_k + B\bar{u}_k}_{f(\bar{x}_k,\bar{u}_k)} + A(x_k - \bar{x}_k) + B(u_k - \bar{u}_k) \\ &= Ax_k + Bu_k \end{aligned} \tag{3.24}$$

Similarly, the predictions of a multiple shooting model result in,

$$\begin{aligned}\hat{x}_{k+1} &= \bar{x}_{k+1} + A(x_k - \bar{x}_k) + B(u_k - \bar{u}_k) + \underbrace{f(\bar{x}_k, \bar{u}_k) - \bar{x}_{k+1}}_{d_{k+1}} \\ &= \underbrace{A\bar{x}_k + B\bar{u}_k}_{f(\bar{x}_k,\bar{u}_k)} + A(x_k - \bar{x}_k) + B(u_k - \bar{u}_k) \\ &= Ax_k + Bu_k \end{aligned} \tag{3.25}$$

Ultimately this leads to the overall prediction models given by the well known linear MPC structure such as:

$$\hat{X} = Gx_k + H\hat{U} \tag{3.26}$$

with x_k being the actual initial state (not the deviation). □

Autonomous/Auto-correcting Feature of Multiple Shooting Prediction Model

Something which might not be immediately clear from the multiple shooting linearisation models (3.17a) and (3.21) is that they have an "autonomous/auto-correcting" feature where their successive use allows correcting the prediction errors arising from the nonlinearities, independently of whether they are used in an optimisation context or not. To illustrate this, let us use the following example:

Example 3.1. *Multiple Shooting Autonomous/Auto-correction Feature*

Consider the dynamics of a simple pendulum in free-fall/swing (no control action), described by the ODE:

$$\ddot{\theta} = -0.3\dot{\theta} + 9.81\sin(\theta) \tag{3.27}$$

where $\theta, \dot{\theta}, \ddot{\theta}$ are the pendulum's angle, angular velocity, and angular acceleration, respectively.

This ODE can be put in the standard nonlinear state space form ($\dot{x} = f(x)$) given by:

$$\underbrace{\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix}x_2\\ -0.3x_2 + 9.81\sin(x_1)\end{bmatrix}}_{f(x)} \tag{3.28}$$

For the purpose of this example, consider the simulation and linearisation of this system using the Explicit Euler algorithm 3.1 with a sampling time $T = 0.02$ (s), $N_s = 1$ number of steps, and linearised state prediction model (3.17a). Moreover, to include the output linearised prediction model (3.21), consider an output function given by:

$$y = \cos(x_1) \tag{3.29}$$

related to the potential energy of the pendulum as discussed in chapter 6, section 6.6.

Now, let us assume that the pendulum starts swinging from the initial condition $x_0 = [3.992, 0]^T$ and that we want to obtain the predictions for the future states and outputs $(\hat{X}, \hat{Y})$, $N_p = 200$ steps ahead using the linearised prediction models, (3.17a) and (3.21), with the input-related terms ($H\delta\hat{U}$,$H_y\delta\hat{U}$) being zero. In the single-shooting scenario, one would simply simulate the system and linearise around the resulting nominal state $\bar{X}$. Instead, the multiple-shooting approach requires the user to provide an initial guess for $\bar{X}$. If there is no available knowledge about a "good" initial guess for the nominal state, a safe approach is to select the one that results from the forward simulation (as used in the single-shooting approach), something which the ACADO toolkit offers in its auto-generated function "acado_initializeNodesByForwardSimulation()". We will discuss other forms of initialisation later on in section 3.2. However, with the strict objective of illustrating the auto-correcting feature, let us assume that we decide to use a relatively "bad" initial guess for the nominal state trajectory by simply selecting the initial state across all the nominal state, ie $\bar{X} = [x_0^T, x_0^T, \cdots, x_0^T]^T$, and that our initial guess for the nominal output trajectory is given by its evaluation at the initial state, ie. $\bar{Y} = [cos(x_0), cos(x_0), \cdots, cos(x_0)]^T$.

By starting from the aforementioned conditions, formulating the linearised prediction models and iteratively re-linearising around the "predicted" trajectories obtained from the previous iteration (ie. $\bar{X}^{[k]} = \hat{X}^{[k-1]}$ and $\bar{Y}^{[k]} = \hat{Y}^{[k-1]}$), something which the multiple shooting approach does in the "expansion step" discussed earlier, we obtain the behaviour visible in figure 3.2. In this figure, the green dashed line represents the initial guess for both nominal trajectories; the red dashed lines represent the actual trajectory that the pendulum will follow from the initial state; the cian dashed lines are the set of predictions iteratively obtained from procedure described above; and the thick blue line represent the final prediction, naturally converging to the actual trajectory. The predictions converge to the actual trajectory in only in 4 iterations, with each of the iterations signaled by an arrow/number pair. As it can be seen from the iterations, the predictions (cian-dashed lines) at the earliest steps quickly converge to the actual trajectory (red-dashed), whereas taking longer for the later parts of the trajectory.

There are 2 important lessons that we can obtain from this, firstly: 1. The trajectory that the multiple shooting linearised model predicts is NOT necessarily the real one, and secondly 2. The multiple shooting modelling will "automatically" correct for prediction errors as it iterates, typically

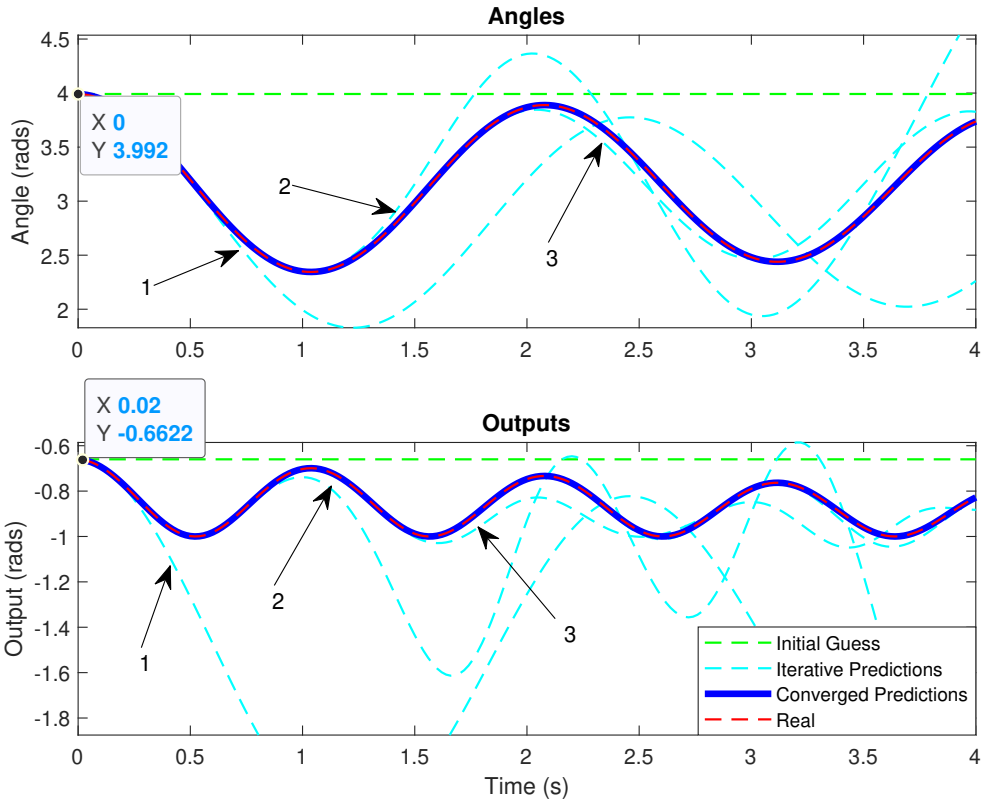


Figure 3.2: Example of the Multiple Shooting Autonomous/Auto-correction Feature

correcting the earliest prediction errors first as they are supported by more "accurate" predictions emerging from the propagation of the offset (d_k) through the linearised state-transition dynamics (A_k). Nonetheless, each specific case will be different as it will obviously depend on the selected value for the initial guess.

3.1.3 Condensing-based Optimisation

Having established the linearised prediction models to be used for the optimisation, we can now proceed to reformulate the general Quadratic Program of interest (3.1). In this section we will show two main ways in which the optimisation can be reformulated using a condensing-based approach: the Relative and Non-Relative formats. In this type of optimisation, the states are removed from the decision variables, leading to dense OCPs that can be solved with any general purpose Quadratic Programming solver [8]. To save space we will only present the derivation of the resulting Quadratic Problems for the "state-only" cost function (3.1), as it is the most common requirement, with the understanding that the "state-and-output" cost function case can be easily derived by following similar steps.

Relative Quadratic Program

In the condensed relative framework, which is the most commonly used as it represents the "classic" Newton type method giving an increment to the nominal point (eg. in the form of $x^+ = x^- + \Delta x$), and indeed the one used by the ACADO toolkit, the decision variables of the optimisation are the

input "deviations", ie. $\delta\hat{U}$. Thus, one can directly substitute the linearised prediction models (3.17a and 3.17b) into cost function (3.1) resulting in:

$$J = \frac{1}{2}(X_r - \bar{X} - D - G\delta x_0 - H\delta\hat{U})^T Q(X_r - \bar{X} - D - G\delta x_0 - H\delta\hat{U}) + \frac{1}{2}(\bar{U} + \delta\hat{U} - U_r)^T R(\bar{U} + \delta\hat{U} - U_r) \tag{3.30a}$$

$$s.t. \quad X_{min} \leq \bar{X} + D + G\delta x_0 + H\delta\hat{U} \leq X_{max} \tag{3.30b}$$

$$U_{min} \leq \bar{U} + \delta\hat{U} \leq U_{max} \tag{3.30c}$$

Note the state dynamics (3.1b) and the initial condition (3.1c) are already included in the prediction model.

After grouping common terms w.r.t. the decision variable ($\delta\hat{U}$), disregarding any constant terms in the cost and rearranging the constraints to be described by an inequality of the form $Ax \leq b$ to express the cost function in the standard dense QP format, results in:

$$\min_{\delta\hat{U}} \quad J = \frac{1}{2}\delta\hat{U}^T E\delta\hat{U} + \delta\hat{U}^T f \tag{3.31a}$$

$$M\delta\hat{U} \leq \gamma \tag{3.31b}$$

$$E = H^T Q H + R \tag{3.31c}$$

$$f = -(H^T Q(X_r - \bar{X} - D - G\delta x_0) - R(\bar{U} - U_r)) \tag{3.31d}$$

$$M = \begin{bmatrix} I \\ -I \\ H \\ -H \end{bmatrix} \quad \gamma = \begin{bmatrix} U_{max} - \bar{U} \\ -(U_{min} - \bar{U}) \\ X_{max} - \bar{X} - D - G\delta x_0 \\ -(X_{min} - \bar{X} - D - G\delta x_0) \end{bmatrix} \tag{3.31e}$$

where $E \in \mathbb{R}^{N_p n_u \times N_p n_u}$ is called the "Hessian" of the optimisation; $f \in \mathbb{R}^{N_p n_u}$ is typically referred to as the "linear term"; $M \in \mathbb{R}^{2N_p(n_x+n_u)\times N_p n_u}$ is the constraint matrix; and $\gamma \in \mathbb{R}^{2N_p(n_x+n_u)}$ is the constraints vector.

We can see from the constraint vector (γ) that the input constraints are expressed relative to the linearised trajectory ($\bar{U}$), hence the name of the approach ("relative") used by the author of this thesis.

Non-Relative Quadratic Program

In contrast, the condensed non-relative framework keeps the actual inputs, ie. $\hat{U}$, as the decision variables. In this case, notice the relationships $\delta\hat{U} = \hat{U} - \bar{U}$ can be replaced in the linearised state prediction model (3.17a) resulting in an equivalent prediction model of the form:

$$\hat{X} = \bar{X} + D + G\delta x_0 + H(\hat{U} - \bar{U}) \tag{3.32}$$

Following the same steps, we can now proceed to substitute this model into the cost function (3.1), group common terms w.r.t the decision variable $\hat{U}$ and disregard any constant terms in the cost to express the cost function the standard QP format, thus resulting in:

$$\min_{\hat{U}} \quad J = \frac{1}{2}\hat{U}^T E\hat{U} + \hat{U}^T f \tag{3.33a}$$

$$M\hat{U} \leq \gamma \tag{3.33b}$$

$$E = H^T Q H + R \tag{3.33c}$$

$$f = -(H^T Q(X_r - \bar{X} - D - G\delta x_0 + H\bar{U}) + RU_r) \tag{3.33d}$$

$$M = \begin{bmatrix} I \\ -I \\ H \\ -H \end{bmatrix} \quad \gamma = \begin{bmatrix} U_{max} \\ -U_{min} \\ X_{max} - \bar{X} - D - G\delta x_0 + H\bar{U} \\ -(X_{min} - \bar{X} - D - G\delta x_0 + H\bar{U}) \end{bmatrix} \tag{3.33e}$$

Notice in this case, the input constraints are expressed in a "non-relative" format (U_{max}/U_{min}), thus the name of the approach.

Remark 3.4. *Constraints Selection*

As discussed earlier, the user might require only certain states or inputs which can be done by selecting (or computing) only the relevant rows of the constraint matrix (M) and constraint vector (γ).

Optimal Solutions

Having defined any of these Quadratic Programs, any general purpose solver such as the quadprog function from Matlab, or QP OASES [38] can be used to solve the optimisation.

For unconstrained problems, the well known solution can be found by differentiating J w.r.t the decision variable, equating to zero (eg. $\frac{\partial J}{\partial \hat{U}} = 0$) and solving, resulting in:

$$\delta\hat{U}^*_{unc} = -E^{-1}f \qquad \text{For Relative} \tag{3.34}$$

or

$$\hat{U}^*_{unc} = -E^{-1}f \qquad \text{For Non-Relative} \tag{3.35}$$

For constrained problems, the optimal solution can be shown to be the optimal unconstrained solution plus an optimal correction term ($\hat{U}^*_\lambda$ or $\delta\hat{U}^*_\lambda$) related to the Lagrange multipliers (λ) that satisfy the Karush-Kush-Tucker (KKT) conditions as discussed in [146], ie. having the form of:

$$\delta\hat{U}^* = \delta\hat{U}^*_{unc} + \delta\hat{U}^*_\lambda \qquad \text{For Relative} \tag{3.36}$$

or

$$\hat{U}^* = \hat{U}^*_{unc} + \hat{U}^*_\lambda \qquad \text{For Non-Relative} \tag{3.37}$$

After obtaining the solution, only the first input of the resulting trajectory ($\hat{u}_0$) is implemented to the actual system, and the whole process is repeated in the next iteration which is the well known "receding horizon" strategy. In the particular case of multiple shooting, an "expansion step" is required to calculate the nominal state (and if required, the nominal output) trajectory to be used in the following step, which can be done by calculating the terms:

$$\left.\begin{aligned}\hat{X} &= \bar{X} + D + G\delta x_0 + H\delta\hat{U}^* \\ \hat{Y} &= \bar{Y} + D_y + G_y\delta x_0 + H_y\delta\hat{U}^*\end{aligned}\right\} \quad \text{For Relative} \tag{3.38}$$

$$\left.\begin{aligned}\hat{X} &= \bar{X} + D + G\delta x_0 + H(\hat{U}^* - \bar{U}) \\ \hat{Y} &= \bar{Y} + D_y + G_y\delta x_0 + H_y(\hat{U}^* - \bar{U})\end{aligned}\right\} \quad \text{For Non-Relative} \tag{3.39}$$

It is this particular step which provides better convergence properties to the multiple shooting approach by allowing the linearisation at the next step to be around the predicted trajectory in the previous step, and compensating for any prediction errors that emerge from the linearisation through the aforementioned offsets. In contrast, the single shooting approach completely disregards or "ignores" any possible prediction errors emerging from the linearisation process and simply linearises at whatever trajectory emerges from $\bar{U}$ and x_0 which consequently suffers from bad convergence properties, particularly if the system has unstable dynamics [8, 77].

Note that even in the case where one would opt to optimise a set of nonlinear outputs, eg. by having an "output-only" cost, the whole multiple shooting procedure would STILL require the "expansion step" applied to the nominal state trajectory ($\bar{X}$) to be able to obtain the required linearised prediction models. It is for this reason that a distinction was made earlier in this chapter.

Theorem 3.2. *The Equality of the Solutions*

The optimal solutions for both, "relative" and "non-relative" types of optimisation will always result in the exact same solution.

Proof. To satisfy this statement, both optimal solution must satisfy the following equality:

$$\bar{U} + \delta\hat{U}^* = \hat{U}^* \tag{3.40}$$

Consider substituting the unconstrained optimal solution for both approaches in (3.40), resulting in:

$$\begin{aligned}\bar{U} + E^{-1}\left(H^T Q(X_r - \bar{X} - D - G\delta x_0) - R(\bar{U} - U_r)\right) \\ = E^{-1}\left(H^T Q(X_r - \bar{X} - D - G\delta x_0 + H\bar{U}) + RU_r\right)\end{aligned} \tag{3.41}$$

After eliminating common terms and reorganising, this reduces to:

$$\begin{aligned}\bar{U} &= E^{-1}\underbrace{\left(H^T QH + R\right)}_{E}\bar{U} \\ &= \cancel{E^{-1}}\cancel{E}\bar{U}\end{aligned} \tag{3.42}$$

Thus satisfying the equality 3.40 for unconstrained solutions.

Given the unconstrained solutions have been proved to be the same, proving the equality of constrained solutions reduces to the additional correction terms ($\hat{U}^*_\lambda$ and $\delta\hat{U}^*_\lambda$) satisfying:

$$\hat{U}^*_\lambda = \delta\hat{U}^*_\lambda \tag{3.43}$$

Following the procedure in [146], both of this terms can be shown to have the form:

$$\hat{U}^*_{\lambda} = -E^{-1}M^T\lambda^*_{\hat{U}} \tag{3.44a}$$

$$\delta\hat{U}^*_{\lambda} = -E^{-1}M^T\lambda^*_{\delta\hat{U}} \tag{3.44b}$$

where $\lambda^*_{\hat{U}}$ and $\lambda^*_{\delta\hat{U}}$ are the optimal vectors of the Lagrange Multipliers of each solution containing only positive or zero values related to the active/inactive constraints, respectively.

Given both solutions have the same term $E^{-1}M^T$, the problem reduces to proving both solutions have the same optimal vector of Lagrange Multipliers ($\lambda^*_{\hat{U}} = \lambda^*_{\delta\hat{U}}$). The optimal solution for both of this vectors can be obtained based on the active-set methodology described in [146] which results in:

$$\lambda^*_{act_{\hat{U}}} = -(M_{act}E^{-1}M^T_{act})^{-1}(\gamma_{act_{\hat{U}}} - M_{act}\hat{U}^*_{unc}) \tag{3.45a}$$

$$\lambda^*_{act_{\delta\hat{U}}} = -(M_{act}E^{-1}M^T_{act})^{-1}(\gamma_{act_{\delta\hat{U}}} - M_{act}\delta\hat{U}^*_{unc}) \tag{3.45b}$$

Note that we have made a momentary distinction between both constraint vectors ($\gamma_{act_{\hat{U}}}$ and $\gamma_{act_{\delta\hat{U}}}$).

If one requires to implement this expression in an active-set method, a basic requirement is the invertibility of the term $(M_{act}E^{-1}M^T_{act})^{-1}$ which can only be achieved if there is linear independence between the active constraints, and the number of active constraints is less than the decision variables [146]. However, by equating both of this optimal vectors, this term cancels out leaving only the equality of the terms:

$$\gamma_{act_{\hat{U}}} - M_{act}\hat{U}^*_{unc} = \gamma_{act_{\delta\hat{U}}} - M_{act}\delta\hat{U}^*_{unc} \tag{3.46}$$

For the strict purpose of this proof, consider the active set being the entire constraint vectors (something which cannot be done in practice but is equally valid for the proof). By substituting the expressions for matrix M and each of the constraint vectors ($\gamma_{\hat{U}}$ and $\gamma_{\delta\hat{U}}$), results in:

$$\underbrace{\begin{bmatrix} U_{max} \\ -U_{min} \\ X_{max} - \bar{X} - D - G\delta x_0 + H\bar{U} \\ \bar{X} + D + G\delta x_0 - H\bar{U} - X_{min} \end{bmatrix}}_{\gamma_{\hat{U}}} - \underbrace{\begin{bmatrix} I \\ -I \\ H \\ -H \end{bmatrix}}_{M} \hat{U}^*_{unc} = \underbrace{\begin{bmatrix} U_{max} - \bar{U} \\ -U_{min} + \bar{U} \\ X_{max} - \bar{X} - D - G\delta x_0 \\ \bar{X} + D + G\delta x_0 - X_{min} \end{bmatrix}}_{\gamma_{\delta\hat{U}}} - \underbrace{\begin{bmatrix} I \\ -I \\ H \\ -H \end{bmatrix}}_{M} \delta\hat{U}^*_{unc} \tag{3.47}$$

Given we have proved the unconstrained optimal solutions to be the same (ie. $\hat{U}^*_{unc} = \bar{U} + \delta\hat{U}^*_{unc}$), by evaluating the relevant matrix expressions and grouping terms this reduces to:

$$\begin{aligned} U_{max} - \boxed{\hat{U}^*_{unc}} &= U_{max} - \boxed{(\bar{U} + \delta\hat{U}^*_{unc})} \\ \boxed{\hat{U}^*_{unc}} - U_{min} &= \boxed{\bar{U} + \delta\hat{U}^*_{unc}} - U_{min} \\ X_{max} - \bar{X} - D - G\delta x_0 - H\boxed{(\hat{U}^*_{unc} - \bar{U})} &= X_{max} - \bar{X} - D - G\delta x_0 - H\boxed{\delta\hat{U}^*_{unc}} \\ \bar{X} + D + G\delta x_0 + H\boxed{(\hat{U}^*_{unc} - \bar{U})} - X_{min} &= \bar{X} + D + G\delta x_0 + H\boxed{\delta\hat{U}^*_{unc}} - X_{min} \end{aligned} \tag{3.48}$$

where relevant equality terms are signaled by dashed boxes which concludes the proof.

Notice this proof also holds for single-shooting scenarios where the term $D = \mathbb{O}$. Moreover, it is trivial to show that the solution will also be exactly the same as in the Linear MPC case simply by observing that the term $\bar{X} + D + G\delta x_k - H\bar{U}$, present in the non-relative solution are the predictions of the state when $\hat{U} = O$, thus represent what is commonly known as the "free-response" [19], ie. $\bar{X} + D + G\delta x_k - H\bar{U} = Gx_k$.

A similar procedure will be used in chapters 4 and 6 for proving certain properties of interest of the proposed approaches. □

3.2 The Real-Time Iteration Scheme

To achieve real-time performance of the optimisation, one of the most successful approaches is the Real-Time Iteration Scheme, originally developed in [31], which exploits the fact that NMPC is required to successively solve OCPs which are closely linked to each other [55]. The latter is briefly summarised in this section, and for more details, the reader is referred to [55] which gives an excellent tutorial like paper of this method.

The scheme consists of 3 main strategies for the multiple-shooting approach:

1. Initial Value Embedding:

   It uses a shifted version of the solution for the nominal state, input (and if required output) trajectories obtained in the previous time step to hot-start the trajectories over which the SQP will linearise, typically duplicating the last input $\bar{u}_{k+N_p-1|k} = \hat{u}^*_{k+N_p-2|k-1}$, and shifting the state $\bar{X}_{k|k-1} = [\hat{x}^T_{k+1|k-1}, \ldots, \hat{x}^T_{k+N_p-1|k-1}, \hat{x}^T_{k+N_p|k}]^T$, where the last state (and if required, the last output) is typically obtained from a simple simulation $\hat{x}_{k+N_p|k} = f(\bar{x}_{k+Np-1|k-1}, \bar{u}_{k+N_p-2|k-1})$, ie. using the single shooting philosophy, but only at that step [28]. Moreover, in the case of optimisation methods based on active-set strategies, it also uses a shifted version of the vector of Lagrange multipliers (λ) obtained in the previous step. The shifting procedure of this vector will depend on the particular "organisation" used for the constraint matrices and vectors, (M and γ), and so it will vary from case to case, but the important part is to make a "consistent shifting" of them such that they represent the "same" active-set of constraints relative to the next step.

2. Single SQP Iteration:

   It performs only a single SQP iteration given the hot-started trajectory is expected to be close, provided no significant disturbances have entered the system. Assuming the latter and other conditions discussed in [31, 55] such as smooth reference changes, as well as starting at a global optimum are satisfied, the scheme can guarantee nominal closed loop stability. Moreover, the approach typically takes a full Newton step [55], and one has to be satisfied with local optimallity.

3. Computation Separation:

   It separates the computations required for the optimisation into preparation and feedback phases to reduce the computation delay required by the optimisation and avoid solving a problem that is "only getting older" [115], also referred to as the "Real-Time Dilemma" in [55]. It can also be seen as a "distributed-in-time" optimisation procedure [115]. An excellent diagram presenting the timelines of this methodology is given in figure 3 of [55]. The tasks of each phase are as follows:

(a) Preparation Phase: In between sampling times $k-1 \rightarrow k$, it uses the predicted state for the next sampling time ($\bar{x}_0 = \hat{x}_{k|k-1}$) as an initial condition of (3.1) which enables the computation of all the matrices and vectors required by the optimisation (D, G, H, E, M), and if appropriate, the partial calculation of (f and γ) given the dependency on δx_0.

(b) Feedback Phase: As soon as the state (x_0) becomes available, either through measurement or estimation, it calculates the initial deviation ($\delta x_0 = x_0 - \bar{x}_0$), completes the calculation of f and γ, and solves the QP. It is this particular term (δx_0) which gives the whole approach the ability to reject disturbances and have an inherent robustness to prediction errors. After obtaining the solution, the "expansion step" described in the previous subsection must be taken to calculate the predicted state (and if required, the output) for the next iteration.

A slightly different approach that captures the essence of the "feedback" phase was also implemented and experimentally tested in this thesis, in particular in the double inverted pendulum work presented in chapter 6, section 6.6. The aforementioned approach further avoids any possible delay related to the solution of the QP as this task alone can prove to be quite expensive by itself, particularly if there are significant active set changes from one step to the next, thus being unable to take advantage of the hot-starting capabilities of active set methods.

The approach goes as follows: If we assume that the predicted state ($\bar{x}_0 = \hat{x}_{k|k-1} = f(x_{k-1}, u_{k-1})$) will be the actual state measured (or estimated), then the initial state deviation will be zero $\delta x_0 = 0$, thus we can already proceed to solve the QP during the preparation phase where a vector of Lagrange Multipliers ($\bar{\lambda}$) can be found. Having obtained this vector, and assuming that it won't be significantly affected by the actual value δx_0 (noting that active set changes can also happen without this term due to the term D or $H\delta\hat{U}$ from the previous iteration, or simply because of reference changes), a solution can be found that re-includes the initial state deviation (δx_0) to result in an "unconstrained" correction to the "approximated constrained" solution.

Following the the procedure in [146], the solution to both types of optimisation (relative and non-relative) will be given by (3.49) and (3.50), where each of the parts is clearly indicated.

1. Relative

$$\hat{U} = \bar{U} - E^{-1}\left[\underbrace{\overbrace{-(H^TQ(X_r - \bar{X}) - R(\bar{U} - U_r))}^{\text{Unconstrained}}\ \overbrace{+M^T\bar{\lambda}}^{\text{Constrained}}}_{\text{Preparation Phase}}\ \underbrace{+H^TQG\delta x_k}_{\text{Feedback Phase}}\right] \tag{3.49}$$

2. Non-Relative

$$\hat{U} = -E^{-1}\left[\underbrace{\overbrace{-H^TQ(X_r - \bar{X} + H\bar{U}) + RU_r}^{\text{Unconstrained}}\ \overbrace{+M^T\bar{\lambda}}^{\text{Constrained}}}_{\text{Preparation Phase}}\ \underbrace{+H^TQG\delta x_k}_{\text{Feedback Phase}}\right] \tag{3.50}$$

A general drawback of the RTI Scheme is that because a single SQP step is taken, the solution might be subject to approximation errors that arise from the linearisation, something which wouldn't happen if the standard SQP approach would be used where the dynamics are re-linearised over and over until convergence.

For a comparison between the standard NMPC and the RTI Scheme approach, refer to algorithms 2 and 3, as well as figure 5 of [55]. As stated earlier, the method can only guarantee local optimality if certain conditions such as no reference or state jumps, and starting at the global optimum are satisfied, as described in section 4.4 of [55]. In addition to these condition, other tuning parameters such as sampling time, prediction horizon length, integrator accuracy and passing the reference trajectory in a "smart way" can significantly improve the response of the solution [55].

3.3 Algorithm Details and Auto-generation

In order to provide a clear procedure to implement the overall standard RTI NMPC approach, this section provides a set of algorithms that allow its application. Ultimately, all the algorithms included in this section were implemented in efficiently auto-generated C/C++ codes based on the Eigen 3 library for evaluation against the ACADO toolkit as discussed in the introduction chapter of this thesis.

For simplicity, only the algorithm for implementing the relative framework of the "state-only" cost function will be provided, although the modifications required to implement the non-relative framework and/or "state-and-output" cost functions, are quite minimal.

3.3.1 Rederivation of the $O(N^2)$ and $O(N)$ algorithms

Before providing the final RTI preparation and feedback algorithms, it is important to understand the underlying pattern of certain operations beyond their mathematical representation which could allow the development of key algorithms that would speed-up the whole operation of the implementation. A key example of this are the $O(N^2)$ and $O(N)$ algorithms from the Ph.D. thesis [8] which are absolutely essential given that they perform the efficient calculation of the Hessian term H^TQH, and linear term $H^TQ(X_r - X - D - G\delta x_0)$ through a recursive-like operation that takes advantage of the known lower-block-triangular structure of matrix H to avoid the zero terms computations, as well as any repeated terms that might result from the direct calculation.

As the derivation presented in [8] is slightly unclear given it skips a few steps along the way, we present a re-derivation of this methodology in this subsection bearing in mind that we will require a similar procedure for the extensions of this algorithms presented in chapters 5, 6 and 8.

To understand the fundamental operation of these algorithm, let us consider a system with a short horizon of $N_p = 3$ that will allow us to derive both of these algorithms. The resulting hessian operation H^TQH for this optimisation would be given by:

$$\begin{aligned} E &= \underbrace{\begin{bmatrix} B_0 & O & O \\ A_1B_0 & B_1 & O \\ A_2A_1B_0 & A_2B_1 & B_2 \end{bmatrix}^T}_{H^T} \underbrace{\begin{bmatrix} q_1 & O & O \\ O & q_2 & O \\ O & O & q_3 \end{bmatrix}}_{Q} \underbrace{\begin{bmatrix} B_0 & O & O \\ A_1B_0 & B_1 & O \\ A_2A_1B_0 & A_2B_1 & B_2 \end{bmatrix}}_{H} \\ &= \begin{bmatrix} E_{1,1} & E_{1,2} & E_{1,3} \\ E_{2,1} & E_{2,2} & E_{2,3} \\ E_{3,1} & E_{3,2} & E_{3,3} \end{bmatrix} \end{aligned} \tag{3.51}$$

For a starting point, note that the operation ($\tilde{W} = QH$), momentarily assigned to a new "dummy" variable as in [8], is simply given by:

$$\tilde{W} = \begin{bmatrix} \tilde{w}_{1,1} & O & O \\ \tilde{w}_{2,1} & \tilde{w}_{2,2} & O \\ \tilde{w}_{3,1} & \tilde{w}_{3,2} & \tilde{w}_{3,3} \end{bmatrix} = \begin{bmatrix} q_1 B_0 & O & O \\ q_2 A_1 B_0 & q_2 B_1 & O \\ q_3 A_2 A_1 B_0 & q_3 A_2 B_1 & q_3 B_2 \end{bmatrix} \tag{3.52}$$

which in the specific case that the weights (q_k) are purely diagonal, can be calculated using the optimised "asDiagonal()" product operation available in the Eigen 3 library.

Let us now look only at the first column of the Hessian (E), ie. the term given by:

$$\begin{bmatrix} E_{1,1} \\ E_{2,1} \\ E_{3,1} \end{bmatrix} = \begin{bmatrix} B_0^T & B_0^T A_1^T & B_0^T A_1^T A_2^T \\ O & B_1^T & B_1^T A_2^T \\ O & O & B_2^T \end{bmatrix} \begin{bmatrix} \tilde{w}_{1,1} \\ \tilde{w}_{2,1} \\ \tilde{w}_{3,1} \end{bmatrix} \tag{3.53}$$

The method is based on iteratively modifying the terms of the W matrix backwards starting from the term in the last row ($w_{3,1}$) which allows a straight forward calculation of the term $E_{3,1}$. Therefore, to understand the algorithm, let us begin by calculating this term which is given by:

$$E_{3,1} = B_2^T \tilde{w}_{3,1} \tag{3.54}$$

Notice the following term ($E_{2,1}$) can be calculated as a function of $\tilde{w}_{3,1}$ by:

$$\begin{aligned} E_{2,1} &= B_1^T \tilde{w}_{2,1} + B_1^T A_2^T \tilde{w}_{3,1} \\ &= B_1^T \underbrace{(w_{2,1}^{[0]} + A_2^T \tilde{w}_{3,1}^{[0]})}_{w_{2,1}^{[1]}} \\ &= B_1^T \tilde{w}_{2,1}^{[1]} \end{aligned} \tag{3.55}$$

where the notation $\tilde{w}_{k,j}^{[0]}$ and $w_{k,j}^{[1]}$ indicate the initial value, and the value after the modification required by the algorithm.

Following a similar procedure, the last term ($E_{1,1}$) can be calculated as a function of $\tilde{w}_{2,1}^{[1]}$ (and consequently of $\tilde{w}_{3,1}^{[0]}$) by:

$$\begin{aligned} E_{1,1} &= B_0^T \left(\underbrace{\tilde{w}_{1,1}^{[0]} + A_1^T \overbrace{(\tilde{w}_{2,1}^{[0]} + A_2^T \tilde{w}_{3,1}^{[0]})}^{\tilde{w}_{2,1}^{[1]}}}_{\tilde{w}_{1,1}^{[1]}} \right) \\ &= B_0^T \tilde{w}_{1,1}^{[1]} \end{aligned} \tag{3.56}$$

Thus a clear pattern can be seen where the Hessian can be calculated by recursively modifying the terms with an expression like $\tilde{w}_{k,i}^{[1]} = \tilde{w}_{k,i}^{[0]} + A_k^T \tilde{w}_{k+1,i}^{[1]}$ and subsequently calculating the Hessian term as $E_{k,i} = B_{k-1}^T \tilde{w}_{k,i}^{[1]}$.

Notice this methodology is applicable for calculating any of the other columns of $\tilde{W}$, however, given that is known that the "Hessian" is symmetric, the final algorithm only calculates the lower triangular part of the Hessian and copies the rest of the terms. Moreover, this exact same procedure can be used to derive the $O(N)$ algorithm from [8] used for calculating the linear term $-H^T Q(X_r - \bar{X} - D - G\delta x_0) = -H^T Q X_e$ assuming the operation $\tilde{W} = -QX_e$ is captured by the "dummy" variable. This again can be computed efficiently using the "asDiagonal()" method of the Eigen 3 library. Moreover, the procedures developed above lacked the Hessian term R, and the linear term $R(\bar{U} - U_r)$ to avoid confusion. However, notice these terms can be easily included in the overall procedure with very small modifications. Thus, the final $O(N^2)$ and $O(N)$ algorithms which include these calculations are given in algorithms 3.2 and 3.3.

Algorithm 3.2: Standard Condensing $O(N^2)$ Algorithm

Data: H, Q, R, A_k, B_k, N_p

begin
  for $i = 1$ **to** N_p **do**
    $\tilde{w}_{N_p,i} = q_{N_p} h_{N_p,i}$; // Initial value of the dummy variable
    // For loop running backwards $k = N_p, N_p - 1, \cdots, i + 1$
    for $k = N_p$ **to** $i + 1$ **do**
      $E_{k,i} = B_{k-1}^T \tilde{w}_{k,i}$; // Calculate Hessian term component
      $\tilde{w}_{k-1,i} = q_{k-1} h_{k-1,i} + A_{k-1}^T \tilde{w}_{k,i}$; // Propagate recursively
      $E_{i,k} = E_{k,i}^T$; // Copy transpose
    end
    $E_{i,i} = B_{i-1}^T \tilde{w}_{i,i} + r_i$; // Diagonal term calculation
  end
end

Result: E

Algorithm 3.3: Standard Condensing $O(N)$ Algorithm

Data: $Q, R, A_k, B_k, X_e, \bar{U}, U_r, N_p$

begin
  $\tilde{w}_{N_p} = q_{N_p} X_{e_{N_p}}$; // Initial value of the dummy variable
  // For loop running backwards $k = N_p, N_p - 1, \cdots, 2$
  for $k = N_p$ **to** 2 **do**
    $f_k = -B_{k-1}^T \tilde{w}_k + r_{k-1}(\bar{u}_{k-1} - u_{r_{k-1}})$; // Calculate linear term component
    $\tilde{w}_{k-1} = q_{k-1} X_{e_{k-1}} + A_{k-1}^T \tilde{w}_k$; // Propagate recursively
  end
  $f_1 = -B_0^T \tilde{w}_1 + r_0(\bar{u}_0 - u_{r_0})$; // Initial term outside the loop
end

Result: f

3.3.2 Core Algorithms

In addition to the basic $O(N^2)$ and $O(N)$ algorithms, there are 4 additional "core" algorithms that will form part of the final RTI NMPC approach composed by the "preparation" and "feedback" phases which are (given in order introduction and usage): 1. Forward Simulation algorithm 2. H Matrix calculation 3. D Vector calculation 4. $H\delta\hat{U}$ Expansion Step.

The Forward Simulation algorithm is perhaps one of the most fundamental parts of the whole approach which focuses on the computation of all the linearisation matrices (A_k, B_k), as well as the simulation of the system for the calculation of the state offsets (d_k) required by the multiple shooting approach. This task can be performed by algorithm 3.4. The algorithm essentially uses the previously introduced Explicit Euler algorithm 3.1, although as mentioned earlier, one can use any type of integrator that provides the required information (eg. RK4, implicit approaches, etc) for this step. The $\tilde{x}_k$ variable used in the algorithm represent a "dummy" variable that essentially captures the "simulated" value, ie. $x_{k+1} = f(x_k, u_k)$. Moreover, notice the algorithm considers the single shooting case in which $\bar{x}_k = \tilde{x}_k$, thus making the d_k zero.

Algorithm 3.4: Forward Propagation Algorithm

Data: $\bar{X}, \bar{U}, \bar{x}_0, N_p$
begin
  for $k = 1$ **to** N_p **do**
    // Simulation and Linearisation, eg. using algorithm 3.1
    $[\tilde{x}_k, A_{k-1}, B_{k-1}] = Simulate\_And\_Linearize(\bar{x}_{k-1}, \bar{u}_{k-1})$
    if *Single Shooting=1* **then**
      $\bar{x}_k = \tilde{x}_k$; // Special case for the single shooting case
    end
    $d_k = \tilde{x}_k - \bar{x}_k$; // Calculate individual offset at each step
  end
end
Result: A_k, B_k, d_k

Following this, another fundamental part is the computation of the H matrix which can be efficiently obtained by the algorithm:

Algorithm 3.5: Standard Condensing H Matrix Calculation

Data: A_k, B_k, N_p
begin
  for $i = 1$ **to** N_p **do**
    $h_{i,i} = B_{i-1}$; // Calculate initial value in the diagonal
    for $k = i + 1$ **to** N_p **do**
      $h_{k,i} = A_{k-1} h_{k-1,i}$; // Propagate recursively
    end
  end
end
Result: H

The reader might be wondering why the G matrix has not been calculated. What is important to note is that this matrix is related to the initial "offset" or "state deviation" (δx_0), which for all practical purposes represents exactly the same type of offset as d_k given both are propagated through the state-transition dynamics matrix A_k. Although it is convenient to have this initial offset "separated" from the rest (ie. the D vector) to be able to obtain closed loop expressions such as (3.49), this is not strictly required (and indeed, it is not how its operated in the ACADO toolkit) and one can directly embed the initial offset into the "first" offset by $\bar{d}_1 = A_0\delta x_0 + d_1$. As a result, a decent number of computations are avoided by not calculating the G matrix and simply including the initial offset in the D matrix. This task can be done by using algorithm 3.6.

Algorithm 3.6: Standard Condensing D Vector Calculation

Data: $A_k, d_k, \delta x_0, N_p$
begin
  $\bar{d}_1 = A_0\delta x_0 + d_1$; // Initial value calculate with δx_0
  for $k = 2$ **to** N_p **do**
    $\bar{d}_k = A_{k-1}\bar{d}_{k-1} + d_k$; // Propagate recursively
  end
end
Result: D

Finally, as explained previously, a key part required by the multiple shooting approach is the "expansion" step ($\bar{X} = \bar{X} + D + H\delta\hat{U}^*$), performed by the function "acado_expand()" in the auto-generated codes from the ACADO toolkit. Notice to perform this operation the algorithm only requires the efficient calculation of the term $\delta\tilde{X} = H\delta\hat{U}^*$ (assigned a new "dummy" variable) as the other terms are simple column-vectors. To calculate this, the user can use algorithm 3.7 which propagates the terms in $\delta\hat{U}^*$ recursively through the dynamics $\delta\tilde{x}_k = A_{k-1}\delta\tilde{x}_{k-1} + B_{k-1}\delta\hat{u}^*_{k-1}$, thus avoiding the zero terms present in the H matrix.

Algorithm 3.7: Standard Condensing $\delta\tilde{X} = H\delta\hat{U}^*$ Expansion

Data: $A_k, B_k, \delta\hat{U}^*, N_p$
begin
  $\delta\tilde{x}_1 = B_0\delta\hat{u}^*_0$; // Initial value
  for $k = 2$ **to** N_p **do**
    $\delta\tilde{x}_k = A_{k-1}\delta\tilde{x}_{k-1} + B_{k-1}\delta\hat{u}^*_{k-1}$; // Propagate recursively
  end
end
Result: $\delta\tilde{X}$

3.3.3 RTI Algorithms

Having established all the aforementioned "core" algorithms, the final RTI NMPC is finally defined by the preparation and feedback algorithms, (3.8) and (3.9) respectively, which are based in terms of the previous algorithms to simplify the verification process of each working part of the overall algorithm.

Algorithm 3.8: Standard RTI NMPC Preparation Phase Algorithm

Data: $\bar{X}, \bar{U}, \bar{\lambda}, x_{-1}, u_{-1}, Q, R, N_p$

begin

$\bar{x}_0 = f(x_{-1}, u_{-1})$; // Calculate predicted state from previous state and input

Shift $\bar{X}$,$\bar{U}$, and optionally $\bar{\lambda}$ consistently ; // Initial Value Embedding

$[A_k, B_k, d_k] = Forward(\bar{X}, \bar{U}, \bar{x}_0, N_p)$; // Run algorithm 3.4

$[H] = CalculateH(A_k, B_k, N_p)$; // Run algorithm 3.5

$[E] = CalculateE(H, Q, R, A_k, B_k, N_p)$; // Run algorithm 3.2

$M = \begin{bmatrix} I & -I & H^T & -H^T \end{bmatrix}^T$; // Form M Matrix

end

Result: $E, M, A_k, B_k, d_k, \bar{x}_0$

Algorithm 3.9: Standard RTI NMPC Feedback Phase Algorithm

Data: $x_0, \bar{x}_0, \bar{X}, \bar{U}, \bar{\lambda}, X_r, U_r, E, M, A_k, B_k, d_k, Q, R, N_p, U_{max}, U_{min}, X_{max}, X_{min}$

begin

$\delta x_0 = x_0 - \bar{x}_0$; // Calculate state deviation from measurement

$[D] = FormD(A_k, d_k, \delta x_0, N_p)$; // Run algorithm 3.6

$X_e = X_r - \bar{X} - D$; // Calculate X error

$[f] = Calculatef(Q, R, A_k, B_k, X_e, \bar{U}, U_r, N_p)$; // Run algorithm 3.3

$\gamma = \begin{bmatrix} U_{max} - \bar{U} \\ \bar{U} - U_{min} \\ X_{max} - \bar{X} - D \\ \bar{X} + D - X_{min} \end{bmatrix}$; // Calculate constraint vector γ

$[\delta\hat{U}^*, \bar{\lambda}] = QPSolve(E, f, M, \gamma, \bar{\lambda})$; // Solve the Quadratic Program

$\bar{U} = \bar{U} + \delta\hat{U}^*$; // Calculate new nominal input

$[\delta\tilde{X}] = Expand(A_k, B_k, \delta\hat{U}^*, N_p)$; // Run algorithm 3.7

$\bar{X} = \bar{X} + D + \delta\tilde{X}$; // Calculate new nominal state

end

Result: $\bar{X}, \bar{U}, \bar{\lambda}$

3.3.4 Auto-generation

An important contribution of this thesis is on the popular topic of "auto-generation", available for example in the ACADO toolkit [66], where the user can express a given OCP in terms of "symbolic" variables, expressions and nonlinear functions, as well as penalisation terms and constraints which are then translated by an "auto-generation" routine to automatically coded C/C++ routines that include declarations of functions as well as matrices and vectors of specific sizes that implement the overall RTI approach for that specific OCP. The ACADO auto-generation routine includes automatic differentiation process typically based on CasADi (Computer Algebra Systems for Algorithmic Differentiation and integration) [8], thus relieving the user from providing the underlying expressions required for the linearisation matrices such as A_k, B_k or C_k, which by itself can be quite a daunting task for complex nonlinear systems. In our case, the symbolic toolbox from Matlab was used in combination with the Matlab Coder toolbox which allows writing expression in C/C++ notation. Moreover, although most of the required steps can be done by a "generic" implementation of algorithms 3.9 and 3.8, a key step performed by the ACADO toolkit is the unrolling of the required for-loops to avoid any unnecessary

additional calculation related to pointers or indexing variables of the respective matrices, although this may be avoided simply by activating the $-O3$ C optimisation flag when compiling the C/C++ codes which performs optimised for-loop unrolling. Finally, one important thing that the ACADO lacked was the use of linear algebra routines readily available in popular packages such as LAPACK or BLAS to perform matrix-matrix or matrix-vector operations efficiently, and instead relied on tailored auto-generated matrix-matrix and matrix-vector functions to achieve an overall "self-contained" solver and "reduce the software maintenance effort" [64]. The recent development of ACADOS [143] tackled this issue by using the Basic Linear Algebra Subroutines For Embedded Optimisation (BLASFEO) implementation. In our case, the solvers developed in this thesis make use of the Eigen 3 library which offers competitive performance with extremely clean and expressive C++ templates (similar to MATLAB), as well as supports AVX (Advanced Vector Extension) vectorisation instructions and FMA (Fused Multiply-Addition) operations [15] which overall were observed to speed the code significantly, and resulted in faster solvers when compared to the original ACADO toolkit using the standard RTI NMPC case detailed in this chapter as seen for example in tables 5.12 or 6.6.

3.3.5 Generic Computations

To be able to evaluate the computational performance of the RTI NMPC approach of this chapter as a whole, one would typically need to define a specific dynamical system to analyse and implement all the algorithms presented above to be able to measure computation times. Thus, the resulting overall computational performance (or "speed") would inevitably vary from case to case, for example depending on the initial conditions of the whole simulation, the time it takes to simulate and linearise the system, the type of QP solver used, the number of active-set changes that an active-set Quadratic Program requires, etc. However, there are a few key operations and algorithms which DO NOT depend on this such as the computation of algorithms 3.2, 3.3, 3.5, 3.6 and 3.7. Hence, we can test their computational performance using random or "generic" matrices (A_k, B_k) for given optimisation sizes (N_p, n_x, n_u) of interest without requiring to define a specific dynamical system, resulting in a "generic computations" analysis. This is something that we will use throughout the thesis to compare certain algorithms against the standard algorithms introduced in this chapter to demonstrate the inherent advantages (or in some cases, disadvantages) that the proposed approaches have.

3.4 Stability, Recursive Feasibility and Convergence of NMPC

One of the most important properties in the topic Model Predictive Control in general (linear or non-linear) is stability [28, 122], particularly if one is interested in a stabilisation problem, with the understanding that the NMPC optimisation framework can also be used for non-stabilising problems such as the Wave Energy Conversion device presented in chapter 5. In addition to this basic property, another important property is "recursive feasibilty" which is the ability to obtain a feasible solution after having obtained one in the previous step [94]. To guarantee both of these properties, one would ideally solve an "infinite horizon" problem where $N_p \rightarrow \infty$ which would allow you to obtain a "stationary" (non-changing) optimal control sequence or "plan" that automatically guarantees a decrease in the overall cost function from step to step $(J_{k+1} < J_k)$ resulting in Lyapunov stability [28], and guarantees

recursive feasibility simply by being able to follow the stationary plan. In practice, however this is not only intractable, given the resulting optimisation would require a significantly higher time which would prevent the user to implement it in real-time, but also improbable, given that real systems typically contain a relatively large amount of uncertainty which would cause the optimisation to "change the plan" at each time step. Instead, one typically has to select a finite horizon (N_p), which brings the question of *how can we guarantee* ***nominal*** *stability and recursive feasibility using a finite horizon?*, ie. stability and recursive feasibility of the optimisation WITHOUT uncertainty, as it would be the least basic requirement for the overall methodology to be relevant in theory.

On the simplest case one can simply test or "tune" the optimisation for different prediction horizons and weights, and the properties can be investigated "after" the optimal solution is obtained (sometimes referred as "a-posteriori" [18]). Alternatively, one can embed the desired stability and recursive feasibility characteristics into the optimisation "before" even obtaining a solution (sometimes referred as "a-priori"), thus resulting in an optimisation that "by design" has the properties of interest. Two of the most well known methods for guaranteeing these properties are "zero-terminal constraints" [28], and "terminal weights" (sometimes called terminal-modes or dual-mode [122]) with invariant sets, which we will discuss in the following.

3.4.1 Zero-Terminal Constraints

In the case of "zero-terminal constraints", an equality constraint is imposed in the last state of the predictions in the form of $\hat{x}_{k+N_p} = 0$, which can be done by simply selecting the last row of the linearised prediction model (3.17a) and embedding the equality in the optimisation. To prove nominal stability in this scenario let us consider the following. Without loss of generality, consider that the nonlinear systems dynamics evaluated at the origin stays at the origin, ie. $(f(0,0) = 0)$, with the overall cost function targeting the origin (ie. no state or input references $X_r = 0, U_r = 0$) and the non-linear optimisation run until convergence under second order necessary conditions using an appropriate step-size selection scheme which ensures proper nonlinear-optimisation [31]. By embedding the aforementioned equality, and assuming the optimisation starts at a converged optimal feasible solution, the very first optimisation, ie. the cost at time k (J_k) would result in an "optimal" feasible sequence of predicted states and inputs, eg. given by $[\hat{x}^*_{k+1|k}, \hat{u}^*_{k|k}, \hat{x}^*_{k+2|k}, \hat{u}^*_{k+1|k}, \cdots, \hat{u}^*_{k+N_p-1}, 0]$, for which only the first input ($u^*_{k|k}$) would be applied, thus resulting (under nominal conditions) in the predicted state $\hat{x}^*_{k+1|k}$. Note that the value of this cost function (J_k) depends, not only on x_k, but also on where the solution was started due to the existence of local minimas. This makes obtaining rigorous stability guarantees of nonlinear system a much harder problem given the solution can no longer be written as a Lyapunov function as discussed in [117]. Nonetheless, under ideal conditions, some sense of stability can be found when the solution is properly hot-started at the following time step $(k+1)$. Notice that the costs related to the first state/input pair $(\hat{x}^*_{k+1|k}, \hat{u}^*_{k|k})$ at this time step are now disregarded, thus decreasing the overall cost precisely by $J_{k+1} = J_k - {x^*_{k+1|k}}^T Q x^*_{k+1|k} - {u^*_k}^T R u^*_k$. Moreover, given that the previous solution can be used, the equality $(x_{k+N_p+1|k+1} = 0)$ is feasible and does not add any additional cost. Furthermore, because the initial equality $(x_{k+N_p|k+1} = 0)$ has now been "relaxed", the optimisation can now do "better", and consequently will give a decrease of AT LEAST $J_{k+1} \leq J_k - {x^*_{k+1|k}}^T Q x^*_{k+1|k} - {u^*_k}^T R u^*_k$ as discussed in [28], resulting in nominal stability.

One important thing to understand is that when using this type of methodology, the resulting closed loop will be sub-optimal when compared to the "original" infinite horizon optimisation (or "target"), or even when compared to the cost without the terminal constraint, and may lead to "ill-posed" solutions where the "plan" made at a given step may differ significantly from the next [122].

To illustrate this, let us consider the following example:

Example 3.2. *Zero-Terminal Constraint Sub-optimality and Ill-Posedness*

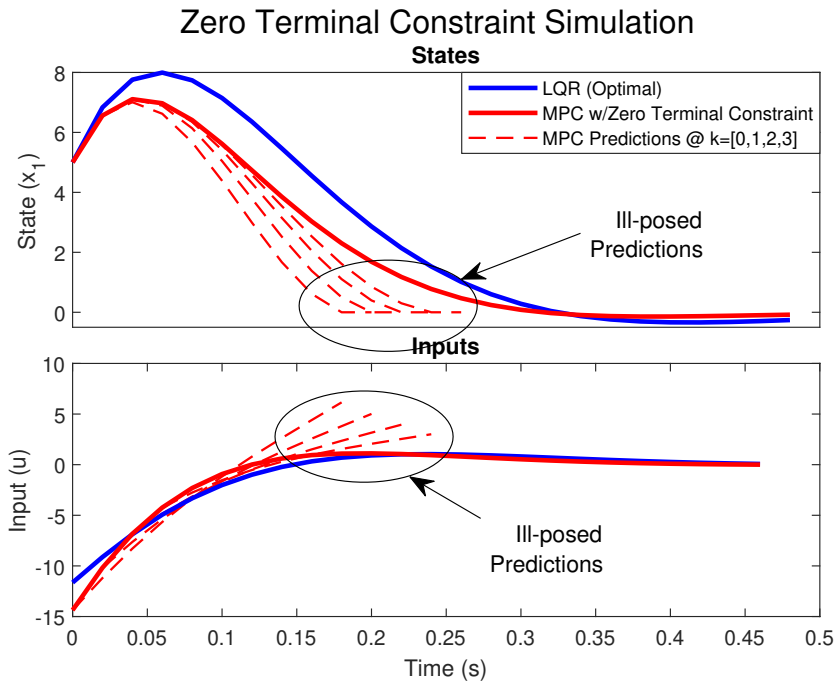
Consider the optimisation of the following discrete-time state space $x_k = [x_1, x_2]^T$ given by:

$$x_{k+1} = \underbrace{\begin{bmatrix} 2 & -1 \\ 1 & 0 \end{bmatrix}}_{A_k} x_k + \underbrace{\begin{bmatrix} 0.1 \\ 0 \end{bmatrix}}_{B_k} u_k \tag{3.57}$$

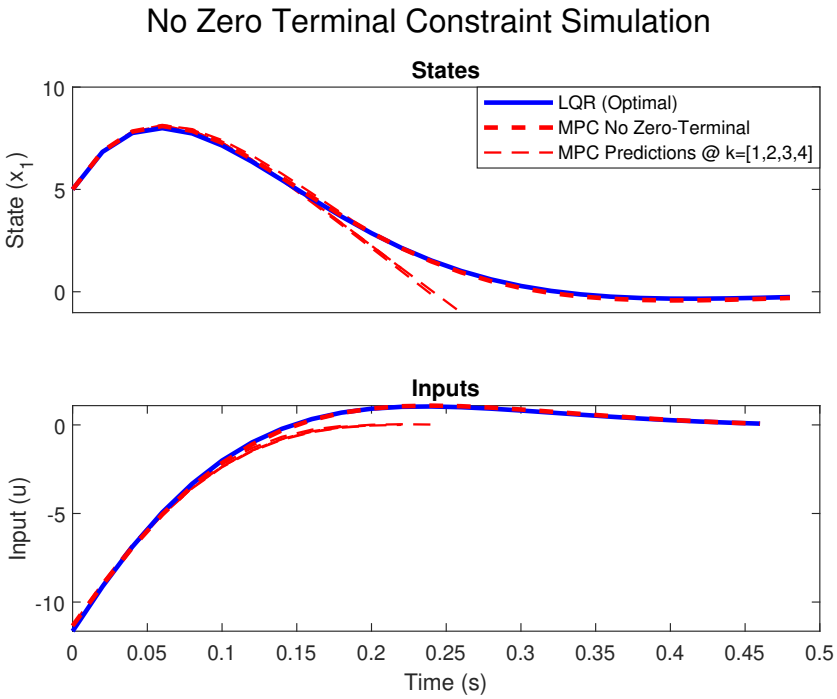
subject to the penalisation weights $q_k = diag([1, 1]), r_k = 5$, with a prediction horizon of $N_p = 10$, and an initial condition of $x_0 = [5, 2]^T$ (No input or state constraints to keep the example simple).

Figure 3.3a presents a comparison between the optimal closed loop infinite horizon solution obtained via the standard Linear Quadratic Regular (LQR), depicted by the blue line; and the closed loop performance that results from implementing the zero-terminal constraint approach in this system, depicted by the red line. From this figure it can be seen how the open loop predictions of the approach, depicted by the red-dashed lines which clearly present the embedded zero-terminal constraint, differ moderately from the final closed loop trajectory (ie. the red line), thus resulting in an "ill-posed" prediction/optimisation framework [122] that essentially represent a "meaningless" optimisation where the predicted optimal is not even close to the closed-loop optimal. Moreover, it is worthwhile noting that the total costs (J) of both simulated scenarios (ie. the red and blue trajectories) resulted in $J_{LQR} = 2,431$ and $J_{zero} = 2,538$, thus giving a sub-optimallity of around $\Delta J_{zero} = \left(\frac{J_{zero}}{J_{LQR}} - 1\right) \times 100 = 4.42\%$. What is most interesting is that if we simply optimise using the standard cost, ie. without the zero-terminal constraint; without the "a-priori"/"by design" stability, the closed loop solution also results in a stabilising feedback control law. The resulting behaviour can be seen in figure 3.3b where the closed loop solution, depicted by the red thick dashed line, is almost indistinguishable from the LQR solution (blue line) resulting in a sub-optimality of $\Delta J_{no-zero} = 0.06\%$; and the open loop predictions being somewhat "closer" to the closed loop response. This behaviour is to expected given that the zero-terminal constraint essentially moves the optimal "as far away as necessary" from the solution without the terminal constraint (in this case, the unconstrained solution) to satisfy the requested condition, even if this implies having to "over-react", something which can be appreciated from the initial input of figure 3.3a being "bigger", ie. having a greater cost. As a result, the optimisation can give rather poor performance and have significantly reduced feasibility "volumes" when using this type of approach.

The key takeaway from this example is that even though the method has the desired basic properties, it presents a potential disadvantage which can be avoided by using the terminal weight methodology described in the following subsection.



(a) Zero Terminal Constraint MPC



(b) No Zero Terminal Constraint MPC

Figure 3.3: Example Comparison of Zero vs No Zero Terminal Constraint Solution

3.4.2 Terminal Weights, Invariant Sets and Dual Modes

An alternative way to ensure nominal stability and recursive feasibility of the closed-loop solution, and indeed one of the most competitive approaches, is by including a combination of a terminal weight, sometimes called infinite horizon costing [66]; and a terminal set of inequality constraints, sometimes called the Maximum Admissible Set (MAS) [122] which achieve the same property but are less restrictive [28]. The problem is therefore a two-fold part, each of which deserves it own separate explanation which we will discuss in the following.

Terminal Weights, Infinite Horizon Costing and Dual Modes

The idea of infinite horizon costing is based on Bellman's "principle of optimality" which set the foundation for the Dynamic Programming approach back in the 1950s [28]. This principle essentially states that each step of the predictions ($k, k+1, k+2$, etc.) in an Optimal Control Problem is its own "subproblem", and we can solve the "total" Optimal Control Problem by optimising each step "separately" and connecting them with their immediate following step through a "cost-to-go" function that captures the "rest" of the optimal costs that emerge from that step on-wards. The simplest way of demonstrating this principle if through a re-derivation of the so called Discrete Algebraic Riccati Equation (DARE) [154] (an essential part of LQR), baring in mind that a similar procedure will be used in chapter 7 to derive the terminal weight for the approach proposed in chapter 5.

Without loss of generality, consider the optimisation of a linear state space with a cost function (J) connecting the quadratic costs of two subsequent steps (k and $k+1$) given by:

$$\min_{u_{k-1},x_k,u_k,x_{k+1}} \quad J = \underbrace{u_{k-1}^T R u_{k-1} + x_k^T Q x_k}_{J_k} + \underbrace{u_k^T R u_k + x_{k+1}^T P_{k+1} x_{k+1}}_{J_{k+1}} \tag{3.58a}$$

$$s.t. \quad x_{k+i} = A x_{k+i-1} + B u_{k+i-1} \quad \forall i = [0, 1] \tag{3.58b}$$

Note that this cost function is essentially unconstrained (no inequalities), hence we can apply the standard methods for unconstrained optimisation discussed subsection 3.1.3.

By substituting the system dynamics (3.58b) in the second stage (J_{k+1}) and grouping terms w.r.t. the decision variable of this stage (u_k) to formulate a standard QP we obtain:

$$\begin{aligned} \min_{u_k} \quad J_{k+1} &= (Ax_k + Bu_k)^T P_{k+1}(Ax_k + Bu_k) + u_k^T R u_k \\ &= u_k^T \underbrace{(B^T P_{k+1} B + R)}_{E_k} u_k + 2u_k^T \underbrace{(B^T P_{k+1} A x_k)}_{f_k} + A^T P_{k+1} A \end{aligned} \tag{3.59}$$

Considering x_k as an "initial condition" to this stage (J_{k+1}), the optimal solution for u_k is given by:

$$u_k^* = \min_{u_k}(J_{k+1}) = -E_k^{-1} f_k = -\underbrace{(B^T P_{k+1} B + R)^{-1} B^T P_{k+1} A}_{K_k} x_k = -K_k x_k \tag{3.60}$$

Note that this is true even if the cost of the first stage (J_k) were included in the optimisation.

Substituting this control law in the state dynamics (3.58b) results in the "closed loop" state dynamics given by:

$$x_{k+1} = \underbrace{(A - BK_k)}_{\Phi_k} x_k = \Phi_k x_k \tag{3.61}$$

By substituting this "new" state dynamics (3.61) and the optimal input (3.60) into (J_{k+1}) of the original cost function (3.58a), the second stage cost (J_{k+1}) can now be represented in terms of the "initial/previous state" (ie. the state to be optimised in the previous stage) exactly as:

$$J_{k+1} = x_k^T(\Phi_k^T P_{k+1}\Phi_k + K_k^T R K_k)x_k \tag{3.62}$$

By grouping terms w.r.t x_k of the original cost function with the new "closed loop" expression (3.62) results in alternative way of expressing "total" cost with only a single stage that "captures" the cost of the following stage in the form of:

$$J = J_k + J_{k+1} = u_{k-1}^T R u_{k-1} + x_k^T \underbrace{(Q + \Phi_k^T P_{k+1}\Phi_k + K_k^T R K_k)}_{P_k} x_k \tag{3.63}$$

where the term $P_k = Q + \Phi_k^T P_{k+1}\Phi_k + K_k^T R K_k$ is known as the Discrete Algebraic Riccati Equation (DARE) which is called a "dynamic recursion" [28].

A more commonly found expression for DARE is $P_k = Q + A^T P_{k+1} A - A^T P_{k+1} B K_k$ which can be found by proving:

$$P_k = Q + \underbrace{\Phi_k^T P_{k+1}\Phi_k + K_k^T R K_k}_{term1} = Q + \underbrace{A^T P_{k+1} A - A^T P_{k+1} B K_k}_{term2} \tag{3.64}$$

This reduces to proving that "$term1$" and "$term2$" are exactly the same which is given in the following:

$$\begin{aligned}
&\Phi_k^T P_{k+1}\Phi_k + K_k^T R K_k \\
&= (A - BK_k)^T P_{k+1}(A - BK_k) + K_k^T R K_k \\
&= A^T P_{k+1} A - 2A^T P_{k+1} B K_k + K_k^T (B^T P_{k+1} B + R) K_k \\
&= A^T P_{k+1} A - 2A^T P_{k+1} B K_k + A^T P_{k+1} B \cancel{(B^T P_{k+1} B + R)^{-1}} \cancel{(B^T P_{k+1} B + R)} K_k \\
&= A^T P_{k+1} A - A^T P_{k+1} B K_k
\end{aligned} \tag{3.65}$$

Based on this idea, we can obtain a "static" cost-to-go (typically denoted P_N) by iterating the DARE until convergence (eg. until $||P_k - P_{k+1}|| < \epsilon$) which represent the "infinite horizon cost" of the optimal unconstrained solution [28]. Thus, we can embed this cost-to-go into the optimisation by imposing it into the last state penalisation ($q_{k+N_p} = P_N$), which would capture the costs of the stages related to $[\hat{x}_{k+N_p}, \hat{u}_{k+N_p}, \rightarrow, \hat{x}_{k+\infty}, \hat{u}_{k+\infty}]$. This strategy is sometimes called the open-loop paradigm of "Dual-Mode" Predictive Control given this terminal cost represents a "secondary/dual static" controller of the form $u_{k+i} = -K_{LQR} x_{k+i}$ embedded in the steps $i = [N_p, N_p + 1, \cdots, \infty]$ [122] and uses an "open-loop" optimisation in the prediction horizon. We will introduce the closed loop paradigm later in chapter 6.

Note that we can use this cost-to-go in the context of NMPC to approximate the infinite horizon optimal solution around a steady state referent point (x_{ss}, u_{ss}) [66]. In the specific case where the initial optimisation enters a "terminal region" where the linear approximation is valid and can also be guaranteed recursive feasibility (something which can be tackled separately with invariant sets), the controller will follow the initial "infinite horizon optimal plan". A demonstration of this will be presented in example 3.3. This will automatically ensure a decrease in the costs of precisely $J_{k+1} = J_k - x_k^T Q x_k - u_k^T R u_k$ (in Linear MPC), thus allowing to prove nominal stability [28, 66, 122].

Remark 3.5. *Exact Terminal Weight for Linear MPC*

In contrast to nonlinear systems where the cost-to-go is only an approximation, in linear systems under nominal conditions it represents the exact infinite horizon cost to the desired precision.

Recursive Feasibility, Invariant Sets and Maximum Admissible Sets

In order to prove recursive feasibility of the optimisation when using the aforementioned terminal weights methodology, we require the use of invariant sets, defined as:

Definition 3.1. *Invariant Set*

Considering dynamical systems of the general form $x_{k+1} = f(x_k)$, a set of states $x_k \in \mathbb{S}$ is said to be an "invariant set" if for any initial condition $x_0 \in \mathbb{S}$, all future states that result from this condition propagated through the dynamical system remain inside the set, ie. $x_{k+1} = f(x_k) \in \mathbb{S} \quad \forall k = [0, 1, \cdots, \infty]$.

In the context of recursive feasibility, we can use this to define an invariant set which satisfies that for any feasible initial condition (x_0), all future state and inputs satisfy $x_{min} \leq x_k \leq x_{max}$ and $u_{min} \leq u_k \leq u_{max} \quad \forall k = [0, 1, \cdots \infty]$. In the specific case where the system enters the terminal region, we can use this concept to calculate a "Maximum Admissible Set" (MAS) which captures the "maximum" set of states inside the constraints that would satisfy this requirement, particularly when the "static" infinite horizon feedback control law, ie. $u_k = -K_{LQR} x_k$ is applied [122].

By embedding the resulting MAS into the terminal state inequality constraint ($x_{MAS-min} \leq \hat{x}_{k+N_p} \leq x_{MAS-max}$) along with the aforementioned terminal weights, the optimisation at any point can only find a solution that will satisfy, not only the states and inputs within the prediction horizon ($k \to k + N_p$), but all the future states and inputs that will emerge after it ($k + N_p \to \infty$) considering the unconstrained control law ($u_k = -K_{LQR} x_k$) was applied, thus guaranteeing recursive feasibility of the infinite horizon optimisation.

It is important to note that the MAS has not only one inequality constraint per state, but a set of inequalities that in combination capture the "overall requirements" for all future states and inputs to be satisfied [122]. Depending on the specific dynamical system and optimisation setup, this may increase the overall number of constraints significantly, although one can typically apply a reduction by disregarding the so called "redundant constraints".

Example 3.3. *Dual Mode Infinite Horizon Example*

To illustrate the performance that one can obtain from using both of this methodologies, consider the optimisation of the discrete system from example 3.2 using a prediction horizon of $N_p = 15$.

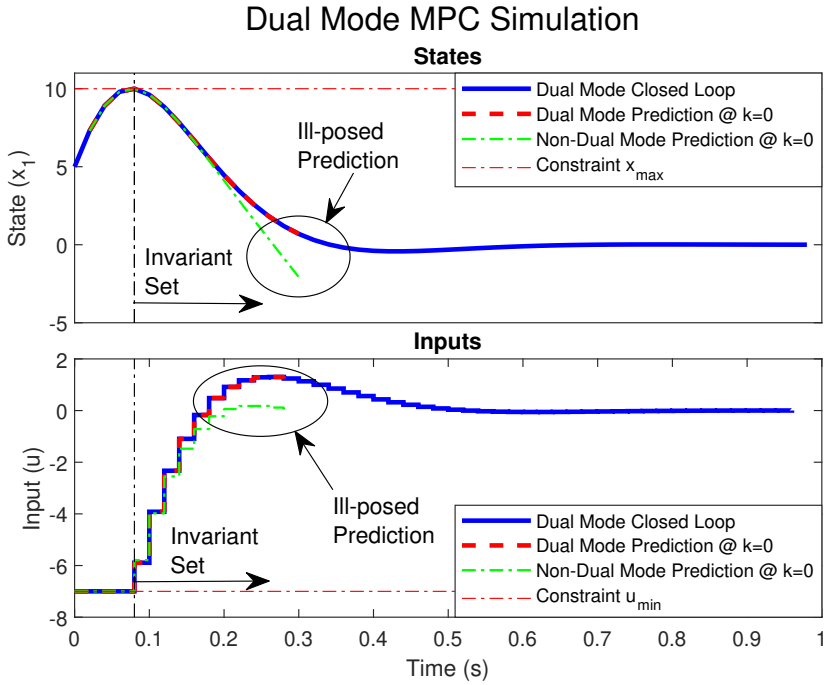


Figure 3.4: Example Predictions for Dual Mode and Non-Dual Mode MPC using Infinite Horizon Terminal Weights

Moreover consider optimisation uses the same weights $q_k = diag([1,1])$, $r_k = 5$, the same initial condition $x_0 = [5,2]^T$, with the additional state and input constraints, $-10 \leq (x_k)_1 \leq 10$ and $-7 \leq u_k \leq 10$.

Figure 7.1 shows a comparative simulation for this scenario in two cases: with and without the terminal weight. In this figure, the closed loop infinite horizon optimal response, ie. the response if $N_p \to \infty$, is depicted by the blue line; the red dashed line represents the predicted trajectory at the start of the simulation WITH the terminal weights, ie. using a Dual Mode prediction framework; and the green dot-dashed line represents the predicted trajectory WITHOUT the terminal weights, ie. a Non-Dual Mode Prediction. Note that the solution enters an invariant set at approximately $T = 0.08$ (s) where the rest of the solution within the horizon lies inside the state constraints. This allows the predicted response of the dual-mode approach (red-dashed line) at the start of the simulation to match the infinite horizon optimal response exactly. In contrast, the predicted response of the non-dual approach suffers from ill-posed prediction as signaled from the ellipsoids in the figure.

3.4.3 Convergence

In addition to the methods presented in the previous subsections which tackled the problem of stability and recursive feasibility in general MPC frameworks, a potential issue in the specific case of Nonlinear MPC is the convergence of the underlying Gauss-Newton method [28], inevitably related to the linearisation process. In the neighbourhood of the optimal solution, that is when $\delta\hat{X}^* \approx 0, \delta\hat{U}^* \approx 0$ and $\delta\hat{Y}^* \approx 0$, one can expect the Gauss-Newton method to have a linear local convergence rate [28], ie. $||\hat{U}^{[i+1]} - \hat{U}^*|| \leq \kappa||\hat{U}^{[i]} - \hat{U}^*||$, which allows the method to converge relatively fast if the devi-

ations are small. What we must understand is that the underlying linearised prediction models can result in prediction errors which affect the overall "decision making" process where the optimisation may take actions based on prediction models that are not representative of what will actually happen. We have already showed an example of this in the autonomous/auto-correcting feature example 3.1 presented in section 3.1.2. This prediction errors must not be confused with the concept of "ill-posed" or "meaningless" optimisation discussed in the zero-terminal constraint approach where the predicted responses differed significantly from the closed loop performance, something which can happen even in linear MPC. Instead, one can think of the convergence problems of SQP method even in a "stationary" manner, ie. without even moving forward in the horizon. In the case of multiple shooting, given the method enforces the linearisation around the predicted trajectory and compensates for any prediction errors using the aforementioned offsets, it embeds a certain "consistency" between the predictions from one iteration to the next which ultimately result in better convergence properties.

Likewise, a simple way of ensuring small prediction errors and consequently, convergence of the Gauss-Newton method is by enforcing the deviations $(\delta\hat{U}, \delta\hat{X}, \delta\hat{Y})$ to be sufficiently small. The standard approach of ensuring this is by using smaller steps, eg. in the form of $x^+ = x^- + \alpha\Delta x$ with $\alpha = (0, 1]$, which allows the user to take reduced step in the optimal direction. The selection of the step-length is not a trivial task, but can be done in a relatively straight forward manner using backtracking line search methods [28], where the linearised prediction is compared with the actual response, and the step-length is iteratively reduced until the difference is sufficiently low. However, given that this thesis is based on the RTI scheme which typically makes a full step [55], step-selection was not implemented as it can be a rather time consuming task [28]. Instead, in the case where it was absolutely necessary to improve convergence, a relatively simple approach was used based on imposing additional penalisation weights (Q_δ, R_δ) in the input and/or state deviations of the original cost function (3.1), resulting in an alternative cost (J_δ) in the form of:

$$J_\delta = (X_r - \hat{X})^T Q(X_r - \hat{X}) + (U_r - \hat{U})^T R(U_r - \hat{U}) + \delta\hat{X}^T Q_\delta \delta\hat{X} + \delta\hat{U}^T R_\delta \delta\hat{U} \tag{3.66}$$

This allows the optimal solution to have the same overall target, but intrinsically avoid any "big" movements that may emerge whilst allowing small movements in the deviations. Because the prediction errors have a "growing" nature where the predictions at the end are typically higher, as demonstrated in the autonomous/auto-correcting feature example 3.1, we can further modify this method by imposing exponentially decreasing weights in (Q_δ, R_δ) which would prevent big movements in the earlier stages and allow bigger movements at later stages, thus preventing big deviations on the earlier stages from propagating through the linearised dynamics.

This method was observed to produce excellent results in various systems that were used as case studies such as the Unmanned Surface Vehicle of chapter 4 and the Inverted Pendulum of chapter 5. The method was particularly required for the challenging Triple Inverted Pendulum System which is a highly nonlinear system that was used in case study 6.5 of chapter 6. It should be noted however that although the optimisation has the same overall "target" given that the deviations at convergence will be zero, including the additional penalisation term will affect how fast it reaches the optimum and therefore will affect the overall frequency response of the solution. Thus, this method must be used according to the requirements of the specific optimisation setup.

3.5 Offset Free Control

Although it may sound trivial, offset free control is an important aspect to consider in an (N)MPC optimisiation. There are 3 main methods in which offset-free control can typically be achieved, namely: disturbance estimation methods [69], and explicit [109, 114] or implicit [122, 146] integral methods. A basic requirement to achieve this is that the optimisation uses "unbiased" prediction models along with "unbiased" optimisation costs [121, 122]. In simple terms, the prediction models must account for any possible low-frequency or steady state disturbance which would allow the optimisation to understand precisely where the system is going when a given action is taken. On the other hand, an "unbiased" cost allows the optimisation to understand precisely where does it need to go in order to reach the desired reference whilst avoiding the penalisation terms of the optimisation to "fight" each other.

Example 3.4. *Disturbance Estimation Offset-Free Example*

To illustrate these concepts and provide an example of disturbance estimation methods, consider the optimisation of a second order model given by:

$$x_{k+1} = \underbrace{\begin{bmatrix} 1.8 & -0.81 \\ 1 & 0 \end{bmatrix}}_{A} x_k + \underbrace{\begin{bmatrix} 0.1 \\ 0 \end{bmatrix}}_{B} u_k \tag{3.67}$$

Let us assume that the model is augmented with an "input-disturbance" state as in [69], in the form:

$$\begin{bmatrix} x_{k+1} \\ d_{k+1} \end{bmatrix} = \begin{bmatrix} A & B \\ O & I \end{bmatrix} \begin{bmatrix} x_k \\ d_k \end{bmatrix} + \begin{bmatrix} B \\ O \end{bmatrix} u_k \tag{3.68}$$

This model allows the predictions to be unbiased in the case where there is a "known" input disturbance (d_k). Moreover, the model assumes that the disturbance is low-frequency, or ideally constant and it assumes it will be kept constant for the entire prediction horizon as seen from the identity matrix.

Now, let us consider the optimisation of this system using a prediction horizon of $N_p = 100$ with penalisation weights $q = diag([1, 1, 0])$ and $r = 10$ where the objective is to bring the state to the origin ($X_r = 0$ - regulation) from the random initial condition ($x_k = [-1.3499, 3.0349]^T$), and considering a disturbance of $d_k = 10$. To cancel this disturbance and achieve the desired regulation task, the objective function must set the input references as $U_r = -10$. In an ideal world, both of this variables can be found, eg. using disturbance estimation methods as in [69], and calculating the required input reference to achieve the desired objective, including the cancellation of the input disturbance. However, in order to illustrate the concepts of "biased" predictions, and "biased" costs, consider the 3 scenarios of figure 3.5: the first one being correct knowledge of both input disturbance, and input reference, depicted by the blue line; the second one being correct knowledge of disturbance, but incorrect input reference, thus resulting in a "biased" cost function with an "unbiased" prediction model, depicted by the red dashed line; and the third one being incorrect knowledge of disturbance, but correct input reference, resulting in a "biased" prediction model with an "unbiased" cost function, depicted by the green dot-dashed line.

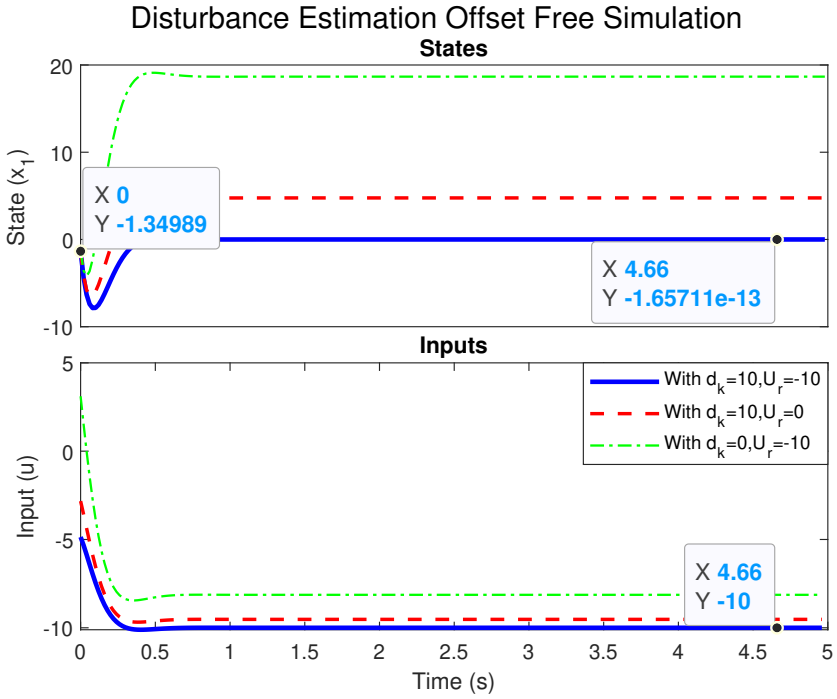


Figure 3.5: Comparison of Different Scenarios using Disturbance Estimation

From this figure, it can be appreciated that the only response reaching the origin is the blue line which has both "unbiased" prediction models and costs, thus reaching a state at the end of the simulation where the overall costs are zero, as signaled by the inner data tips. On the other hand, the optimisation in the case of the red dashed line is able to calculate precisely where the solution would go for a given set of decision variables ("unbiased prediction model") given it has the correct disturbance value. However, given that it requires an input or $\hat{U} = -10$ to reach the objective $\hat{X} = X_r = 0$, and the objective penalises any input deviation from zero, the terms of the optimisation "fight" each other ("biased cost") to achieve a consensus that minimises the overall cost. Lastly, the optimisation in the case of the green dot-dashed line has the correct overall target ("unbiased cost"), but is unable to understand how or why it should take the solution to the desired input reference given it predicts that the state would go elsewhere ("biased prediction model") with that input value which again, causes the terms of objective to "fight" each other until reaching a consensus.

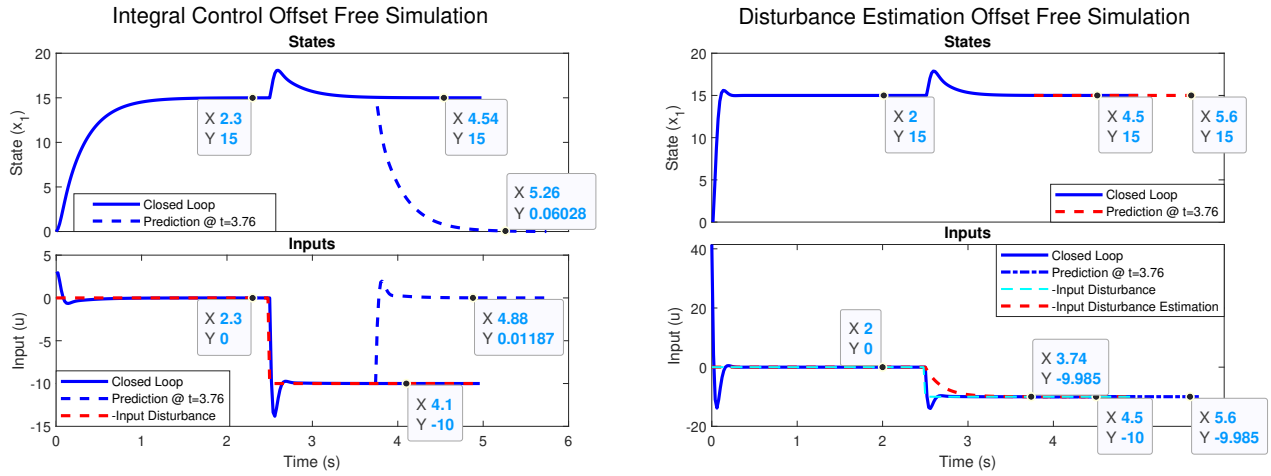
The second type of methods to be considered are integrator methods such as the one used for the NMPC of a Quadcopter in [114] where the integral errors are "explicitly" included in the predictions. Although they are very commonly used, this type of methods pose an interesting disadvantage, particularly when considered in an NMPC framework. To illustrate this, let us use a simple example.

Consider the optimisation of the double integrator model of example 3.2. In the case of integrator methods, the model must be augmented in the form:

$$\begin{bmatrix} x_{k+1} \\ e_{i_{k+1}} \end{bmatrix} = \begin{bmatrix} A & O \\ -CA & I \end{bmatrix} \begin{bmatrix} x_k \\ e_{i_k} \end{bmatrix} + \begin{bmatrix} B \\ -CB \end{bmatrix} u_k \tag{3.69}$$

where $e_{i_{k+1}} = e_{i_k} - C(Ax_k - Bu_k) = e_{i_k} - y_{k+1}$ represent the so called integral error-dynamics [109].

To optimise this system, the user must then select penalisation weights, including one for the integral state which will be used to optimally minimise the predicted integral state and achieve reference tracking. For this example, we selected penalisation weights of $q = diag([1, 1, 0.01])$ and $r = 0.1$, and used a prediction horizon of $N_p = 100$. The overall objective was to reach an output of $y_k = (x_1) = 15$ in the first state, ie. with the state-to-output matrix $C = [1, 0]$. Moreover, to provide a comparison which will allow us to illustrate the disadvantage of the method, the system was also optimised using the disturbance method of equation (3.68) with the weights $q = diag([1, 1, 0])$ and $r = 0.1$. Both scenarios were simulated for $T = 5$ (s) starting from the origin under nominal conditions (no disturbances), and an input-disturbance of $d_k = 10$ was injected at time $T = 2.5$ (s). This disturbance was estimated using the first-order estimation method described in equation (14) of [49], with a constant of $\alpha = 0.9$. Finally, the reference of the input for the disturbance estimation method was set at $U_r = -d_k$ to cancel the disturbance. The simulations can be seen in figure 3.6.



(a) Integral Control

(b) Disturbance Estimation

Figure 3.6: Difference between Steady State Predictions of Integral Control (3.6a) and Disturbance Estimation (3.6b) Offset Free Strategies

From the integral control figure 3.6a, it can be appreciated that despite the solution reaching the desired state both, with and without the presence of the disturbance, the optimisation results in a steady ill-posed prediction where the optimisation is attempting to balance the costs between the integral state (e_{i_k}), the original state (x_k) and the input u_k, as depicted by the dashed blue line. In contrast, the disturbance estimation method of figure 3.6b is able to estimate the disturbance with the first-order method as seen from the red-dashed line, and results in a well-posed optimisation where the prediction and closed loop responses are identical. Although the ill-posed effect seen in the integral control example may not be as significant in practice, it does represent a potential problem specifically when considered for NMPC given the optimisation would be linearising the system around potentially ill-posed predicted trajectories which could otherwise prove to be harmful for the optimisation. On the other hand, it is quite notorious how the disturbance estimation method achieved a significantly faster convergence to the specified state-references whilst having a similar disturbance rejection response. This is because the disturbance method does not require to "integrate" the error as in the integral method in order to achieve a desired objective. As a conceptual example, the disturbance method could

be seen as having only Proportional+Derivative actions of the well known PID controller, thus avoiding the well known 90° phase shift that comes from including integral actions. This could ultimately result in faster and better performance, and so the user must be aware of the inevitable performance degradation when considering integral methods.

Finally, the last type of optimisation are those of "implicit" integrated models. An example of these are the so called Controlled Auto-Regressive Integrated Moving Average (CARIMA) models [122], which implicitly embed an integrator into the system dynamics. Two variants for this exist, the first one being an augmented system state space with the form:

$$\begin{bmatrix} \Delta x_{k+1} \\ y_{k+1} \end{bmatrix} = \begin{bmatrix} A & 0 \\ CA & I \end{bmatrix} \begin{bmatrix} \Delta x_k \\ y_k \end{bmatrix} + \begin{bmatrix} B \\ CA \end{bmatrix} \Delta u_k \tag{3.70}$$

as presented in [146]. Note the remarkable resemblance with the integrator model of (3.69).

The second variant, which is mostly used for ARMAX (or equivalently CARMA) models, is to use the polynomial convolution where the system's output is convoluted with the Δ operator representing the embedded integrator in the system, resulting in the so called Controlled Auto-Regressive Integrated Moving Average (CARIMA). This model can then be arranged into a state space approach by using Non-Minimal State Space (NMSS) representations as discussed in chapter 9 of [146]. A general disadvantage of the CARIMA models is that they are known to be quite sensitive for predictions, and for this reason they typically require filtering methods such as T-Filters [122], or Kalman Filters [69].

At this point it may still be unclear which method should be used for a given situation. However, to quote the work done by [69], the authors concluded that "*For systems with unmeasured non zero mean disturbances, the disturbance model implementation in MPC offers the best closed loop performance"* when comparing disturbance estimation with CARIMA models. For this reason, this thesis will consider that any of the dynamical systems and problems to be presented can achieve offset-free control with any of the presented methods, with the disturbance estimation method being the preferred approach which was used for the Adaptive Laguerre-based MPC for Attitude Stabilisation of a Quadrotor publication presented in [49].

3.6 Summary

In this chapter we presented the main theory, notation, mathematical expressions, algorithms and methods related to standard NMPC which will be used throughout the thesis, provided as a "detailed" part of the Literature Review which could serve for reference or as an entry point for the new student.

The chapter included a thorough discussion on the topic of Direct Optimal Control in section 3.1.3 where the condensing-based framework for Single/Multiple Shooting "discretisation" was presented. In this section, we presented the main objective functions of interest along with the prediction models to be used, and provided the solution to the underlying Optimal Control Problems (OCPs) in two formats, namely: the Relative and Non-Relative optimisation frameworks. Both of this frameworks were proven to be equivalent in theorem 3.2, and will be used for reference in the developments of chapters 4 and 5. Moreover, section 3.2 presented a detailed explanation of Real-Time Iteration (RTI) Scheme which is considered as the "state-of-the-art" real-time NMPC solution for the rest of the thesis.

Furthermore, section 3.3 provided a set of algorithms which allow the efficient implementation of the standard NMPC. This included a re-derivation of the $O(N^2)$ and $O(N)$ algorithms from Ph.D. thesis [8], which are absolutely critical for achieving an efficient implementation and will be used for reference in chapters 5, 6 and 8 where extensions of these algorithms will be proposed. Additionally, this section included a discussion around the concepts of Auto-generation and Generic Computations which will be relevant for some of the computational comparisons performed throughout the thesis. Afterwards, section 3.4 discussed some of the nominal stability, recursive feasibility and convergence properties and methods that are typically used by NMPC which will be used for reference, particularly in chapters 5-8. The chapter ends with section 3.5 discussing 3 of the main methodologies which are commonly used to achieve offset-free control in MPC optimisation frameworks.

Chapter 4

Input Parameterised Nonlinear Model Predictive Control with Applications

In the previous chapter, we introduced the general technical background and baseline methodologies to be used in this thesis. Along the chapter we discussed topics such as multiple shooting discretisation, prediction models, optimisation methods, the Real Time Iteration (RTI) Scheme, algorithms, auto-generation, common techniques for stability, recursive feasibility guarantees, methods for improving convergence and offset-free control to provide a basis for the following chapters.

Following the contents of chapter 3 of [146]: Discrete-Time MPC Using Laguerre Functions, as well as following the works of Dr. Rossiter and co-authors along the topic of Laguerre and Kautz Polynomials eg. in [2–4, 78, 79, 125], all of which focused on their implementation in Linear MPC, these type of approaches were initially considered an interesting research direction for their implementation in the context of the RTI NMPC. Thus, in this chapter we will introduce the concept of Input Parameterised Nonlinear Model Predictive Control, along with important characteristics, advantages and disadvantages, which will provide a clear insight and motivation for their use in specific applications. It's worth mentioning that this approach has been investigated for NMPC in other works such as [33, 42], but it is often found that certain important details, alternatives and analysis that emerge from its application have been omitted in their general discussion. This provided the opportunity for the contents of this chapter which include a set of algorithms for its efficient implementation to be considered as an initial contribution of this thesis. Moreover, the contents of this chapter are considered a pre-requisite for the method of the following chapter 5, given it implements a specific type of input parameterisation.

Along the contents of this chapter, a number of conference papers were published, all of which will be used as reference and can be found in the Appendix D of this thesis. These include an initial conference paper presented in UKACC 2018 on the topic of Laguerre-based Adaptive MPC for Attitude Stabilisation of Quadrotors [49], which is related to the work presented in case study of section 4.3. Moreover, the work done with Daniel McCullough on the topic of Unmanned Surface Vehicles published in IFAC 2020 [100], provided the models and certain key characteristics for the application of the novel Laguerre/Fourier Input Parameterisation presented in case study of section 4.4. Finally, the work done with Juan Guerrero-Fernandez on the topic of Wave Energy Converters published in IFAC 2020 [59] was used in the case study from section 4.5 to illustrate a key disadvantage surrounding this general methodologies, providing the main motivation behind the approach later introduced in chapter 5.

The chapter is organised as follows: Section 4.1 introduces the general concepts, formulations and optimisation procedures for obtaining Input Parameterised NMPC solutions, along with theorem 4.1 which will provide the foundation for important discussions in chapter 5. Section 4.2 introduces a set of algorithms, along with the relevant RTI preparation and feedback phases algorithms required for the implementation of the proposed approach under the RTI Scheme, and concludes with with a generic computation comparison of the relevant algorithms where the computational advantages can be seen. Section 4.3 presents the case study of an Attitude Stabilisation NMPC controller for a Quadrotor using the so called Laguerre polynomials as an extension of the work published in [49] to illustrate the potential for significantly increased computational performance along with minimal performance losses when using the RTI Scheme. Section 4.4 presents the case study of an Unmanned Surface Vehicle (USV) using a novel Fourier/Laguerre polynomial basis function to provide the control system with tailored frequency responses that achieve the minimisation of hull impact-forces on the USV from [62, 100], as well as providing improved disturbance rejection properties when compared to the standard Fourier basis function. Finally, 4.5 presents an extension to the case study of the Wave Energy Converter (WEC) published in [59] which applies the so called Chebyshev Polynomials to illustrate important disadvantages on this type of methodology. The chapter ends with a brief summary of the key ideas and contributions to be taken from within.

4.1 General Formulations, Optimisations and Solutions

As in the previous chapter, we will begin by deriving two possible alternatives for the solution of Input Parameterised optimisations, namely; Relative and Non-Relative based optimisations.

The main underlying idea behind this methodology is based on considering a general input parameterisation (or input structure) of the form (4.1a) for non-relative optimisation, and (4.1b) for relative optimisation, given by:

$$\hat{U} = \mathbb{N}\hat{\mathbb{U}} \qquad \rightarrow \qquad \delta\hat{U} = \mathbb{N}\hat{\mathbb{U}} - \bar{U} \tag{4.1a}$$

$$\delta\hat{U} = \mathbb{N}\delta\hat{\mathbb{U}} \qquad \rightarrow \qquad \hat{U} = \bar{U} + \mathbb{N}\delta\hat{\mathbb{U}} \tag{4.1b}$$

where $\mathbb{N} \in \mathbb{R}^{N_p n_u \times n_{\mathbb{N}} n_u}$ is an input-parameterisation matrix defining the relationship between the original input/decision vectors ($\hat{U} \in \mathbb{R}^{N_p n_u}, \delta\hat{U} \in \mathbb{R}^{N_p n_u}$), and the "compressed" input/decision vectors ($\hat{\mathbb{U}} \in \mathbb{R}^{n_{\mathbb{N}} n_u}, \delta\hat{\mathbb{U}} \in \mathbb{R}^{n_{\mathbb{N}} n_u}$), where $n_{\mathbb{N}}$ are the number of degrees-of-freedom that each input trajectory will have, which for obvious reasons, will be considered less than the original, ie. $n_{\mathbb{N}} n_u < N_p n_u$.

The idea behind these methodologies is that in some cases, we could "replicate" the original decision variables by using fewer degrees-of-freedoms with certain "patterns" or trajectories embedded in them. To achieve this, there exist a wide variety of input structures such as Laguerre Polynomials [146], Kautz Polynomials [79], Fourier Polynomials [10], Chebyshev Polynomials [148] and so on, some of which will be used as examples in the case studies of sections 4.3, 4.4 and 4.5. Along these input structures it belongs the common approach of Generalised Predictive Control (GPC) [122] where the inputs have a so called "control horizon" different than the prediction horizon, typically by leaving the last decision variable constant, ie. ($u_{k+N_c-1} = u_{k+N_c} = \cdots = u_{k+N_p-1}$), although this variant is not typically recognised as such. We will discuss this variant later in chapter 5.

Let us now consider the cost functions that would result from using the input structures given above. By substituting the appropriate input structures (4.1a or 4.1b) in the linearised state prediction model (3.17a) and the linearised input prediction model (3.17b), and substituting this new input-parameterised prediction models into the original cost function (3.1), the resulting input-parameterised cost functions including their respective inequality constraints would be given by:

Non-Relative Input Parameterised Cost Function

$$J = \frac{1}{2}(X_r - \bar{X} - D - G\delta x_0 - H\mathbb{N}\hat{\mathbb{U}} + H\bar{U})^T Q(X_r - \bar{X} - D - G\delta x_0 - H\mathbb{N}\hat{\mathbb{U}} + H\bar{U}) + \frac{1}{2}(\mathbb{N}\hat{\mathbb{U}} - U_r)^T R(\mathbb{N}\hat{\mathbb{U}} - U_r) \tag{4.2a}$$

$$U_{min} \leq \mathbb{N}\hat{\mathbb{U}} \leq U_{max} \tag{4.2b}$$

$$X_{min} \leq \bar{X} + D + G\delta x_0 + H\mathbb{N}\hat{\mathbb{U}} - H\bar{U} \leq X_{max} \tag{4.2c}$$

Relative Input Parameterised Cost Function

$$J = \frac{1}{2}(X_r - \bar{X} - D - G\delta x_0 - H\mathbb{N}\delta\hat{\mathbb{U}})^T Q(X_r - \bar{X} - D - G\delta x_0 - H\mathbb{N}\delta\hat{\mathbb{U}}) + \frac{1}{2}(\bar{U} + \mathbb{N}\delta\hat{\mathbb{U}} - U_r)^T R(\bar{U} + \mathbb{N}\delta\hat{\mathbb{U}} - U_r) \tag{4.3a}$$

$$U_{min} \leq \bar{U} + \mathbb{N}\delta\hat{\mathbb{U}} \leq U_{max} \tag{4.3b}$$

$$X_{min} \leq \bar{X} + D + G\delta x_0 + H\mathbb{N}\delta\hat{\mathbb{U}} \leq X_{max} \tag{4.3c}$$

Following the standard procedure of expressing the costs and inequalities in terms of the decision variables, and disregarding for constant terms, results in the following Input-Parameterised Quadratic Programs (QP):

Non-Relative Input Parameterised Quadratic Program

$$J = \frac{1}{2}\hat{\mathbb{U}}^T E_{\mathbb{N}}\hat{\mathbb{U}} + \hat{\mathbb{U}}^T f_{\mathbb{N}} \tag{4.4a}$$

$$M_{\mathbb{N}}\hat{\mathbb{U}} \leq \gamma \tag{4.4b}$$

$$E_{\mathbb{N}} = \mathbb{N}^T(H^T Q H + R)\mathbb{N} \tag{4.4c}$$

$$f_{\mathbb{N}} = -\mathbb{N}^T\left[H^T Q(X_r - \bar{X} - D - G\delta x_0 + H\bar{U}) + RU_r\right] \tag{4.4d}$$

$$M_{\mathbb{N}} = \begin{bmatrix} \mathbb{N} \\ -\mathbb{N} \\ H\mathbb{N} \\ -H\mathbb{N} \end{bmatrix} \qquad \gamma = \begin{bmatrix} U_{max} \\ -U_{min} \\ X_{max} - \bar{X} - D - G\delta x_0 + H\bar{U} \\ -(X_{min} - \bar{X} - D - G\delta x_0 + H\bar{U}) \end{bmatrix} \tag{4.4e}$$

Relative Input Parameterised Quadratic Program

$$J = \frac{1}{2}\delta\hat{\mathbb{U}}^T E_{\mathbb{N}}\delta\hat{\mathbb{U}} + \delta\hat{\mathbb{U}}^T f_{\mathbb{N}} \tag{4.5a}$$

$$M_{\mathbb{N}}\delta\hat{\mathbb{U}} \leq \gamma \tag{4.5b}$$

$$E_{\mathbb{N}} = \mathbb{N}^T(H^T Q H + R)\mathbb{N} \tag{4.5c}$$

$$f_{\mathbb{N}} = -\mathbb{N}^T\left[H^T Q(X_r - \bar{X} - D - G\delta x_0) - R(\bar{U} - U_r)\right] \tag{4.5d}$$

$$M_{\mathbb{N}} = \begin{bmatrix} \mathbb{N} \\ -\mathbb{N} \\ H\mathbb{N} \\ -H\mathbb{N} \end{bmatrix} \qquad \gamma = \begin{bmatrix} U_{max} - \bar{U} \\ -(U_{min} - \bar{U}) \\ X_{max} - \bar{X} - D - G\delta x_0 \\ -(X_{min} - \bar{X} - D - G\delta x_0) \end{bmatrix} \tag{4.5e}$$

Remark 4.1. *Constraint Selection*

As in the previous chapter, the inequality constraints can be reduced to include only the relevant inputs or states by selecting (or computing) only the respective rows of constraint matrix $M_{\mathbb{N}}$, and constraint vector γ.

Notice that in both resulting QPs (4.4 and 4.5), the expressions of $E_{\mathbb{N}}$, $f_{\mathbb{N}}$ and $M_{\mathbb{N}}$ are essentially "compressions" of the original QPs (3.31 and 3.33) Hessian (E), linear-term (f) and constraint matrix (M) presented in the previous chapter 3, given by:

$$E_{\mathbb{N}} = \mathbb{N}^T E\mathbb{N} \tag{4.6a}$$

$$f_{\mathbb{N}} = \mathbb{N}^T f \tag{4.6b}$$

$$M_{\mathbb{N}} = M\mathbb{N} \tag{4.6c}$$

Indeed, an alternative way of obtaining these QPs is to directly substitute input structures (4.1a and 4.1b) in Quadratic Programs (3.31 and 3.33), respectively, essentially compressing the previously derived optimisations. However, this must not overshadow the underlying mathematical procedure and concepts that are behind this input parameterisation where the actual reason for this compression/decompression equivalency is the modification of the underlying prediction models. Also, by avoiding the computation (and then compression) of E, f, M, one can go directly to the resulting expressions, potentially avoiding computations and memory allocation. This will be discussed further in section 4.2 where a method for directly calculating this compressed matrices will be presented.

Having established the QPs to be solved, any standard general purpose solver can be used such as QP OASES or quadprog. Once the solution is found, the original decision vector can be recovered by using the original input structure expressions (4.1a or 4.1b). Additionally, in the case of using the multiple shooting approach, the expansion step must be performed to obtain the nominal trajectories for the next sampling time, as explained in chapter 3.

Remark 4.2. *Condition: Positive-Definite Compressed Hessian and Properties of* $\mathbb{N}$

In order to obtain a convex problem for the compressed linearised QP, the compressed Hessian $E_{\mathbb{N}} = \mathbb{N}^T E\mathbb{N}$ must be a positive-definite matrix which can be satisfied simply by ensuring $\mathbb{N}$ is full-column rank. Other properties of interest for $\mathbb{N}$ are orthogonality and recursive feasibility.

Remark 4.3. *Condition: Independent Input Parameterisations*

For the purpose of this chapter, we will only consider input parameterisations which result in completely independent compressed decision vectors for each input ($1 \rightarrow n_u$), ie. each input having its own input-structure. An example of non-independent input parameterisation is the control-allocation as discussed in the literature review, which is not the interest of this chapter.

4.1.1 Real-Time Iteration

To implement this approach using the Real Time Iteration (RTI) Scheme the user can simply follow the guidelines established in chapter 3, section 3.2 and apply the standard steps, ie. performing the Initial Value Embedding (IVE) shifting approach, performing a single full-step SQP iteration and separating the computations in Preparation and Feedback Phases by using the predicted state obtained from the previous state and input ($\bar{x}_0 = \hat{x}_{k|k-1} = f(x_{k-1|k-1}, u_{k-1|k-1})$), calculating the initial offset ($\delta x_0 = x_0 - \bar{x}_0$) as soon as the measurement arrives and solving the QP.

On the other hand, if the user requires to achieve strict deterministic timings, one option is to implement the alternative approach discussed in chapter 3 to completely remove the time-delay related to the QP solution in the feedback phase by implementing equations (3.49 or 3.50) which as discussed, is based on solving the problem with the assumption of $\delta x_0 = 0$, and then re-including the feedback term to provide an optimal "unconstrained correction" to the "approximated constrained solution".

Non-Relative Input-Parameterised Solution

$$\hat{U} = -\mathbb{N}E_{\mathbb{N}}^{-1}\mathbb{N}^T \left[\underbrace{\overbrace{-H^TQ(X_r - \bar{X} + H\bar{U}) + RU_r}^{\text{Unconstrained}} \overbrace{+M^T\bar{\lambda}}^{\text{Constrained}}}_{\text{Preparation Phase}} \underbrace{+H^TQG\delta x_k}_{\text{Feedback Phase}} \right] \tag{4.7}$$

Relative Input-Parameterised Solution

$$\hat{U} = \bar{U} - \mathbb{N}E_{\mathbb{N}}^{-1}\mathbb{N}^T \left[\underbrace{\overbrace{-(H^TQ(X_r - \bar{X}) - R(\bar{U} - U_r))}^{\text{Unconstrained}} \overbrace{+M^T\bar{\lambda}}^{\text{Constrained}}}_{\text{Preparation Phase}} \underbrace{+H^TQG\delta x_k}_{\text{Feedback Phase}} \right] \tag{4.8}$$

Theorem 4.1. *The Equality Condition*

In contrast to theorem 3.2 where the solution for Non-Relative and Relative optimisations will always be exactly the same, this equality can only be satisfied for input-parameterised solutions if the nominal input $\bar{U}$ can be exactly replicated with the input structure $\mathbb{N}$.

Proof. If $\bar{U}$ can be exactly replicated with the input structure $\mathbb{N}$, then there exist a compressed nominal input vector $\bar{\mathbb{U}}$ that is capable of satisfying the equality (4.9). We will refer to this as the "equality condition".

$$\bar{U} = \mathbb{N}\bar{\mathbb{U}} \tag{4.9}$$

Now, following a similar procedure as in theorem 3.2, we can begin by equating both unconstrained solutions resulting in:

$$\begin{aligned}\bar{U} + \mathbb{N}E_{\mathbb{N}}^{-1}\left(\mathbb{N}^T\left[H^TQ(X_r - \bar{X} - D - G\delta x_0) - R(\bar{U} - U_r)\right]\right) \\ = \mathbb{N}E_{\mathbb{N}}^{-1}\left(\mathbb{N}^T\left[H^TQ(X_r - \bar{X} - D - G\delta x_0 + H\bar{U}) + RU_r\right]\right)\end{aligned} \tag{4.10}$$

From this, we can immediately see that the equality reduces to:

$$\begin{aligned}\bar{U} &= \mathbb{N}E_{\mathbb{N}}^{-1}\mathbb{N}^T\underbrace{\left(H^TQH + R\right)}_{E}\bar{U} \\ &= \mathbb{N}(\mathbb{N}^TE\mathbb{N})^{-1}\mathbb{N}^TE\bar{U}\end{aligned} \tag{4.11}$$

Substituting equality (4.9) in the right hand $\bar{U}$ of (4.11), results in:

$$\bar{U} = \mathbb{N}\cancel{(\mathbb{N}^TE\mathbb{N})^{-1}(\mathbb{N}^TE\mathbb{N})}\bar{\mathbb{U}} = \boxed{\mathbb{N}\bar{\mathbb{U}} = \bar{U}}$$

Thus, satisfying the equality (4.10) with the use of equality (4.9), as signaled by the red box. □

A similar proof can be obtained for constrained solutions, though it involves a much longer procedure, similar to the one presented for theorem 3.2. Regardless, this equality was consistently observed during experiments with constrained solutions, and will be discussed further in chapter 5.

The essential part of what theorem 4.1 tell us is that if the Relative-based input-structure can replicate the Non-Relative-based solution, it will. Moreover, it provides insight into an important characteristic to keep in mind which is that, conceptually, it is not the same to embed a desired input-structure into relative decision variables than it is to do so in the non-relative alternative. Thus, one must take this into account when designing the input structure to ensure the desired outcome, which could otherwise result in an entirely different target (optimisation). On a side note, the most common way of performing input parameterisation is based on the Relative input structure format as this is naturally considered a Newton's method-like framework where the optimisation calculates a desired deviation to the nominal/linearisation point/value, eg. $(x^+ = x^- + \Delta x)$. Instead, the Non-Relative framework provides the final result directly.

4.2 Algorithm Details and Autogeneration

Having defined the relevant theory and methods required by the Input-Parameterisation method, we can now proceed to introduce a set of algorithms that allow its efficient implementation using the RTI NMPC framework. Thus, this section will present a set of methods and algorithms that were used for developing an auto-generation toolkit that allowed us to benchmark the proposed approach against the solution obtained by the ACADO toolkit. We will first introduce some important details related to handling the input-parameterisation matrix ($\mathbb{N}$) itself in sections 4.2.1, 4.2.2 and 4.2.3, then provide a set of "core" algorithms in section 4.2.4, which will be used by the final RTI preparation and feedback algorithms provided in section 4.2.5. The section concludes with a generic computation comparison on some of the relevant "core" algorithms provided in section 4.2.6.

4.2.1 Input Structure Order

One of the key steps required to achieve computational efficiency for applying this type of general input-parameterisation is that of having appropriate input structures for multi-input systems, as well as appropriate array ordering in general. This can be better explained with a simple example.

Example 4.1. *The Two-Input Structure Alternatives*

Consider the following two input structures alternatives (4.12a and 4.12b) for representing the 3 initial steps $(k, k+1, k+2)$ of a 2-input $(\hat{u}^1, \hat{u}^2)$ system given by:

$$\begin{bmatrix} \hat{u}^1_k \\ \hat{u}^2_k \\ \hat{u}^1_{k+1} \\ \hat{u}^2_{k+1} \\ \hat{u}^1_{k+2} \\ \hat{u}^2_{k+2} \end{bmatrix} = \begin{bmatrix} (a_1)_0 & (a_2)_0 & 0 & 0 \\ 0 & 0 & (b_1)_0 & (b_2)_0 \\ (a_1)_1 & (a_2)_1 & 0 & 0 \\ 0 & 0 & (b_1)_1 & (b_2)_1 \\ (a_1)_2 & (a_2)_2 & 0 & 0 \\ 0 & 0 & (b_1)_2 & (b_2)_2 \end{bmatrix} \begin{bmatrix} \hat{\mathbb{U}}^1_0 \\ \hat{\mathbb{U}}^1_1 \\ \hat{\mathbb{U}}^2_0 \\ \hat{\mathbb{U}}^2_1 \end{bmatrix} \tag{4.12a}$$

$$\begin{bmatrix} \hat{u}^1_k \\ \hat{u}^2_k \\ \hat{u}^1_{k+1} \\ \hat{u}^2_{k+1} \\ \hat{u}^1_{k+2} \\ \hat{u}^2_{k+2} \end{bmatrix} = \begin{bmatrix} (a_1)_0 & 0 & (a_2)_0 & 0 \\ 0 & (b_1)_0 & 0 & (b_2)_0 \\ (a_1)_1 & 0 & (a_2)_1 & 0 \\ 0 & (b_1)_1 & 0 & (b_2)_1 \\ (a_1)_2 & 0 & (a_2)_2 & 0 \\ 0 & (b_1)_2 & 0 & (b_2)_2 \end{bmatrix} \begin{bmatrix} \hat{\mathbb{U}}^1_0 \\ \hat{\mathbb{U}}^2_0 \\ \hat{\mathbb{U}}^1_1 \\ \hat{\mathbb{U}}^2_1 \end{bmatrix} \tag{4.12b}$$

where $(a_1, a_2, b_1, b_2)_k$ are the coefficients of the parameterisation matrix and $(\mathbb{U}_0, \mathbb{U}_1)^k$ are the two decision variables available for each of the k_{th} inputs.

Clearly both input structures have exactly the same decision variables available, with the only difference being the way the input structure is ordered in the parameterisation matrices ($\mathbb{N}$). This inevitable changes the way the compressed decision vectors ($\mathbb{U}$) need to be arranged, and the way they will be related to the other terms of the optimisation such as the compressed linear term ($f_{\mathbb{N}}$). As an example, the reader can verify that just by choosing an arrangement in the form of equation (4.12a), the first two elements of ($f_{\mathbb{N}}$) will be related to the first order derivatives of the cost function w.r.t the first compressed input ($\hat{\mathbb{U}}^1$), and the last two elements to the first order derivatives w.r.t. the second compressed input ($\hat{\mathbb{U}}^2$). What is important to understand from this is that the order of the input structure will affect the organisation of all the terms of the optimisation.

One may intuitively want to arrange the structure in the form of equation (4.12b), ie. first coefficient of both inputs, then second coefficient, and so on. However, from an algorithmic perspective it may be easier to have the input structure arranged as provided in equation (4.12a) given that in order to access the coefficients related to a given input, one may access a simple row in a given range to avoid the zero-terms computations. Note that from a computer memory arrangement perspective, one can access the complete set of coefficients in a given row simply by giving a pointer to the initial value, provided the matrix is stored using "row-major" order. Otherwise, the coefficients determining this relationship may be scattered around that row as it is the case of equation (4.12b) which may present additional complications for an algorithm to handle. We will discuss this further in the following subsection.

4.2.2 Direct Computation of Compressed State Prediction Matrix $H_{\mathbb{N}}$

By now it may still not be evident that one of the main and most important differences with the standard framework of chapter 3 is that of the compressed state prediction matrix ($H_{\mathbb{N}} = H\mathbb{N}$). As an example, notice the operation of $\mathbb{N}^T H^T Q H \mathbb{N}$ required by the compressed Hessian ($E_{\mathbb{N}}$) can be obtained simply by $H_{\mathbb{N}}^T Q H_{\mathbb{N}}$. Similarly, the operation $\mathbb{N}^T H^T$ required by the linear term ($f_{\mathbb{N}}$) would be simply $H_{\mathbb{N}}^T$. This matrix $H_{\mathbb{N}}$ also forms part of the compressed constraint matrix ($M_{\mathbb{N}}$). Finally, its direct calculation would avoid the additional memory required to store H matrix. Thus, the efficient computation of this matrix ($H_{\mathbb{N}}$) is of extreme importance and will be the focus of this subsection.

Let us begin by expanding the first few terms of this computations to illustrate the underlying pattern in question. Using the arrangement preference discussed in the previous subsection (ie. that of equation 4.12a), we have the following operation:

$$H_{\mathbb{N}} = \underbrace{\left[\begin{array}{ccc|ccc|ccc} B_0^1 & \cdots & B_0^{n_u} & 0 & \cdots & 0 & 0 & \cdots & 0 \\ A_1B_0^1 & \cdots & A_1B_0^{n_u} & B_1^1 & \cdots & B_1^{n_u} & 0 & \cdots & 0 \\ A_2A_1B_0^1 & \cdots & A_2A_1B_0^{n_u} & A_2B_1^1 & \cdots & A_2B_1^{n_u} & B_2^1 & \cdots & B_2^{n_u} \end{array}\right]}_{H} \underbrace{\left[\begin{array}{ccc} \mathbb{N}_0^1 & 0 & \cdots \\ \vdots & \ddots & \vdots \\ \cdots & 0 & \mathbb{N}_0^{n_u} \\ \hline \mathbb{N}_1^1 & 0 & \cdots \\ \vdots & \ddots & \vdots \\ \cdots & 0 & \mathbb{N}_1^{n_u} \\ \hline \mathbb{N}_2^1 & 0 & \cdots \\ \vdots & \ddots & \vdots \\ \cdots & 0 & \mathbb{N}_2^{n_u} \end{array}\right]}_{\mathbb{N}} \tag{4.13}$$

where B_k^i represents the i_{th} column of the matrix B_k at time step k, ie. related to the i_{th} input.

The resulting matrix can be rewritten in the following form

$$H_{\mathbb{N}} = \left[\begin{array}{c|c|c} B_0^1\mathbb{N}_0^1 & \cdots & B_0^{n_u}\mathbb{N}_0^{n_u} \\ A_1B_0^1\mathbb{N}_0^1 + B_1^1\mathbb{N}_1^1 & \cdots & A_1B_0^{n_u}\mathbb{N}_0^{n_u} + B_1^{n_u}\mathbb{N}_1^{n_u} \\ A_2A_1B_0^1\mathbb{N}_0^1 + A_2B_1^1\mathbb{N}_1^1 + B_2^1\mathbb{N}_2^1 & \cdots & A_2A_1B_0^{n_u}\mathbb{N}_0^{n_u} + A_2B_1^{n_u}\mathbb{N}_1^{n_u} + B_2^{n_u}\mathbb{N}_2^{n_u} \end{array}\right] \tag{4.14}$$

Thus it can be clearly seen how this whole operation can be separated into independent operations for the parameterisation of each input. In particular, what is most important is the efficient computation of compressed input-to-state matrix ($B_k\mathbb{N}_k$), which can be achieved by:

$$B_k\mathbb{N}_k = \left[\begin{array}{c|c|c} B_k^1\mathbb{N}_k^1 & \cdots & B_k^{n_u}\mathbb{N}_k^{n_u} \end{array}\right] = (B_{\mathbb{N}})_k \tag{4.15}$$

Having defined this, matrix $H_{\mathbb{N}}$ reduces to:

$$H_{\mathbb{N}} = \left[\begin{array}{c} B_0\mathbb{N}_0 \\ A_1(B_0\mathbb{N}_0) + B_1\mathbb{N}_1 \\ A_2(A_1B_0\mathbb{N}_0 + B_1\mathbb{N}_1) + B_2\mathbb{N}_2 \end{array}\right] \tag{4.16}$$

Or simply:

$$H_{\mathbb{N}} = \begin{bmatrix} (h_{\mathbb{N}})_1 \\ (h_{\mathbb{N}})_2 \\ \vdots \\ (h_{\mathbb{N}})_{N_p} \end{bmatrix} \tag{4.17a}$$

$$(h_{\mathbb{N}})_k = \begin{cases} B_{k-1}\mathbb{N}_{k-1} & k = 1 \\ A_{k-1}(h_{\mathbb{N}})_{k-1} + B_{k-1}\mathbb{N}_{k-1} & k > 1 \end{cases} \tag{4.17b}$$

Remark 4.4. *Fully Dense Hessian and Linear Term*

*Note that because ($H_{\mathbb{N}}$) does not have any zero terms, and assuming the term ($QH_{\mathbb{N}}$) can be computed efficiently using the special diagonal-matrix operation (mat3 = scalar * diag1 * mat1;) available in Eigen 3 library [15], the computations of the Hessian term ($H_{\mathbb{N}}^T Q H_{\mathbb{N}}$) and linear term ($H_{\mathbb{N}}^T Q X_e$) automatically become fully dense (no zeros), thus simplifying their computation and not requiring any special handling, eg. as in the $O(N)/O(N^2)$ algorithms 3.3 and 3.2 of section 3.3.1. Although symmetry could be exploited for the Hessian term by a special algorithm, it was considered unnecessary given the already large increase in computational performance as seen in table 4.1.*

Looking to use this method, the total parameterisation matrix $\mathbb{N}$ can be organised more efficiently by storing each input-parameterisation $\mathbb{N}^i \ \forall i = [1, n_u]$, separately. In our specific case, we decided to arrange each input-structure vertically in our programs as:

$$\mathbb{N} = \begin{bmatrix} \mathbb{N}^1 \\ \vdots \\ \mathbb{N}^{n_u} \end{bmatrix} \tag{4.18}$$

using "row-major" order which helps to provide a straight forward access to the rows of each input.

The algorithm will then compute $B_k\mathbb{N}_k$ based on the form of equation (4.15), and then execute the recursive formula (4.17b). This can be seen explicitly in lines 4,7 and 9 of the core algorithm 4.1.

Remark 4.5. *Only For Storage*

Note that matrix (4.18) MUST NOT to be used explicitly for multiplication (eg. $H\mathbb{N}$ or $\hat{U} = \mathbb{N}\hat{\mathbb{U}}$) given it not correctly ordered, and it is only suggested to be used as part of the provided algorithm to be accessed by the program whilst using minimum memory storage (ie. avoiding zeros storage).

Once the solution ($\hat{\mathbb{U}}$ or $\delta\hat{\mathbb{U}}$) is found, the recovery is done programatically by re-accessing this same input structure and distributing the solution according to the original arrangement. Note that because of the way the input-parameterisation is arranged, this can be done simply by:

$$\hat{U}^i = \mathbb{N}^i\hat{\mathbb{U}}^i \qquad \forall i = [1, n_u] \tag{4.19a}$$

$$\hat{U}^i = \bar{U}^i + \mathbb{N}^i\delta\hat{\mathbb{U}}^i \qquad \forall i = [1, n_u] \tag{4.19b}$$

This can be seen explicitly in the line 3 of the core algorithm 4.4 which performs the decompression by assigning this result to a dummy variable ($\tilde{W}$) and then re-distributing it on lines 4-6.

4.2.3 The $\mathbb{N}$ Input-Related Terms

Lastly, there are three other important operations related to the input penalisation weights and constraints which are present in all the QP variables, ie. the compressed Hessian ($E_{\mathbb{N}}$), eg in the term $\mathbb{N}^T R\mathbb{N}$; the linear term ($f_{\mathbb{N}}$), eg. in the term $\mathbb{N}^T R(\bar{U} - U_r)$; and finally constraint matrix and vector ($M_{\mathbb{N}}$ and γ), in the way that input-constraints must be embedded now that the input matrix has been "reordered", ie. given $\mathbb{N}$ from equation (4.18) can no longer be used explicitly as stated in remark 4.5.

On the one hand, the term $\mathbb{N}^T R\mathbb{N}$ can be pre-computed and stored, provided $\mathbb{N}$ and R are time-invariant, although if this is not possible, the computation can be shown to simply be given by:

$$\mathbb{N}^T R\mathbb{N} = \begin{bmatrix} (\mathbb{N}^1)^T R^1 \mathbb{N}^1 & \mathbb{0} & \cdots \\ \vdots & \ddots & \vdots \\ \cdots & \mathbb{0} & (\mathbb{N}^{n_u})^T R^{n_u} \mathbb{N}^{n_u} \end{bmatrix} \tag{4.20}$$

This operation can be seen explicitly in line 3 of the core algorithm 4.2.

On the other hand, a simple analysis of the term $\mathbb{N}^T R(\bar{U} - U_r)$ can be proved to result in:

$$\mathbb{N}^T R(\bar{U} - U_r) = \begin{bmatrix} (\mathbb{N}_0^1)^T r_0^1 (\bar{u}_0^1 - u_{r_0}^1) + \cdots + (\mathbb{N}_{N_p-1}^1)^T r_{N_p-1}^1 (\bar{u}_{N_p-1}^1 - u_{r_{N_p-1}}^1) \\ \vdots \\ (\mathbb{N}_0^{n_u})^T r_0^{n_u} (\bar{u}_0^{n_u} - u_{r_0}^{n_u}) + \cdots + (\mathbb{N}_{N_p-1}^{n_u})^T r_{N_p-1}^{n_u} (\bar{u}_{N_p-1}^{n_u} - u_{r_{N_p-1}}^{n_u}) \end{bmatrix} \tag{4.21}$$

which can be done through a simple recursive operation. Indeed, this last operation can be seen explicitly in line 4 of the core algorithm 4.3 where the linear-term ($f_{\mathbb{N}}$) is iteratively modified. A very small modification would be required for the case of Non-Relative optimisation.

Finally, the last operation relates to the organisation of the matrix $M_{\mathbb{N}}$, and constraint vector γ, particularly that of the input constraints given matrix $\mathbb{N}$ cannot be used explicitly as stated from remark 4.5. One can fix this with a simple re-organisational fix by using a "reorganised" matrix $(\mathbb{N})_{reorg}$, and "reorganised" input constraints $(U_{min} \leq \hat{U} \leq U_{max})_{reorg}$ in the resulting QP. As an example, applying this to the following for the Non-Relative QP would result:

$$(M_{\mathbb{N}})_{reorg} = \begin{bmatrix} \mathbb{N}_{reorg} \\ -\mathbb{N}_{reorg} \\ H_{\mathbb{N}} \\ -H_{\mathbb{N}} \end{bmatrix} \quad (\gamma)_{reorg} = \begin{bmatrix} (U_{max})_{reorg} \\ -(U_{min})_{reorg} \\ X_{max} - \bar{X} - D - G\delta x_0 - H\bar{U} \\ \bar{X} + D + G\delta x_0 + H\bar{U} - X_{min} \end{bmatrix} \tag{4.22}$$

with $\mathbb{N}_{reorg}$, $(U_{max})_{reorg}$ and $(U_{min})_{reorg}$ defined as:

$$\mathbb{N}_{reorg} = \begin{bmatrix} \mathbb{N}^1 & \cdots & \mathbb{0} \\ \vdots & \ddots & \vdots \\ \mathbb{0} & \cdots & \mathbb{N}^{n_u} \end{bmatrix} \quad (U_{max})_{reorg} = \begin{bmatrix} U_{max}^1 \\ \vdots \\ U_{max}^{n_u} \end{bmatrix} \quad (U_{min})_{reorg} = \begin{bmatrix} U_{min}^1 \\ \vdots \\ U_{min}^{n_u} \end{bmatrix} \tag{4.23}$$

This last operation is only provided to generate awareness that a reorganisation is required, and a plausible and simple way of achieving it with the expressions above.

4.2.4 Core Algorithms

Having defined the relevant details that support the core algorithms, we can now proceed to introduce them. The approach uses 4 "core" algorithms for its efficient implementation, namely (given in order of introduction and usage): the compressed state prediction matrix $H_{\mathbb{N}}$ computation; the compressed Hessian $E_{\mathbb{N}}$ computation; the compressed linear term $f_{\mathbb{N}}$ computation; and the decompression routine for obtaining the original uncompressed variable $\delta\hat{U}^*$. These algorithms will then be used by the final RTI preparation and feedback algorithms introduced in the following subsection 4.2.5, and will be tested in the generic computations comparison of section 4.2.6. All the algorithms assume that the input-parameterised matrix $\mathbb{N}$ is handled via the "separated" vertical arrangement of equation (4.18).

Algorithm 4.1 calculates the compressed state prediction matrix ($H_{\mathbb{N}}$) based on the method described in section 4.2.2.

Algorithm 4.1: Compressed State Prediction Matrix $H_{\mathbb{N}}$ Calculation

Data: A_k, B_k, N_p, n_u
begin
  for $k = 1$ **to** N_p **do**
    for $i = 1$ **to** n_u **do**
      $(B_{\mathbb{N}})^i_{k-1} = B^i_{k-1}\mathbb{N}^i_{k-1}$; // Compute Compressed Input-to-State Matrix
    end
    if $k = 1$ **then**
      $(h_{\mathbb{N}})_k = (B_{\mathbb{N}})_{k-1}$; // Initial Value
    else
      $(h_{\mathbb{N}})_k = A_{k-1}(h_{\mathbb{N}})_{k-1} + (B_{\mathbb{N}})_{k-1}$; // Propagate Recursively
    end
  end
end
Result: $H_{\mathbb{N}}$

On the other hand, algorithm 4.2 calculates the compressed Hessian ($E_{\mathbb{N}}$) via the method described in section 4.2.3, noting that the term $H_{\mathbb{N}}^T Q H_{\mathbb{N}}$ is fully dense and can be efficiently calculated using the special diagonal matrix multiplication in the Eigen 3 library as discussed in remark 4.4. In this algorithm, the notation $(E_{\mathbb{N}})_{(i-1)n_{\mathbb{N}}+1 \to in_{\mathbb{N}},(i-1)n_{\mathbb{N}}+1 \to in_{\mathbb{N}}}$ indicates an operation in the compressed Hessian ($E_{\mathbb{N}}$) in the rows and columns over the range $(i-1)n_{\mathbb{N}} + 1 \to in_{\mathbb{N}}$. Finally, we assume that the column-vector R^i, represents the i_{th} input penalisation terms, which can be either pre-stored or simply assigned to a extra dummy variable which would require minimum storage.

Afterwards, algorithm 4.3 performs the calculation of the compressed linear term ($f_{\mathbb{N}}$) based on the method established in section 4.2.3, noting that the term $H_{\mathbb{N}} Q X_e$ can also be calculated, as in the previous algorithm, using the special diagonal matrix multiplication in the Eigen 3 library. Moreover, similar to the previous algorithm, the notation $(f_{\mathbb{N}})_{(i-1)n_{\mathbb{N}}+1 \to in_{\mathbb{N}}}$ indicates an operation in the compressed linear term ($f_{\mathbb{N}}$) in the rows over the range $(i-1)n_{\mathbb{N}} + 1 \to in_{\mathbb{N}}$.

Finally, algorithm 4.4 performs the decompression of the optimal compressed decision vector $\delta\hat{\mathbb{U}}^*$ by storing the i_{th} input parameterisation variables in a dummy variable ($\tilde{W}$), and distributing them.

Algorithm 4.2: Compressed Hessian $E_{\mathbb{N}}$ Calculation

Data: $H_{\mathbb{N}}, Q, R, \mathbb{N}, n_u, n_{\mathbb{N}}$

begin

  $E_{\mathbb{N}} = H_{\mathbb{N}}^T Q H_{\mathbb{N}}$; // Initial Value

  for $i = 1$ **to** n_u **do**

    /* Modify blocks of $E_{\mathbb{N}}$ related to the i_{th} input penalties */

    $(E_{\mathbb{N}})^+_{(i-1)n_{\mathbb{N}}+1 \to in_{\mathbb{N}}, (i-1)n_{\mathbb{N}}+1 \to in_{\mathbb{N}}} = (E_{\mathbb{N}})^-_{(i-1)n_{\mathbb{N}}+1 \to in_{\mathbb{N}}, (i-1)n_{\mathbb{N}}+1 \to in_{\mathbb{N}}} + (\mathbb{N}^i)^T R^i \mathbb{N}^i$

  end

end

Result: $E_{\mathbb{N}}$

Algorithm 4.3: Compressed Linear Term $f_{\mathbb{N}}$ Calculation

Data: $H_{\mathbb{N}}, Q, R, \mathbb{N}, X_e, \bar{U}, U_r, N_p, n_u, n_{\mathbb{N}}$

begin

  $f_{\mathbb{N}} = -H_{\mathbb{N}}^T Q X_e$; // Initial Value

  for $k = 0$ **to** $N_p - 1$ **do**

    for $i = 1$ **to** n_u **do**

      /* Recursive calculation */

      $(f_{\mathbb{N}})^+_{(i-1)n_{\mathbb{N}}+1 \to in_{\mathbb{N}}} = (f_{\mathbb{N}})^-_{(i-1)n_{\mathbb{N}}+1 \to in_{\mathbb{N}}} + (\mathbb{N}_k^i)^T r_k^i (\bar{u}_k^i - u_{r_k}^i)$

    end

  end

end

Result: $f_{\mathbb{N}}$

Algorithm 4.4: Decompression of $\delta\hat{\mathbb{U}}^*$

Data: $\delta\hat{\mathbb{U}}^*, \mathbb{N}, N_p, n_u$

begin

  for $i = 1$ **to** n_u **do**

    $\tilde{W} = \mathbb{N}^i (\delta\hat{\mathbb{U}}^i)^*$; // Store i_{th} input in dummy variable

    for $k = 0$ **to** $N_p - 1$ **do**

      $(\delta\hat{u}_k^i)^* = \tilde{w}_k$; // Assign to the corresponding row of $\delta\hat{U}^*$

    end

  end

end

Result: $\delta\hat{U}^*$

4.2.5 RTI Algorithms

Having defined the required algorithms, the implementation of this approach using RTI Scheme is given in terms of the Preparation and Feedback algorithms given in 4.5 and 4.6 respectively. These algorithms are provided in terms of the "core" algorithms of this chapter, and "core" algorithms 3.4 and 3.6 of chapter 3, to facilitate the verification process of each working part of the proposed approach.

Algorithm 4.5: Input-Parameterised RTI NMPC Preparation Phase Algorithm

Data: $\bar{X}, \bar{U}, \bar{\lambda}, x_{-1}, u_{-1}, Q, R, N_p, n_u, n_{\mathbb{N}}$

begin

- $\bar{x}_0 = f(x_{-1}, u_{-1})$; // Calculate predicted state from previous state and input
- Shift $\bar{X}$,$\bar{U}$, and optionally $\bar{\lambda}$ consistently ; // Initial Value Embedding
- $[A_k, B_k, d_k] = Forward(\bar{X}, \bar{U}, \bar{x}_0, N_p)$; // Run algorithm 3.4
- $[H_{\mathbb{N}}] = CalculateH_{\mathbb{N}}(A_k, B_k, N_p, n_u)$; // Run algorithm 4.1
- $[E_{\mathbb{N}}] = CalculateE_{\mathbb{N}}(H_{\mathbb{N}}, Q, R, \mathbb{N}, n_u, n_{\mathbb{N}})$; // Run algorithm 4.2
- $(M_{\mathbb{N}})_{reorg} = \begin{bmatrix}(\mathbb{N})^T_{reorg} & -(\mathbb{N})^T_{reorg} & H^T_{\mathbb{N}} & -H^T_{\mathbb{N}}\end{bmatrix}^T$; // Form $(M_{\mathbb{N}})_{reorg}$ Matrix

end

Result: $E_{\mathbb{N}}, H_N, M_{\mathbb{N}}, A_k, d_k, \bar{x}_0$

Algorithm 4.6: Input-Parameterised RTI NMPC Feedback Phase Algorithm

Data: $x_0, \bar{x}_0, \bar{X}, \bar{U}, \bar{\lambda}, X_r, U_r, E_{\mathbb{N}}, H_{\mathbb{N}}, M_{\mathbb{N}}, A_k, d_k, Q, R, N_p, n_u, n_{\mathbb{N}}, U_{max}, U_{min}, X_{max}, X_{min}$

begin

- $\delta x_0 = x_0 - \bar{x}_0$; // Calculate state deviation from measurement
- $[D] = FormD(A_k, d_k, \delta x_0, N_p)$; // Run algorithm 3.6
- $X_e = X_r - \bar{X} - D$; // Calculate X error
- $[f_{\mathbb{N}}] = Calculatef_{\mathbb{N}}(H_{\mathbb{N}}, Q, R, \mathbb{N}, X_e, \bar{U}, U_r, N_p, n_u, n_{\mathbb{N}})$; // Run algorithm 4.3
- $(\gamma)_{reorg} = \begin{bmatrix}(U_{max} - \bar{U})_{reorg} \\ (\bar{U} - U_{min})_{reorg} \\ (X_{max} - \bar{X} - D)_{reorg} \\ (\bar{X} + D - X_{min})_{reorg}\end{bmatrix}$; // Form reorganised constraint vector $(\gamma)_{reorg}$
- $[\delta\hat{\mathbb{U}}^*, \bar{\lambda}] = QPSolve(E_{\mathbb{N}}, f_{\mathbb{N}}, (M_{\mathbb{N}})_{reorg}, (\gamma)_{reorg}, \bar{\lambda})$; // Solve the Quadratic Program
- $\delta\tilde{X} = H_{\mathbb{N}}\delta\hat{\mathbb{U}}^*$; // Expansion Step Explicit Calculation
- $\bar{X} = \bar{X} + D + \delta\tilde{X}$; // Calculate new nominal state
- $\delta\hat{U}^* = Decompress(\delta\hat{\mathbb{U}}^*, \mathbb{N}, N_p, n_u)$; // Run algorithm 4.4
- $\bar{U} = \bar{U} + \delta\hat{U}^*$; // Calculate new nominal input

end

Result: $\bar{X}, \bar{U}, \bar{\lambda}$

4.2.6 Generic Computations Comparison

In order to evaluate how well the proposed algorithms perform compared to the standard approach, we performed a generic computation comparison of the main algorithms, namely: algorithms 4.1, 4.2 and 4.3, against their respective counterparts from the standard approach (ie. alg. 3.5, 3.2 and 3.3).

To perform this comparison, we selected the Quadrotor system introduced later in case study 4.3 which has $n_x = 7$ states and $n_u = 4$ inputs. The algorithms were evaluated using different numbers of degrees-of-freedom per input $n_{\mathbb{N}} = [1, 3, 5, 7, 10]$, and different Prediction Horizons $N_p = [60, 120]$ to illustrate the potential advantages. Each of these cases were programmed using automatically generated C++ codes based on the Eigen 3 library, and were tested in Ubuntu 20.04 running with Real-Time priority (ie. chrt -r 99 ./main) on a laptop with an Intel i7-8750 CPU @ 3.9 GHz, and 32 GB DDR4 RAM @ 2,666 MHz, with 120000 runs per case. The test codes were compiled using the (-O3) optimisation C-flag, as well as with the fused-multiply-addition operations (-mfma) and auto-vectorisation (-mavx) flags enabled to use the Advanced Vector Instruction set available in the Intel CPU. The results are gathered in table 4.1 where the minimum computation time of each algorithm is reported, indicating the minimum time that could be achieved if a Real-Time OS would be used.

Case	$n_x = 7, n_u = 4, N_p = 60$						$n_x = 7, n_u = 4, N_p = 120$					
Type/$n_{\mathbb{N}}$	Std	1	3	5	7	10	Std	1	3	5	7	10
$H_{\mathbb{N}}$ (alg. 4.1)	-	3_{17}	$7_{7.1}$	10_{5}	$13_{3.8}$	$18_{2.8}$	-	6_{32}	13_{15}	$20_{9.5}$	27_{7}	$37_{5.1}$
$E_{\mathbb{N}}$ (alg. 4.2)	-	2_{42}	8_{10}	$14_{5.9}$	$24_{3.5}$	$42_{1.97}$	-	3_{108}	$14_{23.2}$	27_{12}	$50_{6.5}$	$79_{4.1}$
$f_{\mathbb{N}}$ (alg. 4.3)	-	1	2	2	3	4	-	1	3	4	5	7
H (alg. 3.5)	50	-	-	-	-	-	190	-	-	-	-	-
E (alg. 3.2)	83	-	-	-	-	-	325	-	-	-	-	-
f (alg. 3.3)	2	-	-	-	-	-	3	-	-	-	-	-
Total	134	6_{22}	$17_{7.9}$	$26_{5.2}$	$40_{3.4}$	$64_{2.1}$	545	10_{54}	30_{18}	51_{11}	$82_{6.6}$	$123_{4.4}$

Table 4.1: Generic Computation Times (in μs) Comparison of Standard and Input Parameterised algorithms for a system with $n_x = 7$ and $n_u = 4$, using different number of degrees-of-freedom/input ($n_{\mathbb{N}} = [1, 3, 5, 7, 10]$) with Prediction Horizons ($N_p = [60, 120]$). The gain factor (α) is indicated in red.

In this table (table 4.1), each of the cases are signaled in the pink coloured cases, with the computing times being reported beneath for the $n_{\mathbb{N}} = [1, 3, 5, 7, 10]$ degrees-of-freedom/input for each of the algorithms (alg. 4.1, 4.2 and 4.3), and the "Total" cyan coloured rows representing the summation of this 3 algorithms. Moreover, most of the table cells contain a red-coloured under-script indicating a "gain factor" (α) related to how many times faster is the specific algorithm-case than the standard counterpart (alg. 3.5, 3.2 and 3.3) visible in the lower-left part of each of the cases. As an example, the calculation of $H_{\mathbb{N}}$ calculation with $n_x = 7, n_u = 4, N_p = 120, n_{\mathbb{N}} = 1$ is $\alpha \approx 32$ faster than with the standard approach ($190 \rightarrow 6\mu s$). Note that row of the linear term $f_{\mathbb{N}}$ is not signaled given it did not present any relevant computational gain. What we can see from this table is that the approach does indeed allow substantial computational saving in this "QP preparation" algorithms, being up to 7.9 and 18 times faster in the "Total" times for $n_{\mathbb{N}} = 3$ coefficients. We will see in the case study 4.3 that for certain systems, this number of coefficients is more than enough to give very good approximation to the actual solution. Finally, note that further computational benefits can result from using reduced degrees of freedom to solve the QP itself. We will see this later in table 4.3.

4.3 Case Study: Laguerre Input Parameterization for Quadrotor Attitude Stabilisation

As discussed earlier, one of the first research directions of this Ph.D. was on the topic of Laguerre-based Input Parameterisation, particularly that described in chapter 3 of [146]. This led to the initial contribution of paper [49] where an Adaptive Laguerre-based MPC controller was developed and physically implemented for the Attitude Stabilisation of a Quadrotor as a fast "auto-tuning" method which resulted in the performance that can be seen in (https://www.youtube.com/watch?v=RSe35TjjBPI). The work in this paper focused predominantly on the use of Recursive Least Squares (RLS) as an "adaptation" method for estimating a set of 2 parameters of a linear SISO model which defined the single-axis dynamics of the Quadrotor based on control allocation theory as described in [72]. Additionally, the proposed method used a set of execution rules, described in section III.B of the paper which were used to improve the performance of the RLS algorithm and protect the controller from periods of poor excitation which can easily lead to the co-variance matrix "exploding", ie. becoming numerically unstable. The resulting model was then used to obtain an unconstrained Laguerre-based MPC controller based on only 2 carefully tuned Laguerre Polynomials that were able to be updated in real-time and were observed to result in excellent performance, as the one seen in the provided link.

Having observed the impressive results from the application of this type of approach in the fairly complex Quadrotor system using only linear SISO modelling techniques with linear MPC implementations, it was considered relevant to extend this application to a more advanced Nonlinear MPC Laguerre-based implementation using Multi-variable modelling, which motivated the contents of this case study. We will demonstrate through this case study that the Laguerre Polynomials approach for this kind of systems can result in surprisingly good performance when compared to the standard NMPC approach, whilst also presenting significantly lower computation times.

4.3.1 The Laguerre Polynomials

Let us begin by defining the dynamics of the Laguerre Polynomials as in [146] given by:

$$\begin{bmatrix} L(1) \\ L(2) \\ L(3) \\ L(4) \\ \vdots \\ L(N_L) \end{bmatrix}_{k+1} = \begin{bmatrix} a_L & 0 & \cdots & \cdots & \cdots & 0 \\ \beta & a_L & 0 & \cdots & \cdots & 0 \\ -a_L\beta & \beta & a_L & 0 & \cdots & 0 \\ a_L^2\beta & -a_L\beta & \beta & a_L & 0 & \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ (-a_L)^{N_L-2}\beta & \ddots & \ddots & \ddots & \ddots & a_L \end{bmatrix} \begin{bmatrix} L(1) \\ L(2) \\ L(3) \\ L(4) \\ \vdots \\ L(N_L) \end{bmatrix}_k \tag{4.24}$$

where a_L is the decay-rate, typically selected based on the desired settling time, $\beta = (1 - a_L^2)$ and N_L is the number of Laguerre coefficients.

By taking $L_0 = \sqrt{\beta}\begin{bmatrix} 1 & -a_L & \cdots & (-1)^{N_L-1}a_L^{N_L-1} \end{bmatrix}^T$ as an initial condition and iterating system (4.24) forward N_p times, the following input-parameterisation can be obtained:

$$\mathbb{N} = \begin{bmatrix} L_0 & L_1 & \cdots & L_{N_p-1} \end{bmatrix}^T \tag{4.25}$$

An example of the first 5 Laguerre Polynomials when using a decay rate of $a_L = 0.5$ with a horizon of $N_p = 30$ can be seen in figure 4.6, representing each of the trajectories in the columns of (4.25).

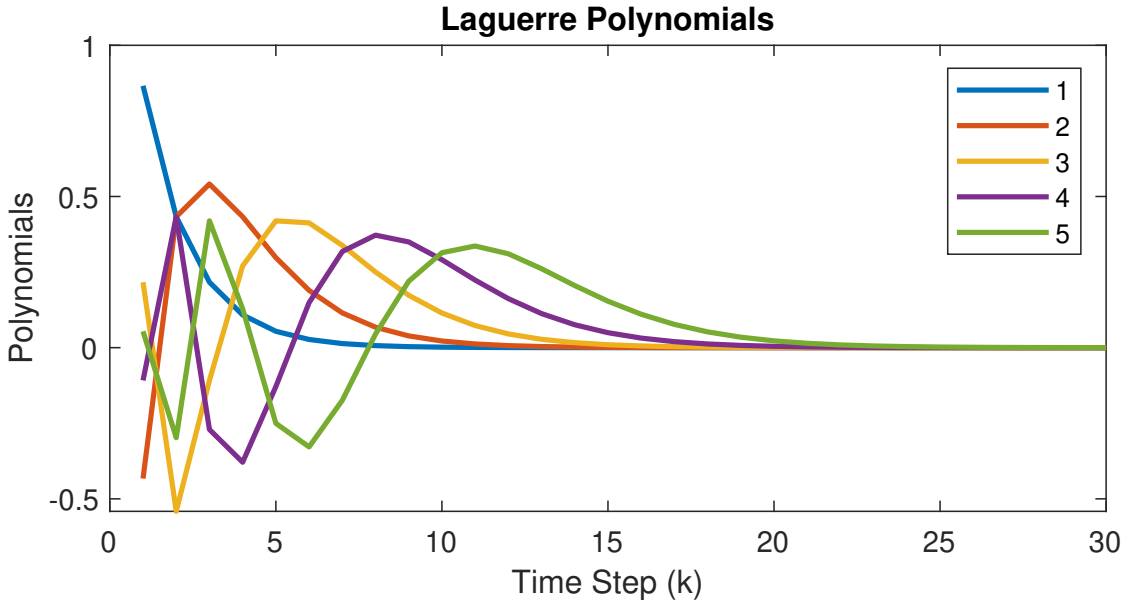


Figure 4.1: Example of First 5 Laguerre Polynomials

To give additional flexibility, input-parameterisation (4.25) was augmented with a constant column of ones to capture any required bias in the input. Based on this, the optimisation can then select any linear combination of these trajectories to "replicate" the trajectory obtained by the standard NMPC.

4.3.2 Quadrotor Model

To simulate the Quadrotor, we selected a slightly simplified version of the nonlinear model from [154], specifying the quaternion and rate dynamics as:

$$\underbrace{\begin{bmatrix}\dot{q}_0\\ \dot{q}_1\\ \dot{q}_2\\ \dot{q}_3\end{bmatrix} = \frac{1}{2}\begin{bmatrix}0 & -p & -q & -r\\ p & 0 & r & -q\\ q & -r & 0 & p\\ r & q & -p & 0\end{bmatrix}\begin{bmatrix}q_0\\ q_1\\ q_2\\ q_3\end{bmatrix}}_{\text{Quaternion Dynamics}} \qquad \underbrace{\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}k_1(\omega_1^2-\omega_3^2)\\ k_2(\omega_2^2-\omega_4^2)\\ k_3(\omega_1^2-\omega_2^2+\omega_3^2-\omega_4^2)\end{bmatrix}}_{\text{Rate Dynamics}} \tag{4.26}$$

where the $\overrightarrow{q} = [q_0, q_1, q_2, q_3]$ is known as the quaternion, $[p, q, r]$ are angular-rates in the body axis, $[\omega_1, \omega_2, \omega_3, \omega_4]$ are the angular velocities of the propellers, and constants $k_1 = k_2 = 0.0613$ and $k_3 = 0.0184$ were calculated using equation (9) and parameters of Table I of the paper [154].

The resulting differential equations were simulated using algorithm 3.1 with a sampling time of $T_s = 0.05$ (s) and using $N_s = 8$ intermediate steps, to improve the accuracy of the integrator. One particular requirement for simulating this system is to maintain a unitary norm on the quaternion $(\overrightarrow{q})$. This can be achieved by embedding the corresponding non-linear equality into the optimisation as in [154]. However, a slightly simplified procedure was implemented in this case study where the quaternion was normalised at the end of every simulation step from algorithm 3.1.

4.3.3 Performance Comparison

To evaluate the performance of the proposed approach, the resulting NMPC controller was tested for the task of "reference tracking". In this case, the objective of the optimisation was to reach a given orientation defined by random quaternion reference $\overrightarrow{q}_{ref}$ which was calculated by assigning a random reference in the well known Euler angles (Roll/Pitch/Yaw), and converting them into the equivalent quaternion via the "eul2quat" function from Matlab. The prediction horizon of the optimisation was set at $N_p = 20$, and the penalisation weights were selected $q_{k+i} = diag([0, 1, 1, 1, 0, 0, 0])\ \forall i = [1, N_p]$ and $r_{k+i} = diag([0.01, 0.01, 0.01, 0.01])\ \forall i = [0, N_p - 1]$. The optimisation was solved using single-shooting based on the Relative Input Parameterised NMPC framework, initialised as in [154] with all the motors running starting from at a hover condition as $x_0 = [1, 0, 0, 0, 0, 0, 0]^T$ and $\bar{u}_{k+i} = [40, 40, 40, 40]^T\ \forall i = [0, N_p - 1]$. Moreover, a reference input of $u_{r_{k+i}} = [40, 40, 40, 40]^T\ \forall i = [0, N_p - 1]$ was selected which is the required input steady state to reach any desired steady state orientation. Finally, no input or state constraints were imposed to focus on the replicability properties of the Laguerre Polynomials. The system was simulated for $N = 300$ times for different numbers of Laguerre Polynomials $N_L = [1, 2, 3, 4, 5]$ with a decay rate of $a_L = 0.5$ as in figure 4.1, and the sub-optimality w.r.t. to the standard NMPC solution $\Delta J = (J_{Laguerre}/J_{Std} - 1) \times 100$ for each of the cases was stored. Table 4.2 gathers the average and standard deviations obtained from the suboptimality of each of the aforementioned cases, and an example simulation comparing the solution using only 1 Laguerre polynomial to the standard NMPC approach can be seen in figure 4.2.

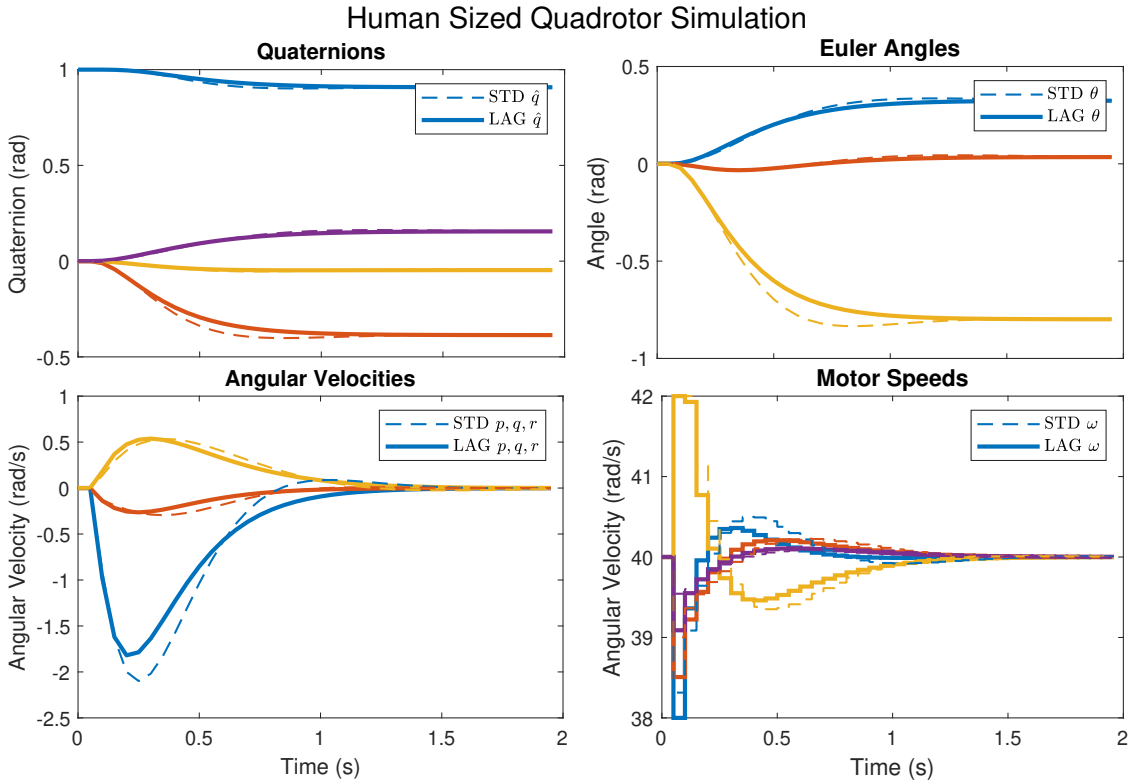


Figure 4.2: Example Comparison of Laguerre (−) and Standard (−−) NMPC for the Quadrotor using only $N_L = 1$ Laguerre Polynomial with a Suboptimality of 3.48.

N Polynomials	1	2	3	4	5
Average Suboptimality (ΔJ)	2.111	0.0834	0.0361	0.00842	0.000131
Standard Deviation	0.794	0.115	0.103	0.102	0.00412

Table 4.2: Suboptimality comparison for different Laguerre Polynomials applied to the Quadrotor

From table 4.2 it can be appreciated how the Laguerre Polynomials are able to reach extremely competent performance with average suboptimalities of 0.0834 with as low as $N_L = 2$ Laguerre polynomials, a total of $n_{\mathbb{N}} = 3$ decision variables per input considering the extra bias. This is to be expected given the system is basically a 2nd order with a few cross-coupled terms having multi-variable actuation on each of the axis. This can also be appreciated in figure 4.2 where the solution with only 1 polynomial results in good (arguably better) performance with virtually no overshot seen in the response of the Euler Angles. As a side note, it was found that tuning the Quadrotor of the UKACC 2018 paper [49] proved to be easier when using Laguerre than when using MPC, most likely related to the damping property of the Laguerre Polynomials embedded into the solutions. Lastly, as it is to be expected, the performance can be seen to become more consistent in terms of standard deviations as the number of Laguerre Polynomials increases $N_L = [1 \rightarrow 5]$. Thus, this gives an example of the powerful performance that can be obtained when using Laguerre Polynomials in this type of systems.

4.3.4 Computation Times Comparison

To evaluate the computational performance, the optimisation was tested using automatically generated C++ codes based on the RTI algorithms of section 4.2.5, supported by the Eigen 3 Library. The codes were tested for the same number of Laguerre Polynomials $N_L = [1 \rightarrow 5]$, using prediction horizons of $N_p = [20, 60]$. Each code was run for a total of 1000 different simulations of $T = 5$ seconds, giving a total of $100,000$ optimisations. The codes were run in the same conditions as in the generic computations test of section 4.2.6, ie. same laptop, running with real-time priority and using the same optimisation flags. To compare the resulting performance, the solution with the ACADO toolkit was also obtained using QPOASES solver with the "FULL_CONDENSING_N2" $O(N^2)$ option enabled, and was run for the same amount of simulations. The results of this comparison can be seen in table 4.3 where the average computation times of the unconstrained solutions are presented, and the gain factor is indicated in the red under-scripts. As it can be seen, the solution with prediction horizon $N_p = 20$ resulted in rather small gains in the range of $3.1 - 4$ times faster. However, the solution with $N_p = 60$ presented significantly higher gains of up to 31, with the minimum being 25 times faster. Indeed, even in the case where constraints would be involved, the optimisation could still perform significantly higher amount of QP iterations before becoming inefficient thanks to the reduced number of degrees of freedom. Thus, this gives another example of the potential of using this type of approach.

N_p	20						60					
(N_L)	ACADO	1	2	3	4	5	ACADO	1	2	3	4	5
Unc. Time	128	32_{4}	$33_{3.9}$	$35_{3.7}$	$41_{3.1}$	$41_{3.1}$	2487	79_{31}	80_{31}	85_{29}	91_{27}	98_{25}

Table 4.3: Average Computation Times Comparison for the Quadrotor using $N_L = [1 \rightarrow 5]$ Laguerre Polynomials, and ACADO toolkit with Different Horizons $N_p = [20, 60]$. Gain factor indicated in red.

4.4 Case Study: A Fourier/Laguerre Input Parameterisation for an Unmanned Surface Vehicle

The second case study that we considered for implementation of the Input-Parameterised framework was that of the Unmanned Surface Vehicle (USV) from the work done in collaboration with Ph.D. student Daniel R. McCullough which resulted in the IFAC World Congress 2020 publication [100]. In this work, the main focus was to achieve the minimisation of hull-impact forces related to incoming waves when going in a straight direction into them, which could ultimately serve as a mechanism to navigate more safely through rough sea states. One of its interesting findings was that the optimal NMPC solution for this task can be precisely related to the first 2 harmonics of the so called "encounter frequency" (ω_e), and a "bias" for maintaining a desired speed, all of which can be seen in the amplitude spectrum of the NMPC controller visible in figure 5 of the paper. This raised the question of whether it would be possible to embed a frequency based input-parameterisation such as a Fourier-basis function, that could replicate this solution more efficiently which motivated the contents of this case study.

Remark 4.6. *No Modelling Work*

This case study won't introduce any dynamic modelling given they were developed in large by Ph.D. student Daniel R. McCullough based on work from [62]. Instead, please refer to the contents already used in the paper for reference. This will allow us to focus on the relevant concepts rather than specifics.

4.4.1 The Fourier/Laguerre Input Parameterisation

The most obvious choice for this situation would be to embed a Fourier basis-function at the predicted average encounter frequency (ω_e), with an additional "bias" to maintain the desired velocity. This, in theory, would capture the most complete information relevant to this problem based on the "offline/a-priori" knowledge that the solution to the objective function can be represented by a couple of harmonics, and a "bias". However, given this would only capture the "steady oscillating state", ie. when the solution is going through the waves with virtually no disturbances, it was deemed relevant to consider an extra decision variable that would allow quick cancellation of any potential disturbance, whilst keeping the main Fourier-basis function as a target. The obvious choice for this, given the knowledge obtained from the implementation of Laguerre Polynomials in the previous case study, was a first order exponentially decaying term (ie. the first Laguerre term), which could allow the system to recover from a disturbed state to the "steady oscillating target" in an exponentially decaying manner.

Thus, the desired input-parameterised trajectory can be expressed in continuous time in the form:

$$u(t) = A + A_e e^{(-\frac{t}{\tau})} + A_1 \cos(\omega_1 t + \phi_1) + A_2 \cos(\omega_2 t + \phi_2) \tag{4.27}$$

where $u(t)$ is the input to the USV; A, A_e, A_1, A_2 are constants; ω_1 and ω_2 are the two frequencies to be considered for the co-sinusoidal terms, in this case $\omega_2 = 2\omega_1$; and ϕ_1 and ϕ_2 are the phases of the co-sinusoidal terms.

Using trigonometric identity $\cos(x + y) = \cos(y)\cos(x) - \sin(y)\sin(x)$ in equation (4.27) leads to:

$$u(t) = A + A_e e^{(-\frac{t}{\tau})} + A_1(\cos(\phi_1)\cos(\omega_1 t) - \sin(\phi_1)\sin(\omega_1 t)) + A_2(\cos(\phi_2)\cos(\omega_2 t) - \sin(\phi_2)\sin(\omega_2 t)) \tag{4.28}$$

This can be simplified to eq. (4.29), simply by considering $\cos(\Phi_1), \sin(\Phi_1), \cos(\Phi_2), \cos(\Phi_2)$ to become parts of the constants A_1, A_2, resulting in the "alternative" constants $A_{11}, A_{12}, A_{21}, A_{22}$.

$$u(t) = A + A_e e^{(-\frac{t}{\tau})} + A_{11}\cos(\omega_1 t) + A_{12}\sin(\omega_1 t) + A_{21}\cos(\omega_2 t) + A_{22}\sin(\omega_2 t) \tag{4.29}$$

Assuming the frequencies of interest (ω_1 and ω_2), and the decaying term (τ) are provided, we can embed this desired behaviour into the inputs of the optimisation by evaluating the terms of (4.29) at all time-steps $k = [1, N_p]$, resulting in the input-parameterisation matrix:

$$\mathbb{N} = \begin{bmatrix} 1 & e^{(-\frac{T_s}{\tau})} & \cos(\omega_1 T_s) & \sin(\omega_1 T_s) & \cos(\omega_2 T_s) & \sin(\omega_2 T_s) \\ 1 & e^{(-\frac{2T_s}{\tau})} & \cos(\omega_1 2T_s) & \sin(\omega_1 2T_s) & \cos(\omega_2 2T_s) & \sin(\omega_2 2T_s) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & e^{(-\frac{N_p T_s}{\tau})} & \cos(\omega_1 N_p T_s) & \sin(\omega_1 N_p T_s) & \cos(\omega_2 N_p T_s) & \sin(\omega_2 N_p T_s) \end{bmatrix} \tag{4.30}$$

where T_s is the sampling time of the system, and $A, A_e, A_{11}, A_{12}, A_{21}, A_{22}$ are now the input parameterised decision variables from (4.29) which are gathered in the compressed input vectors ($\hat{\mathbb{U}}$ or $\delta\hat{\mathbb{U}}$).

Once this is defined, the optimisation will then find the optimal coefficients that optimise the relevant cost function, knowing/enforcing in advance that the input trajectory CAN ONLY take the form of input parameterisation (4.27). However, note that this parameterisation would embed a discretised zero-order-hold (ZOH) version of equation (4.27). Moreover, note that this approach is fundamentally different to those used in "spectral" or "pseudo-spectral" collocation methods [41], given we do not embed this in the states/state derivatives as well, nor we use differentiation matrices which altogether transform the system dynamics into a linear system of equations that is solved for obtaining the simulation of the system [141, 148].

4.4.2 Performance Comparison

To evaluate the performance of the proposed input-parameterisation, the USV was simulated under the same basic conditions discussed in the paper [100], ie. considering the boat heading directly into oncoming periodic waves of 1 meter height with a frequency of $\omega = 0.5$ (rad/s). The system was simulated for $T = 60$ (s) using Explicit Euler algorithm 3.1 with a sampling time of $T_s = 0.08$ (s), and $N_s = 20$ intermediate steps based on the USV model of equation (15) of the paper. The NMPC had a prediction horizon of $N_p = 200$, resulting in a prediction window of $T_p = 16$ (s) which captured just over a complete harmonic of the wave at zero-velocity. The objective function was the minimisation of three main quadratic costs, namely: the surge velocity (ν) deviations from a velocity reference; the hull wave-induced forces (τ_w), considered as nonlinear outputs; and the propeller input (u) deviations from an input reference, which can be solved using the output-cost framework (3.2) introduced in chapter 3. The weights of the NMPC were the selected as: $q_{\nu_{k+i}} = 10\ \forall i = [1, N_p]$ for penalising the velocity (ν) with the references selected as $y_{\nu_{r_{k+i}}} = 5$ (m/s) $\forall i = [1, N_p]$; $q_\tau = 1.4 \times 10^{-7}$ for penalising the hull wave-induced forces (τ_w); and $r_{k+i} = 0.01\ = [0, N_p - 1]$ for penalising the deviations of the propeller input from a reference input of $u_{r_{k+i}} = 10.0598\ \forall i = [0, N_p - 1]$, which is the input required to achieve the reference velocity of 5 (m/s) if there was no wave. Finally, the tuning weight of equation (22) for "virtually" modifying the hull-force equation was selected as in the paper ($\alpha = 100$).

Regarding the setup of the proposed input-parameterisation, the method used the predicted average of the encounter frequency obtained with the nominal input guess $\bar{U}$, ie. $\omega_{e-avg} = \sum_{k=1}^{N_p}(\omega + \frac{\omega^2}{g}\bar{\nu}_k)/N_p$ to define the frequencies $\omega_1 = \omega_{e-avg}$ and $\omega_2 = 2\omega_1$ required by the input parameterisation (4.30). On the other hand, a value of $\tau = 1$ was selected for the first order exponential decaying term, based on the relatively slow dynamics of the boat. Two metrics were selected for comparison, namely: sub-optimality and disturbance rejection. No computation times comparison was made for this system given this has already been performed in various points earlier in the chapter.

Suboptimality Comparison

The first performance metric of interest was how close the standard RTI NMPC solution was able to be replicated using the proposed approach. As the objective is to embed the desired frequency content in the actual input (u), the Non-Relative Input-Parameterised framework (4.4) was used. Interestingly, the proposed approach resulted in almost indistinguishable responses, even obtaining a slightly better overall performance with a percentage "suboptimality" of $\Delta J = (J_{Spectral}/J_{NMPC} - 1) \times 100 = -0.0111$. The reason this can be happening is because the standard NMPC solution does not embed any frequency information about the closed loop response into the solution, and as a result, the optimisation was observed to be "relaxing" the solution near the end of the prediction horizon at every time step. This can be seen in figure 4.3 where the predicted trajectories of both solutions are presented, and the standard NMPC solution can be seen to be breaking the pattern near the end of the horizon resulting in relatively small differences in the predictions, whereas the proposed approach, hereafter denominated "Spectral NMPC", enforces the specified pattern in the input. This causes the standard NMPC optimisation to be slightly ill-posed [122] where the predicted trajectories differ from the closed loop responses, potentially causing problems in the decision making process.

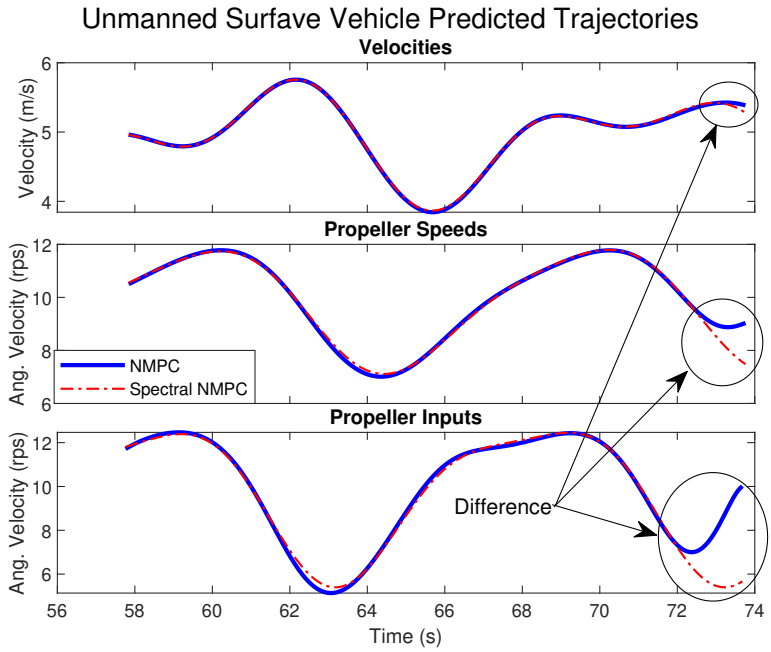


Figure 4.3: Predicted Trajectories of Standard NMPC and Spectral NMPC for the USV

The frequencies of the closed loop solutions obtained with both approaches, Standard NMPC and Spectral NMPC, were analysed using Fast Fourier Transform, and were confirmed to both contain the exact same pair of frequencies at slightly different amplitudes. This can be seen in the single sided amplitude spectrum of the input response (u) given in figure 4.4.

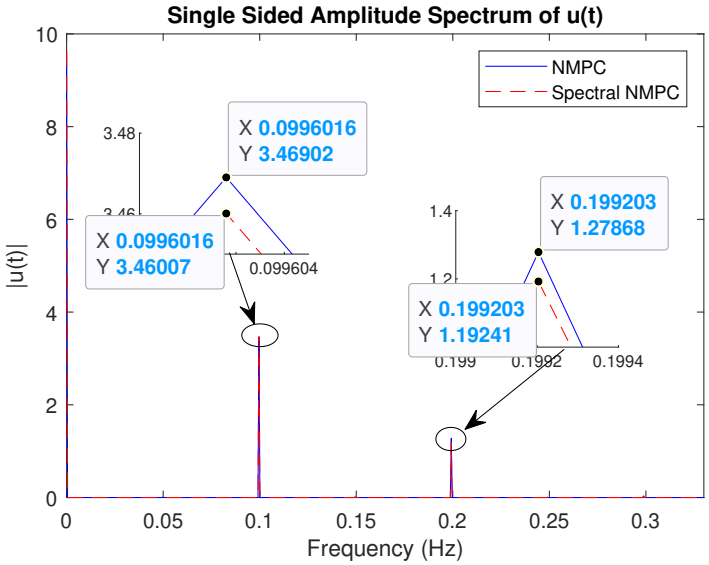


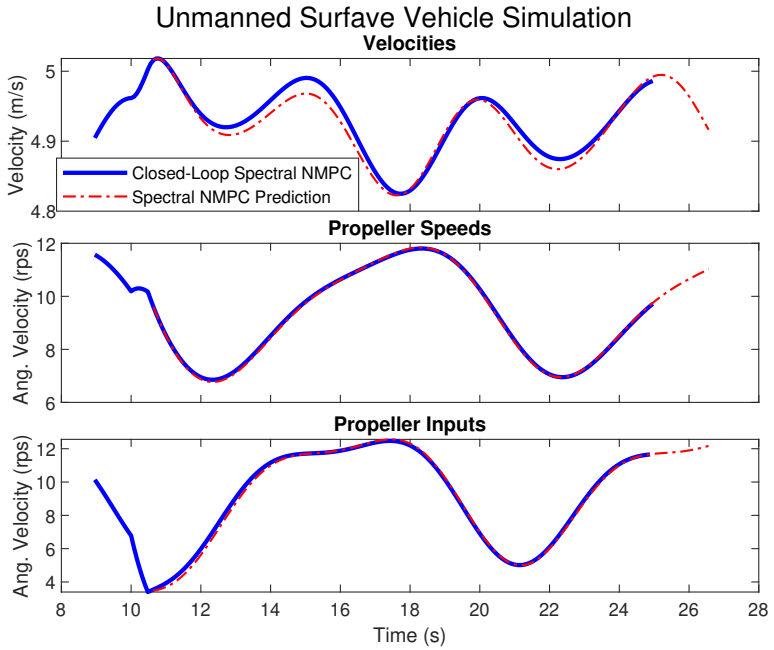
Figure 4.4: Input Spectrum Comparison for the Unmanned Surface Vehicle problem

Disturbance Rejection Comparison

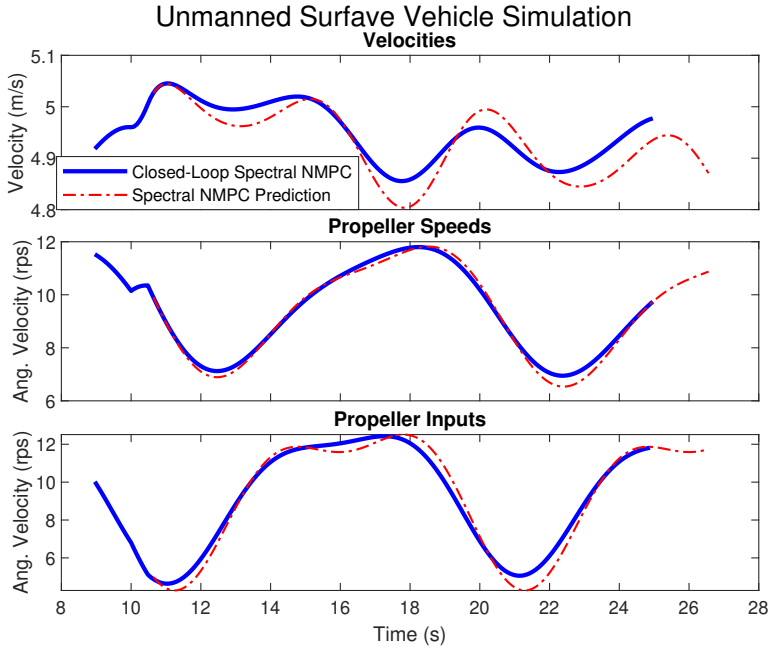
The second performance metric of interest was how well the approach responded to disturbances compared to the standard approach. To evaluate this, an un-measurable disturbance of 5 revolutions per second was injected in the input as $u_{k-real} = u_k + 5$ at simulation time $t = 10\ (s)$, and kept constant for 0.5 seconds, eg. emulating a vacuum encountered throughout a wave. The responses against this disturbance were tested with three main methods: the standard NMPC, the proposed Spectral NMPC with Laguerre term, and the Spectral NMPC without the Laguerre term, denominated "Fourier-Only". A comparison of the percentage sub-optimalities obtained by the two non-standard approaches w.r.t. to the standard NMPC solution subject to the aforementioned disturbance can be seen in table 4.4. From this we can see that the proposed approach, ie. the Spectral NMPC with the Laguerre term, resulted in slightly improved approach when compared to the standard NMPC, with a slightly larger improvement when compared to the Fourier-Only solution. The result of this comparison is by no means unexpected given the predicted trajectories were observed to give much accurate predictions of the closed loop responses in the case where the Laguerre term was included. This can be seen in figure 4.5 where the predictions with and without the proposed Laguerre first order term, depicted by the dot-dashed red lines, are compared to the actual closed loop responses, depicted by the blue lines.

Case/Solution	Suboptimality (ΔJ) in %
Disturbance Spectral w/Laguerre	-0.0070
Disturbance Spectral Fourier-Only	0.0920

Table 4.4: Suboptimality Comparison for disturbance rejection with and without the proposed Fourier/Laguerre Input Parameterised Solution applied to the Unmanned Surface Vehicle



(a) With



(b) Without

Figure 4.5: Comparison of Disturbance Rejection With (4.5a) and Without (4.5b) the additional Laguerre first order term

Thus, this case study gives a relative simple example of the advantages that can be obtained from embedding a-priori knowledge into the system such as desired frequency responses. Moreover, the system was observed to have better disturbance rejection properties with the proposed additional Laguerre term which by no means was an unexpected result given the use of a purely periodic signal for handling non-periodic disturbed behaviour is conceptually insufficient.

4.5 Case Study: Chebyshev Polynomials for Wave Energy Converters

The last case study that was considered for implementing the input-parameterisation framework was on the Wave Energy Converter (WEC) from the IFAC World 2020 conference [59], resulting in an extension to the work presented in the paper. We will discuss the exact method presented in the paper in the following chapter, particularly in section 5.4, given the contents of the article are related to the methods presented in it. For this case study, we decided to apply the so called "Chebyshev Polynomials" following discussions with co-author Juan G. about the so called "pseudo-spectral collocation methods" and their popularity among WEC devices [41, 43, 44, 63, 103], which raised the question of whether they posed a particular advantage other than the so called "spectral accuracy". As this is only a short extension, only this basis function was considered. Moreover, as the methods discussed in this chapter are not considered to be "collocation methods", this comparison will be kept conceptual.

Remark 4.7. *No Modelling Work*

This case study won't introduce any dynamic modelling given they will be introduced briefly in the following chapter in case study 5.4, and they were developed in large thanks to expert WEC knowledge by Ph.D. student Juan Guerrero-Fernandez. Instead, please refer to the contents already used in the paper for reference. This will allow us to focus on the relevant concepts rather than specifics.

4.5.1 The Chebyshev Polynomials

Let us begin by defining the well known Chebyshev Polynomials as in [148], given by:

$$T_1(t) = 1 \tag{4.31}$$

$$T_2(t) = t \tag{4.32}$$

$$T_n(t) = 2tT_{n-1} - T_{n-2} \qquad \forall n > 2 \tag{4.33}$$

By evaluating this polynomials on the range $t = [-1, 1]$ divided into the N_p steps of the prediction horizon to be used, and assigning them to the inputs in the range $k = [0, N_p - 1]$, we obtain an input parameterisation matrix of the form:

$$\mathbb{N} = \begin{bmatrix} T_1^{t_0} & T_2^{t_0} & \cdots & T_{n_{\mathbb{N}}}^{t_0} \\ T_1^{t_0+t_s} & T_2^{t_0+t_s} & \cdots & T_{n_{\mathbb{N}}}^{t_0+t_s} \\ T_1^{t_0+2t_s} & T_2^{t_0+2t_s} & \cdots & T_{n_{\mathbb{N}}}^{t_0+2t_s} \\ \vdots & \vdots & \cdots & \vdots \\ T_1^{t_f} & T_2^{t_f} & \cdots & T_{n_{\mathbb{N}}}^{t_f} \end{bmatrix} \tag{4.34}$$

where $t_0 = -1$, $t_s = 2/N_p$, $t_f = 1 - t_s$, and the notation T_i^t is the i_{th} polynomial evaluated at time t.

As an example of application of this, consider the prediction horizon of $N_p = 100$ used for the WEC paper [59]. Figure 4.6 shows the first 5 polynomials for this case which can be obtained by plotting each of the columns of the resulting input-parameterisation matrix (4.34).

Based on this input-parameterisation, the optimisation can now select any linear combination of this polynomials to achieve the desired result, whether that is satisfying a constraint or performing a desired manoeuvre in the WEC device to maximise the energy generation.

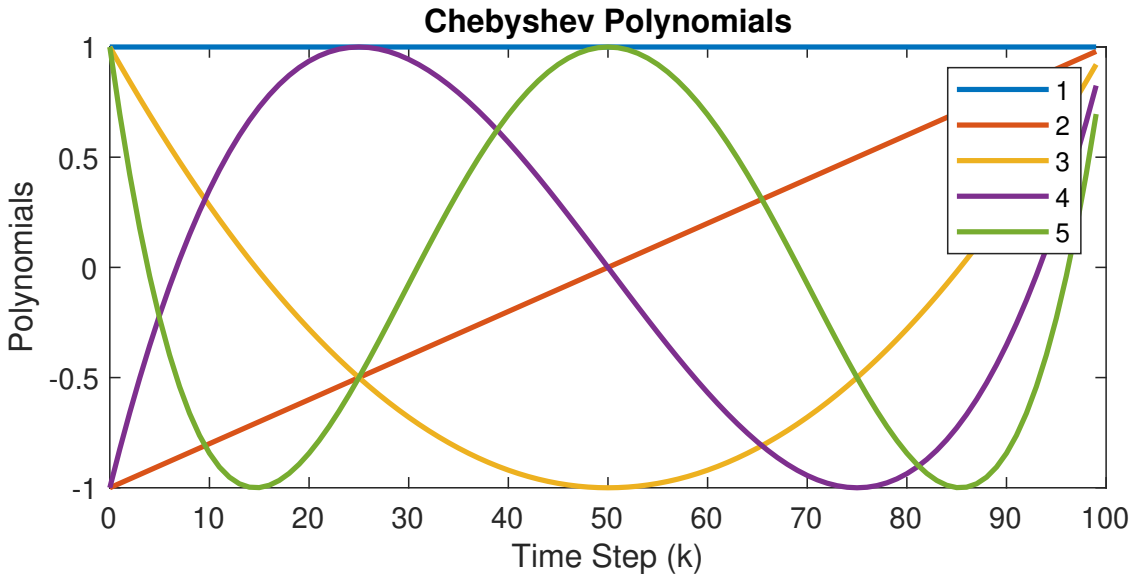


Figure 4.6: Example of First $n_{\mathbb{N}} = 5$ Chebyshev Polynomials

4.5.2 Energy Extraction Comparison

To evaluate the performance of the Chebyshev Polynomials parameterisation (4.34), we performed a simulation of the WEC system using the exact same simulation parameters specified in the paper, ie. same constraint limits on the input and input increments, same parameters specific to the buoy and wave excitation as well as same prediction horizon $N_p = 100$ and same simulation time of $T = 600$ (s). The approach was then tested using various number of Chebyshev Polynomials ($n_{\mathbb{N}} = [5, 10, 20]$) to match the $N_{E_{\mathbb{N}}} = 20$ degrees of freedom used by the Moving Window Blocking (MWB) approach presented in the paper. The results obtained from this were then compared to the Standard MPC with Full-Degrees of Freedom (F-DoF), which is considered here (and in the paper) as the approach with the "maximum" energy extraction that could be obtained with the selected prediction horizon. The comparison can be seen in table 4.5.

From this table we can see that the approach requires at least $n_{\mathbb{N}} = 20$ Chebyshev Polynomials to start to even become close to the 98.8 efficiency of the MWB approach proposed in the paper, as seen in table 5.8 which will be discussed later in section 5.4. Thus, we can conclude that from a energy extraction perspective, the proposed MWB approach of the paper still outperforms this type of solution when applied to this particular system.

Method	Energy Extracted (MJ)	Efficiency (%)
Std MPC	307	100
CHEV ($n_{\mathbb{N}} = 20$)	301	97.9
CHEV ($n_{\mathbb{N}} = 10$)	272	88.6
CHEV ($n_{\mathbb{N}} = 5$)	246	80.3

Table 4.5: Efficiency Comparison of Chebyshev Polynomials and Standard Linear MPC applied to the Wave Energy Converter problem

In order to understand the reason for this significantly degraded performance, it was decided to review the predicted trajectories to look for any inconsistency or ill-posedness that could be affecting the decision making process. It was found that the Chebyshev approach was presenting a Gibbs-type phenomena when the optimal solution was saturated in the input constraints, which in this particular system was nearly 60% of the time. This can be seen in figure 4.7 where the predicted trajectories of both Standard and Chebyshev approaches can be seen, and the Gibbs-type phenomena is signaled in the inner graph of the PTO Forces. What this indicates is that the Chebyshev Polynomials approach is fundamentally incapable to replicate constrained input trajectories adequately which consequently can result in significantly degraded performance. Note that this is also true for the other two input-parameterisation strategies presented in the previous case-studies.

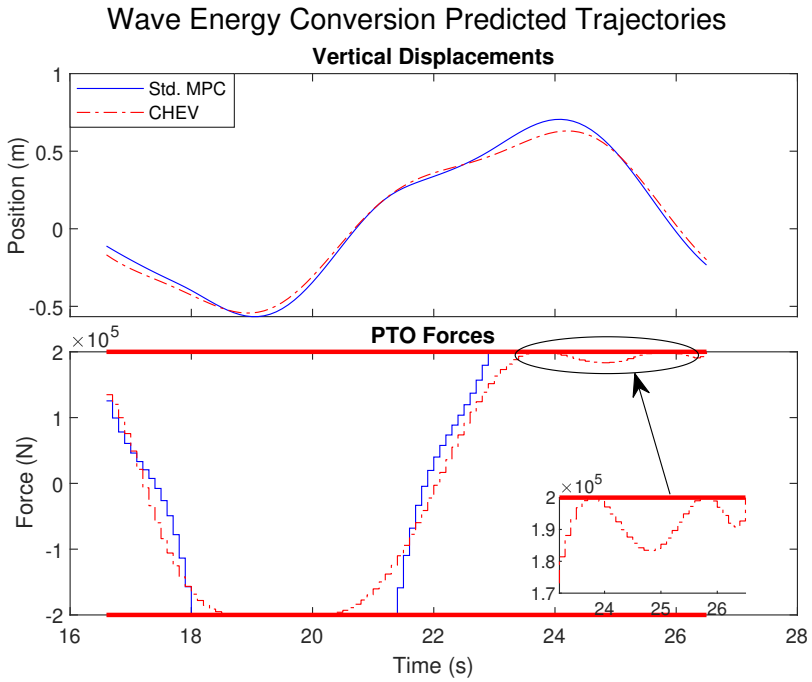


Figure 4.7: Example of Chebyshev Polynomials Incapability for Constrained Solutions. The Inner Graph shows the emergence of a Gibbs-type phenomena at the constraint.

4.5.3 Computation Times Comparison

Lastly, the other metric of interest was the computation time required to perform the optimisation when using the aforementioned approaches, namely: the Std. F-DoF MPC, and the Chebyshev Polynomials with $n_{\mathbb{N}} = [5, 10, 20]$. Table 4.6 presents the average computations times of each solution along with other relevant metrics such as number of QP iterations. All the QPs were solved using the "quadprog" function of Matlab R2019b with the interior-point method, and the computation times involve both, the update of the linear term $(f/f_{\mathbb{N}})$ and the solution of the QP itself, noting that the other matrices/vectors required by the QP were pre-stored offline given the system is linear as discussed in the paper. From this table we can clearly see how the CHEV approach presents a significantly degraded performance with the Standard approach being faster for all $n_{\mathbb{N}} = [5, 10, 20]$, most likely related to the inability of the Chebyshev Polynomials to deliver the very basic input-constrained solutions as discussed in figure 4.7, indicating another disadvantage of applying this method to the WEC problem.

Method	Avg. opt. time (ms)	Avg. opt. time per iter. (ms)	Avg. num. of QP Iterations	Computational Gain
Std MPC	28 ± 4.5	3.43 ± 0.5	8.19 ± 0.79	-
CHEV ($N = 20$)	93.1 ± 24.3	9.88 ± 0.97	9.44 ± 2.67	0.3
CHEV ($N = 10$)	82.5 ± 10.8	9.13 ± 0.79	9.05 ± 1.06	0.326
CHEV ($N = 5$)	82.2 ± 38.7	9.39 ± 0.83	8.8 ± 4.6	0.333

Table 4.6: Computation Times Comparison of Chebyshev Polynomials and Standard Linear MPC when using "quadprog" of Matlab R2019b to solve the optimisation of the Wave Energy Converter problem

4.6 Summary

In summary, this chapter introduced the concept of Input Parameterised NMPC solutions which offer a viable alternative to the Standard NMPC approach with the potential advantage of allowing significantly faster computation times, both in terms of preparation and solution of the underlying QPs, as well as reduced memory usage due to the reduction in the required degrees of freedom. Additionally, this type of methods allow the user to embed various type of characteristics into the solutions such as frequency response and desired settling times whilst generally being able to obtain good performance determined by how well the input-parameterisation is designed for the specific system.

Throughout the chapter we introduced the required methods and mathematical models required by two type of approaches, namely: the non-relative and relative input parameterisations, along with theorem 4.1 which gives a condition on which the solution of both approaches would be equivalent, something which will be used for reference in the following chapter 5. Moreover, the chapter introduced a set of algorithms for its efficient implementation based on the RTI Scheme in section 4.2 where a generic computations comparison was performed to evaluate how well the approach performs compared to the standard NMPC approach for the Quadrotor system of case study 4.3 where significant computation gains were obtained. The chapter concludes with 3 case studies, including: a Laguerre-based NMPC for an Attitude Stabilisation of a Quadrotor; a Fourier/Laguerre Input-Parameterisation for an Unmanned Surface Vehicle; and a Chebyshev Polynomials Parameterisation for a Wave Energy Converter, all of which are used to illustrate some of the relevant properties, advantages and disadvantages of the Input-Parameterisation techniques, including good sub-optimal performance, improved disturbance rejection and faster computation times. The key message of this chapter is the understanding of this basic technique can open the NMPC optimisation methodologies to a wide range of alternatives which may be relevant for specific systems or situations. However, a potential disadvantage is the requirement of having "a-priori" knowledge of the system or optimisation in order to be able to properly design the underlying input structures for which a systematic procedure may be hard to define. Nonetheless, we consider this to be relevant given an offline analysis of a system can typically be performed to select and/or investigate appropriate methods among the existing or common basis functions to be used in an online optimisation setup.

Finally, one of the interesting results that was shown in this chapter was the fundamental incapability of the Chebysheb Polynomials to replicate constrained solutions which results in degraded performance and significantly increased computations times when compared to the standard approach. Note that this incapability is fundamentally present in all the different input-parameterisations used for the case study, which motivated the research direction of this thesis into the so called “Move Blocking” approaches, introduced in the following chapter 5.

Chapter 5

Shifting Strategy for Blocked Nonlinear Model Predictive Control

In the previous chapter, we introduced the concept of input parameterised solutions using a general framework which is capable of embedding a wide variety of input-structures such as Laguerre Polynomial, Fourier Polynomials, Chebyshev Polynomials, and so on, as demonstrated in the case studies provided along the chapter. These input-structures allow the user to reduce the degrees of freedom substantially, potentially allowing a significant decrease in computation times whilst being able to maintain good performance (determined by how well the input-structure is designed according to a certain target/objective), and allowing the user to embed certain characteristics such as desired frequency response and settling times into the solution. Although these advantages proved to be quite useful in certain scenarios and/or applications, one of the key disadvantages or challenges that was observed when using the aforementioned variations of input-structures was constraint satisfaction of input inequalities, which can significantly affect the optimisation (as seen in the Chebyshev Case Study of section 4.5), despite these being the simplest of the inequality constraints in the original optimisation. This suggest that these constraints shouldn't be the ones causing problems at all.

This particular disadvantage motivated the research direction of this thesis into the so-called "Move-Blocking" methods [18, 130, 133, 134]. As it will be seen in this chapter, these type of methods belong to the general input-parameterised solution framework introduced in the previous chapter, with the main difference of presenting a particular form or "sparsity" in their input-structure which allows significantly easier handling of the input inequality constraints. However, when not handled properly, this type of methodology can present significant problems related to stability and recursive feasibility of the optimisation. Thus, this chapter aims to address this with the use of the "Shifting Strategy" proposed by the author of this thesis; a concept built upon the very foundation (or requirement) of the IVE strategy of the RTI Scheme introduced in chapter 3, representing one of the key and most important contributions of this thesis which the following chapters (particularly chapters 7 and 8) will be based on. Additionally, following the same reasoning, it extends the scope of this strategy from its application on the input-structure, to its application on the outputs and inequality constraints of the optimisation, thus resulting in OCPs of reduced size in terms of degrees-of-freedom, targets (or shooting points) and inequality constraints, allowing significant computational benefits whilst being able to preserve excellent performance, in many cases better than the standard alternatives.

The chapter is organised as follows: Section 5.1 introduces the main concepts behind the proposed Shifting Strategy, along with the relevant theory and definitions required to validate the whole methodology. An important part of this section is subsection 5.1.2 where the concept of shifting input blocks is introduced, along with important discussions and details including the novel concept of the proposed Ideal Prediction Horizon (supported by theorem 5.1), the resulting input-structure, as well as Alternative Blocking Structures that can be used which may not be immediately evident and can ultimately motivate its usage further. Following this, subsection 5.1.3 presents the extension of the proposed Shifting Strategy to the shooting points and, more importantly, the inequality constraints of the optimisation along with Alternative Shifting approaches that may be used. The section ends with a discussion of the proposed optimisation framework to be used in subsection 5.1.4, followed by the relevant nominal stability and recursive and feasibility proofs provided in subsection 5.1.5. Afterwards, section 5.2 discusses the algorithm details required for an efficient implementation of the strategy, including an extension of the $O(N)$ and $O(N^2)$ algorithms presented in section 3.3.1 of chapter 3 which forms part of the main contributions of the chapter, followed by the "core" algorithms that form part of the preparation and feedback algorithms required for the implementation of the proposed approach using the RTI Scheme. The resulting algorithms were used for developing an auto-generation toolkit that implemented the proposed approach where its computational performance was able to be evaluated. The section ends with a generic computations analysis of the relevant algorithms of the proposed approach where significant computational benefits can be observed. Finally, the chapter includes 4 case studies presented in sections 5.3, 5.4, 5.5, and 5.6 which were used to demonstrate and discuss the relevant properties, alternatives, advantages and disadvantages of the proposed approach, as well as provide insight into potential applications where the proposed approach can be a viable alternative. The chapter concludes with a brief summary of the key ideas and contributions contained within.

Disclosure: The main contents and methodologies discussed in this chapter were published in a number of articles, including: an initial conference (abstract-only) paper in IFAC NMPC 2018 [48]; a journal paper in IET Control Theory and Applications 2020 [50]; and the IFAC World Congress 2020 conference paper [59] of the work done in collaboration with Ph.D. student Juan-Guerrero Fernandez. As a result, some of the content presented in this chapter will be inevitably repeated.

5.1 The Shifting Strategy

The proposed Shifting Strategy consists of 2, or optionally 3, main sub-strategies, namely:

1. Shifting Input Structure: It uses a time-varying input structure which is shifted in an *absolute time-frame* to maintain *consistency* whilst reducing the degrees of freedom of the optimisation.

2. Shifting State/Output Inequality Constraints: It focuses the efforts of the optimisation to a reduced number of states and outputs inequality constraints which are shifted in an *absolute time-frame* to maintain *consistency* whilst potentially reducing memory and computation times.

3. Shifting Shooting Points and Lagrange Multipliers (Optional): It penalises a reduced amount of output and/or state errors (hereafter referred as shooting points) which are shifted in an *absolute time-frame* to potentially reduce memory and computation times. Additionally, Lagrange Multipliers may also be shifted for hot-starting procedures of active set methods.

5.1.1 Consistency: The Foundation of the Shifting Strategy

One of the key concepts to understand this strategy is *consistency*. Although this concept may not be commonly used in the context of NMPC, it has been key for developing important methods and discussion around the general MPC literature. For example, the Initial Value Embedding (IVE) of the RTI Scheme presented in section 3.2, and indeed similar strategies such as Advanced Multi-Step [112], rely on shifting the nominal inputs and states, as well as the Lagrange Multipliers optimal guesses *consistently* along the prediction horizon. Similarly, nominal stability methods such as the infinite horizon costing approach presented in chapter 3 rely on *consistently* embedding a "complete/total" solution that the optimisation could follow, rather than just a part of it which can lead to ill-posed optimisations as discussed in [122] that suffer from bad performance and/or stability issues where the optimal solution at a given step differs from the next, ie. embedding an *inconsistent* overall "plan". An example of this was shown in the zero-terminal constraint approach of figure 3.3a. Another example can be found in [110] where "*consistency* constraints" are used to preserve recursive feasibility of the optimisation. This concept is also relevant for defining appropriate unbiased costs for achieving offset-free control as introduced in section 3.5 where it was discussed how biased cost functions have an inconsistent overall target where the penalised terms of the optimisation, eg. the inputs and output errors or targets, can be "fighting" each other which ultimately causes offsets in the closed loop response of the resulting control system. What is important to understand from this is that the mathematical tools of Optimal Control will always deliver what is embedded into them, whether that is good or bad performance, appropriate or inappropriate stability and recursive feasibility properties, as well as *consistent* or *inconsistent* objectives.

In the following subsections we will show how the principle of *consistent absolute time-frame shifting* will be the foundation of the proposed Shifting Strategy, and as long as this basic principle is applied correctly it can be extended to a wide range of alternatives whilst preserving nominal stability and recursive feasibility of the optimisation. It should be noted that each of the aforementioned sub-strategies can, in principle, be applied separately as long as the aforementioned principle is embedded.

5.1.2 Shifting Input Blocks: The Ideal Moving Window Blocking Approach

In this section we will introduce the shifting input blocks to obtain what the author of this thesis calls "The Ideal Moving Window Blocking (MWB)" approach which can be represented by a time-varying input-blocking structure. Although some of the ideas and methods introduced in this section are similar to the MWB approach presented in [18], there are several key differences which separate the proposed approach with the aforementioned work, particularly: definition 5.1 which determines the general input structure to be used; the use of what the author of this thesis calls "The Ideal Prediction Horizon", supported by theorem 5.1 which allows the prediction horizon and number of decision variables to be constant, rather than time-varying as in algorithm 1 of [18]; and the implementation of the overall approach in the context of the RTI NMPC framework which can be considered an extension of the latter. Finally, although the approach has some similarities with lifted systems [124], it is fundamentally different given both measurements and control actions are available at all times, and the methodology is applied with the objective of reducing the computation times of the optimisation, thus allowing faster corrections to disturbances/ uncertainties whilst preserving nominal stability and recursive feasibility.

Input-Blocking Structures

In order to understand the proposed time-varying input-blocking structure (5.11), we must first define the basic or "standard" input-blocking structure. Although there exist different ways of obtaining this, the main objective of this method is to reduce the degrees of freedom by embedding an input structure where the inputs or decision variables are blocked in sections that have the same value, eg. by using an equality constraint of the form $\hat{u}_k = \hat{u}_{k+1} = \cdots = \hat{u}_{k+N_b-1}$ spread evenly across all the prediction horizon where N_b is the number of inputs or decision variables having the same value, hereafter referred as the "block size". For a system with a prediction horizon that is an integer multiple of the block size, this can be achieved with an input parameterisation of the form ($\hat{U} = \mathbb{N}\mathbb{U}$ or $\delta\hat{U} = \mathbb{N}\delta\mathbb{U}$), as introduced in the chapter 4, particularly in equations (4.1a and 4.1b) with the input-structure defined as:

$$\mathbb{N} = \begin{bmatrix} \boldsymbol{n}_{N_b} & \mathbb{O}_{N_b} & \cdots & \mathbb{O}_{N_b} \\ \mathbb{O}_{N_b} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbb{O}_{N_b} \\ \mathbb{O}_{N_b} & \cdots & \mathbb{O}_{N_b} & \boldsymbol{n}_{N_b} \end{bmatrix} \tag{5.1}$$

where $\mathbb{N} \in \mathbb{R}^{N_p n_u \times N_{E_\mathbb{N}} n_u}$ is the input-blocking matrix, $N_{E_\mathbb{N}}$ is related to the size of the compressed Hessian as discussed in chapter 4, and where the inner matrices $\boldsymbol{n}_x$ and $\mathbb{O}_x$, represent matrices with x vertically arranged identity and zero matrices of size $n_u \times n_u$, respectively, ie. given by:

$$\boldsymbol{n}_x = \begin{bmatrix} I_1^{n_u \times n_u} & \cdots & I_x^{n_u \times n_u} \end{bmatrix}^T \quad \mathbb{O}_x = \begin{bmatrix} O_1^{n_u \times n_u} & \cdots & O_x^{n_u \times n_u} \end{bmatrix}^T \tag{5.2}$$

An alternative way of obtaining this input-parameterisation which is equivalent is to use the so called "input-increments" ($\Delta\hat{u}_k = \hat{u}_k - \hat{u}_{k-1}$) and embed a similar input-structure to that of (5.1) but with $\boldsymbol{n}_x$ having only the first identity, ie. given by:

$$\boldsymbol{n}_x = \begin{bmatrix} I_1^{n_u \times n_u} & O_2^{n_u \times n_u} & \cdots & O_x^{n_u \times n_u} \end{bmatrix}^T \tag{5.3}$$

This last approach, ie. using input-increments ($\Delta\hat{U}$) as decision variables, is how it was formulated in the IFAC World Congress 2020 conference paper [59], but it should not misunderstood as it achieves the exact same task of blocking the input, thus having the same solution. Moreover, note that input-structure (5.1) can also be embedded in the input increments which would result in a "ramp" type of block or section, similar to that of First Order Hold (FOH) models [109] which can be used to obtain smoother trajectories but may lack the rapid response available in standard input-blocking approaches.

This concept of blocking the inputs is commonly used in the standard Generalised Predictive Control (GPC) approach for the so called "control horizon" (N_c) [19, 122, 146] (although not commonly referred to as blocking) where the optimisation uses a reduced number of degrees of freedom congested at the beginning of the prediction horizon, and the rest of the $N_p - N_c$ control actions are blocked by embedding the equality $\hat{u}_{k+N_c-1} = \hat{u}_{k+N_c} = \cdots = \hat{u}_{k+N_p-1}$ which can be done by using a similar input-structure to that of (5.1). However, one of the main advantages of having decision variables spread over the prediction horizon rather than congested at the beginning as in GPC is that the system or optimisation itself may require control actions available in the future.

To illustrate the potential advantage of input-blocking over GPC, figure 5.1 presents a comparison of the initial open-loop predictions of the optimal trajectories for the Inverted Pendulum system presented in case study of section 5.5 when using 3 approaches, namely: Full Degrees of Freedom (FDoF) with a prediction horizon of $N_p = 52$ ($T_p = 1.3(s)$), depicted by the blue line; Input-Blocking with a block size of $N_b = 4$, ie. having $N_{E_{\mathbb{N}}} = \frac{N_p}{N_b} = 13$ degrees of freedom, depicted by the red dot-dashed line; and GPC with the same degrees of freedom, ie. $N_c = N_{E_{\mathbb{N}}}$, depicted by the green dashed line. Although all the optimal solutions present noticeable differences in the input trajectories, the predicted trajectories for the angle and position are nearly indistinguishable for the blocked and FDoF solutions, as seen from the inner graphs, whereas the GPC solution clearly results in a different trajectory. Similarly, the blocked input can be appreciated to follow the FDoF solution more closely than the GPC approach. As expected, the predicted costs of all the cases were: $J_{FDoF} = 1872$, $J_{blk} = 1878$ and $J_{GPC} = 1994$, which clearly shows the superiority of blocking over GPC for this particular system. Obviously the GPC solution would adjust the input as the horizon moves forward (receding horizon strategy), and might be able to perform similar in closed-loop. However, it is this *inconsistency*/ill-posedness within each prediction that may negatively affect the overall closed loop solution in the long term, especially when constraints come into play. A similar comparison can be found in figure 2 of [59].

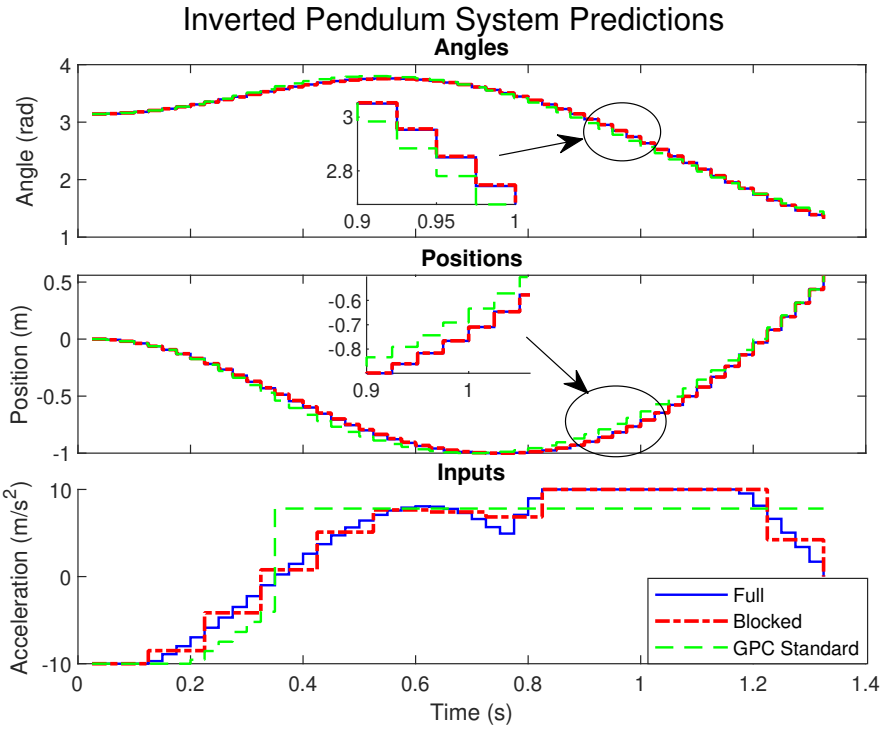


Figure 5.1: Blocking vs Standard GPC Comparison in the Inverted Pendulum System with $N_b = 4$

When To Use Blocking? A Boxing Analogy

Although input-blocking can generally present substantial advantages when compared to other strategies [59], it is not a generic solution that will always present the same advantages, irrespective of the system dynamics or requirements of the optimisation. This naturally brings up the question of *when to use blocking?*, and to answer this question we will use a "Boxing Analogy". Yes, the fighting sport!

As with many other sports, boxing requires having excellent timing and planning of movements such as punching, crouching and footwork, as well as fast reaction times and anticipation or prediction the opponent movements. In this case we can think of the solution of OCPs for Nonlinear Dynamical Systems based on NMPC as a fight between two boxers. On the one side we have a well trained World Champion professional boxer, representing the Optimal Control mathematical tools and methods, capable of executing virtually any technique or skill within its available set flawlessly whilst respecting its underlying constraints such as punching distance, speed of movements, muscle fatigue, etc., eg. representing actuator constraints. On the other hand, we have his opponent; whether that is a veteran professional boxer or a regular person, representing the performance objectives and dynamical systems to be handled by the optimisation which may include certain constraints, eg. not to be hit in certain areas, and will fight back, representing, for example, an unstable system not "giving in" to become stable, or having strong resistance against going to a desired target and presenting disturbances.

Clearly, if the opponent is a regular person, representing eg. a simple stabilisation problem of a linear first order system under nominal conditions, the World Champion boxer might as well beat its opponent with his eyes closed, ie. with virtually no effort, simply by planning and executing a couple of optimal movements. In contrast, if the system was the veteran professional boxer, the World Champion boxer might require significantly more effort including planning more movements, being faster, having faster reaction times and anticipating the opponent movements to make split-second decisions. Certainly, the World Champion could go "full-effort" on the regular person and "micro-manage" its movements and planning to completely destroy him, but it would most certainly be unnecessary. Thus, to answer the original question of when to use blocking, we pose the following questions: *How much effort do you want to put?*; *In how many moves can you beat your opponent?*; *How fast do you need to be to achieve your desired objective?*. The answers to these questions certainly depend on the fighting style that you want the system to have, or indeed that may be required; do you need the system to be like famous boxer Manny Paquiao and throw a large amount of rapid punches, or can it to be like famous boxer Mike Tyson and throw a reduced but mortal amount of punches?

The Time-Varying Shifting Input-Blocking Structure

Despite the standard input-blocking structure (5.1) generally being capable of following or "replicating" the "Full Degrees of Freedom" solution closely (as in figure 5.1), its direct implementation is known to produce significant recursive feasibility problems when considered in a receding horizon closed-loop framework [18, 130, 134]. This is because its repeated implementation prevents the optimisation from including the solution obtained in the previous time-step, ie. "the tail" of the solution [122], as a possible solution to the current optimisation. Instead, the optimisation makes a plan at every time-step which is then immediately forced to disregard at the next sampling time without the guarantee that will be able to find a new one. Thus, a basic requirement to solve this problem is to re-include the tail of the solution somehow which would result in a *consistent* optimisation where at each time-step, the latter is able to improve or "build on top" of the previous solution. To achieve this, we propose to fix the input-blocking structure in an "*absolute time-frame*" where the "breaking points" of the blocks are shifted in time to maintain consistency. This results in a set of blocking structures which if applied sequentially, guarantee that the tail is always included, thus solving the underlying problem.

Definition 5.1. *Shifted Input-Blocking: The Moving Window Blocking Approach*

The proposed shifted input-blocking structures which arguably can be seen as a variant of the MWB approach of [18], can be formally represented with the input equalities (5.4), defined for the entire prediction horizon (N_p), on the *absolute time-frame* steps $k+N_{b_0}$ $\forall N_{b_0} = [0, N_b-1]$, "restarting" at time step $k + N_b$ (hereafter referred as the "breaking point"), and repeating infinitely.

$$\hat{\mathbb{U}}_{n|k+N_{b_0}} = \hat{u}_{k+j+(n-1)N_b|k+N_{b_0}} \tag{5.4a}$$

$$\forall N_{b_0} = [0 \rightarrow N_b - 1] \tag{5.4b}$$

$$\forall n = [1 \rightarrow \left\lceil \frac{N_p - N_b + N_{b_0}}{N_b} \right\rceil + 1] \tag{5.4c}$$

$$\begin{cases} \forall j = [N_{b_0} \rightarrow N_b - 1] & \text{if} \quad n = 1 & \text{(First Block)} \\ \forall j = [0 \rightarrow N_b - 1] & \text{if} \quad 1 < n < \left\lceil \frac{N_p - N_b + N_{b_0}}{N_b} \right\rceil + 1 & \text{(Intermediate Blocks)} \\ \forall j = [0 \rightarrow j_{last}] & \text{if} \quad n = \left\lceil \frac{N_p - N_b + N_{b_0}}{N_b} \right\rceil + 1 & \text{(Last Block)} \end{cases} \tag{5.4d}$$

$$\text{with} \quad j_{last} = N_p + N_{b_0} - 1 - \left\lceil \frac{N_p - N_b + N_{b_0}}{N_b} \right\rceil N_b \tag{5.4e}$$

where $\hat{\mathbb{U}}_n$ is the n_{th} element of the total blocked input $\hat{\mathbb{U}}$ vector; N_{b_0} is a **"virtual block position indicator"** related to the time step at which the equalities are defined in the *absolute time frame*, eg. $N_{b_0} = 0$ being the equalities for initial time step k; n is related to n_{th} block section, eg. $n = 1$ being the first one; and j is related to the size of the n_{th} blocked section. Notice both the size j of the block and number of blocks n vary according to the time step N_{b_0} at which the equalities are defined.

In simple terms, this definition establishes the equalities that must be embedded at each time step $k + N_{b_0}$ on the range $N_{b_0} = [0, N_b - 1]$, ie. the N_b time steps where the initial block was considered. Once the time step reaches the "breaking point" ($N_{b_0} = N_b$), it "restarts" and continues to embed the corresponding equalities in the next block, repeating infinitely. An example of its application is provided next for clarity, and a figure showing its conceptual application is given in 5.2. □

Example 5.1. *Example usage of Definition (5.1)*

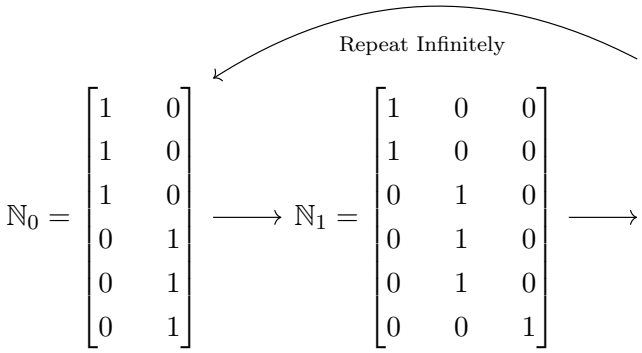
To clarify the utility of definition 5.1 and provide an example of its general usage, consider an optimisation of a single-input system ($n_u = 1$) with a prediction horizon of $N_p = 6$, and a block size of $N_b = 3$. The input equalities that result from using this definition are the following:

$$\left.\begin{aligned} \mathbb{U}_{1|k} = u_{k|k} = u_{k+1|k} = u_{k+2|k} \quad & \text{for } n = 1 \\ \mathbb{U}_{2|k} = u_{k+3|k} = u_{k+4|k} = u_{k+5|k} \quad & \text{for } n = 2 \end{aligned}\right\} \quad \textbf{For Time Step } N_{b_0} = 0 \tag{5.5a}$$

$$\left.\begin{aligned} \mathbb{U}_{1|k+1} = u_{k+1|k+1} = u_{k+2|k+1} \quad & \text{for } n = 1 \\ \mathbb{U}_{2|k+1} = u_{k+3|k+1} = u_{k+4|k+1} = u_{k+5|k+1} \quad & \text{for } n = 2 \\ \mathbb{U}_{3|k+1} = u_{k+6|k+1} \quad & \text{for } n = 3 \end{aligned}\right\} \quad \textbf{For Time Step } N_{b_0} = 1 \tag{5.5b}$$

$$\left.\begin{aligned} \mathbb{U}_{1|k+2} = u_{k+2|k+2} \quad & \text{for } n = 1 \\ \mathbb{U}_{2|k+2} = u_{k+3|k+2} = u_{k+4|k+2} = u_{k+5|k+2} \quad & \text{for } n = 2 \\ \mathbb{U}_{3|k+2} = u_{k+6|k+2} = u_{k+7|k+2} \quad & \text{for } n = 3 \end{aligned}\right\} \quad \textbf{For Time Step } N_{b_0} = 2 \tag{5.5c}$$

These equalities can be embedded into the optimisation with the following $N_b = 3$ matrices $(\mathbb{N}_0, \mathbb{N}_1, \mathbb{N}_2)$ applied sequentially $(\mathbb{N}_0 \rightarrow \mathbb{N}_1 \rightarrow \mathbb{N}_2)$, "resetting or restarting" after reaching the last one $(\mathbb{N}_2)$ and repeating infinitely, ie. by following $(\mathbb{N}_0 \rightarrow \mathbb{N}_1 \rightarrow \mathbb{N}_2 \rightarrow \mathbb{N}_0 \rightarrow \mathbb{N}_1 \rightarrow \mathbb{N}_2 \rightarrow \cdots \rightarrow \infty)$.



$$\mathbb{N}_0 = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \longrightarrow \mathbb{N}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \longrightarrow \mathbb{N}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.6}$$

Remark 5.1. *Equivalence of Relative and Non-Relative Shifted Input-Blocking Optimisation*

Note that when using this shifting input-blocking strategy and shifting the solution ($\bar{U}_{|k} = \hat{U}_{|k-1}$) along in time using the IVE strategy of the RTI Scheme, the decision variables will remain completely aligned with the nominal input $\bar{U}$ at all times (assuming $\bar{U}$ was initially aligned, eg. starting from a free response $\bar{U} = \mathbb{O}$), thus satisfying the "condition equality" (4.9) introduced in chapter 4 which consequently results in both types of optimisation (relative and non-relative) resulting in the exact same solution for this approach. This will be discussed in the inverted pendulum case study of section 5.5.

Remark 5.2. *Reduced Number of Input Constraints: Note that when using this type of approach, the number of input constraints can be reduced to the number of decision variables, eg. $U_{min} \leq \mathbb{U} \leq U_{max}$.*

The Ideal Prediction Horizon

One of the distinctive characteristics of the proposed Shifted Input-Blocking strategy is what the author of this thesis calls "The Ideal Prediction Horizon". This concept emerged after one of the reviewers from the IFAC 2018 conference (abstract only) paper [48] asked the question, "*How do you deal with block sizes that are not integers of the prediction horizon?*". Indeed, although it may seem intuitive to select the prediction horizon as a multiple integer of the block size, the reality is that this is not the best option. We have already seen an example of this in example 5.1, where the number of decision variables $(\mathbb{U}_n)$ related to the number of columns varied from $n = 2$ to $n = 3$. From a computational perspective, this is not a desirable behaviour given it could lead to requiring dynamic memory allocation, something which should be avoided at all cost when developing real-time auto-generated control system routines to improve computational performance.

Although one may find a simple solution to this such as using shrinking horizon strategies [28, 31] or developing a set of different auto-generated routines for each of the cases (eg. $n = 2, n = 3$ for example 5.1), a simpler solution is to use the proposed "Ideal Prediction Horizon" presented below.

Theorem 5.1. *The Ideal Prediction Horizon*

When using definition 5.1 to maintain recursive feasibility, the selected prediction horizon (N_p) must be an integer multiple of the block size (N_b) plus 1 to keep the Hessian dimension $N_{E_{\mathbb{N}}}$ (related to the number of decision variables) constant for any block size N_b, at all time-steps $N_{b_0} = [0 \rightarrow N_b - 1]$.

Proof. The expected size of the Hessian $N_{E_{\mathbb{N}}}$ for any block size (N_b) and prediction horizon (N_p) is given by:

$$N_{E_{\mathbb{N}}} = \left\lceil \frac{N_p - x}{N_b} \right\rceil + 1 \tag{5.7}$$

where x is the length of the first blocked input ($n = 1$ of equation 5.4) of a given blocking structure. In simple terms, this expression can be understood as the size of the first block (+1), plus the size of the rest of the blocks $\left(\left\lceil \frac{N_p - x}{N_b} \right\rceil\right)$.

Looking at the range of j for case $n = 1$ (first block) in equation (5.4d) of definition (5.1), it can be seen that the size of first block will shrink from N_b to 1. Therefore, to have a constant Hessian dimension, the following must hold:

$$\left\lceil \frac{N_p - N_b}{N_b} \right\rceil + 1 = \left\lceil \frac{N_p - x}{N_b} \right\rceil + 1 \quad \forall x = [1, N_b] \tag{5.8}$$

Equation (5.8) can only be satisfied for all x and any N_b by using $N_p = nN_b + 1$, where n is an integer number. Substituting this expression in (5.8) gives:

$$\left\lceil \frac{nN_b + 1 - N_b}{N_b} \right\rceil + 1 = \left\lceil \frac{nN_b + 1 - x}{N_b} \right\rceil + 1 \tag{5.9}$$

After some algebraic manipulation:

$$n - 1 + \left\lceil \frac{1}{N_b} \right\rceil = n + \left\lceil \frac{1 - x}{N_b} \right\rceil \tag{5.10}$$

because $\left\lceil \frac{1}{N_b} \right\rceil = 1$ and $\left\lceil \frac{1-x}{N_b} \right\rceil = 0 \ \forall x = [1 \to N_b]$, equation (5.10) holds. □

The Ideal Moving Window Blocking Input-Structure

Having defined the Ideal Prediction Horizon, it is evident that the resulting Ideal MWB structure ($\mathbb{N}_{N_{b_0}}$) for each "virtual" block position indicator N_{b_0} is given by:

$$\mathbb{N}_{N_{b_0}} = \begin{bmatrix} \boldsymbol{n}_{N_b - N_{b_0}} & \mathbb{O}_{N_b - N_{b_0}} & \cdots & \cdots & \mathbb{O}_{N_b - N_{b_0}} \\ \mathbb{O}_{N_b} & \boldsymbol{n}_{N_b} & \mathbb{O}_{N_b} & \cdots & \cdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \cdots & \cdots & \mathbb{O}_{N_b} & \boldsymbol{n}_{N_b} & \mathbb{O}_{N_b} \\ \mathbb{O}_{1+N_{b_0}} & \cdots & \cdots & \mathbb{O}_{1+N_{b_0}} & \boldsymbol{n}_{1+N_{b_0}} \end{bmatrix} \quad \forall N_{b_0} = [0, N_b - 1] \tag{5.11}$$

where the matrices $\boldsymbol{n}_x$ and $\mathbb{O}_x$ are defined as before, representing matrices with x vertically arranged identity and zeros matrices of size $n_u \times n_u$, respectively, ie. given by:

$$\boldsymbol{n}_x = \begin{bmatrix} I_1^{n_u \times n_u} \\ \vdots \\ I_x^{n_u \times n_u} \end{bmatrix} \quad \mathbb{O}_x = \begin{bmatrix} O_1^{n_u \times n_u} \\ \vdots \\ O_x^{n_u \times n_u} \end{bmatrix} \tag{5.12}$$

From the ideal MWB input structure (5.11) it can be appreciated how the initial block (ie. the one in the left upper corner $\boldsymbol{n}_{N_b-N_{b_0}}$) is shrinking its size from N_b to 1, whereas the last block (ie. the one in the lower right corner $\boldsymbol{n}_{1+N_{b_0}}$) is expanding its size from 1 to N_b. In contrast, the inner block sizes ($\boldsymbol{n}_{N_b}$) maintain the same size throughout the entire shifting process. This can be visualised clearly in figure 5.2 where an example ideal blocking structure is given for a block size of $N_b = 10$ shifting from $N_{b_0} = 0$, depicted by the thick blue line, to $N_{b_0} = 9$, depicted by the red dot-dashed line.

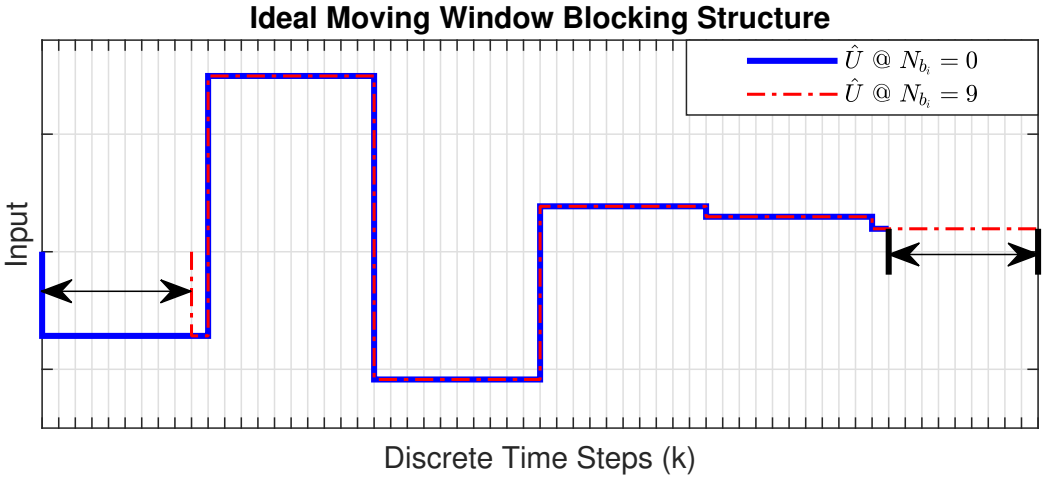


Figure 5.2: Example of Ideal Moving Window Blocking Parameterisation with $N_b = 10$

Example 5.2. *Challenging the Intuitive Block Shifting*

At this point, one might intuitively be asking the question "*Why not just expand the final block beyond the N_b block size to keep the sizes constant instead of using the proposed Ideal Prediction Horizon?*". Indeed, this is how it was done in the IFAC 2020 conference paper [59] (see equation 19b of the paper), not to distract from the main message as the IET 2020 paper [50] containing the Ideal Prediction Horizon proof hadn't been publish by then. Although one might get away with this intuitive way of shifting the blocks, the following numeric example aims at demonstrating how easily this type of strategy can lead to instability.

Consider the unconstrained optimal control problem of replicating a given reference input sequence $U_r = [u_{r_k}, u_{r_{k+1}}, \cdots, u_{r_{k+N_p-1}}]$ using the blocked input $\mathbb{N}_{N_{b_0}}\hat{\mathbb{U}}$. Indeed, we must not forget that one of the purposes of the input-parameterisation framework is to replicate the original input sequence.

This optimisation is then simply given by:

$$J = \frac{1}{2}(U_r - \mathbb{N}_{N_{b_0}}\hat{\mathbb{U}})^T(U_r - \mathbb{N}_{N_{b_0}}\hat{\mathbb{U}}) \tag{5.13}$$

Or written in the standard form:

$$J = \frac{1}{2}\hat{\mathbb{U}}^T \underbrace{(\mathbb{N}_{N_{b_0}}^T \mathbb{N}_{N_{b_0}})}_{E_{\mathbb{N}}} \hat{\mathbb{U}} + \hat{\mathbb{U}}^T \underbrace{(-\mathbb{N}_{N_{b_0}}^T U_r)}_{f_{\mathbb{N}}} \tag{5.14}$$

As it has already been proven in earlier chapters, the unconstrained solution to this problem is given by:

$$\mathbb{U} = -E_{\mathbb{N}}^{-1} f_{\mathbb{N}} = (\mathbb{N}_{N_{b_0}}^T \mathbb{N}_{N_{b_0}})^{-1} \mathbb{N}_{N_{b_0}}^T U_r \tag{5.15}$$

To provide an easily replicable result, consider now a single-input system ($n_u = 1$) with a Non-Ideal Prediction Horizon of $N_p = 4$, and a block size of $N_b = 2$. In simple terms, we want to replicate the first 4 inputs of the sequence using only 2 blocked inputs. To compare this approach with the Ideal Prediction Horizon, consider the next closest ideal horizon of ($N_p = 5$) resulting in the ideal and non-ideal blocking structures given by:

Solution Type	Input-Structure	
Non-Ideal	$\mathbb{N}_0 = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$	$\mathbb{N}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$
Ideal	$\mathbb{N}_0 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\mathbb{N}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$

(5.16)

As it can be seen, both input structures maintain the same number of decision variables from one step to the next. This is because the non-ideal input structure $\mathbb{N}_1$ expands the last block size beyond $N_b = 2$ as it can been from the arrangement of the three vertical ones.

Assume that the total input reference trajectory to be replicated is given by,

$$U_r = [\underbrace{0.2259, 0.1707, 0.2277, 0.4357}_{\text{Initial Horizon}}, 0, \cdots, 0]^T \tag{5.17}$$

Notice the inputs beyond the "initial horizon" ($N_p = 4$) are zero, thus satisfy the common requirement for stability where costs beyond the horizon are zero.

We can then select the following two consecutive "shifted" input references (U_r from 5.18) as they would appear when implemented in a receding horizon fashion, with the additional zeros enclosed by the red boxes representing the input references specifically required by the "Ideal Prediction Horizon" approach which has $N_p = 5$ instead of $N_p = 4$.

$$\underbrace{U_{r|k} = \begin{bmatrix} 0.2259 \\ 0.1707 \\ 0.2277 \\ 0.4357 \\ \boxed{0} \end{bmatrix}}_{\text{Input Reference @ k}} \rightarrow \underbrace{U_{r|k+1} = \begin{bmatrix} 0.1707 \\ 0.2277 \\ 0.4357 \\ 0 \\ \boxed{0} \end{bmatrix}}_{\text{Shifted Input Reference}} \tag{5.18}$$

The solutions for these optimisation problems using both approaches (ideal and non-ideal) at the two initial consecutive steps are given by:

Time Step	Solution Type	Solution		Cost
k	Non-Ideal	$\hat{\mathbb{U}}_k = \begin{bmatrix} 0.1983 \\ 0.3317 \end{bmatrix}$	$\rightarrow$	$J_k = 0.0232$
	Ideal	$\hat{\mathbb{U}}_k = \begin{bmatrix} 0.1983 \\ 0.3317 \\ 0 \end{bmatrix}$	$\rightarrow$	$J_k = 0.0232$
$k+1$	Non-Ideal	$\hat{\mathbb{U}}_{k+1} = \begin{bmatrix} 0.1707 \\ 0.2211 \end{bmatrix}$	$\rightarrow$	$\underbrace{\boxed{J_{k+1} = 0.0950}}_{\text{Problem !}}$
	Ideal	$\hat{\mathbb{U}}_{k+1} = \begin{bmatrix} 0.1707 \\ 0.3317 \\ 0 \end{bmatrix}$	$\rightarrow$	$J_{k+1} = 0.0216$

(5.19)

As it can be seen in (5.19), the solution of both ideal and non-ideal approaches at the initial time step (k) results in identical solutions and costs ($J_k = 0.0232$) with the "additional" variable of the ideal-horizon approach doing nothing ($\mathbb{U}_3 = 0$) as the target for that step is the added 0 of the red box in (5.18). However, the solution of the non-ideal approach in the following step ($k+1$) results in an increase to the cost of $J_{k+1} = 0.0950 > 0.0232 = J_k$, as signaled by the red box, which clearly demonstrates how easily this type of solution can lead to instability. In contrast, the solution of the ideal approach results in a decrease to the cost of $J_{k+1} = 0.0216 < 0.0232 = J_k$, thus satisfying the common nominal stability requirements.

The core message of this example is that by using this small modification of the "Ideal Prediction Horizon", the resulting solution will always outperform the intuitive blocking approach which provides a clear motivation to use it.

Example 5.3. *The Lock-In Condition*

In addition to the potential recursive feasibility problems that arise from not using the proposed shifting strategy, an interesting problem that can happen when embedding a non-shifting blocking structure using the Relative framework presented in chapter 4 is what the author of this thesis calls, *The Lock-In Condition*. This problem can be very easily explained with a simple example.

Consider a single-input system ($n_u = 1$) with the input-structure $\mathbb{N}$ of equation (5.20), representing the shortest possible ideal horizon ($N_p = 3$) with the smallest (non-unity) block size ($N_b = 2$) under the Relative framework, ie. given by:

$$\underbrace{\begin{bmatrix} \hat{u}_0 \\ \hat{u}_1 \\ \hat{u}_2 \end{bmatrix}}_{\hat{U}} = \underbrace{\begin{bmatrix} \bar{u}_0 \\ \bar{u}_1 \\ \bar{u}_2 \end{bmatrix}}_{\bar{U}} + \underbrace{\begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}}_{\mathbb{N}} \underbrace{\begin{bmatrix} \delta\hat{\mathbb{U}}_0 \\ \delta\hat{\mathbb{U}}_1 \end{bmatrix}}_{\delta\hat{\mathbb{U}}} \tag{5.20}$$

Moreover, consider that this input structure is used to optimise a random convex quadratic cost (its actual definition is irrelevant), subject to the inequality:

$$U_{min} - \bar{U} \leq \mathbb{N}\delta\hat{\mathbb{U}} \leq U_{max} - \bar{U} \tag{5.21}$$

Let us assume that the optimisation starts from the "free response", ie. the initial nominal input is zero ($\bar{U} = \mathbb{0}$), and for some reason, the optimisation results in the blocked decision variable $\delta\hat{\mathbb{U}} = [u_{max}, u_{min}]^T$, thus making the input trajectory reach both max-min input limits in the form of:

$$\begin{bmatrix} \hat{u}_0 \\ \hat{u}_1 \\ \hat{u}_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_{max} \\ u_{min} \end{bmatrix} = \begin{bmatrix} u_{max} \\ u_{max} \\ u_{min} \end{bmatrix} \tag{5.22}$$

After shifting the input trajectory as done by the IVE strategy of the RTI Scheme discussed in chapter 3, the resulting nominal input is given by:

$$\bar{U} = \begin{bmatrix} \bar{u}_0 \\ \bar{u}_1 \\ \bar{u}_2 \end{bmatrix} = \begin{bmatrix} u_{max} \\ u_{min} \\ u_{min} \end{bmatrix} \tag{5.23}$$

This causes the inequality (5.21) at the next optimisation to reach a "lock-in condition" that results from attempting to solve the inequalities related to the first two inputs, given by:

$$u_{min} - u_{max} = u_{min} - \bar{u}_0 \leq \delta\hat{\mathbb{U}}_0 \leq u_{max} - \bar{u}_0 = u_{max} - u_{max} = \boxed{0} \tag{5.24a}$$

$$\boxed{0} = u_{min} - u_{min} = u_{min} - \bar{u}_1 \leq \delta\hat{\mathbb{U}}_0 \leq u_{max} - \bar{u}_1 = u_{max} - u_{min} \tag{5.24b}$$

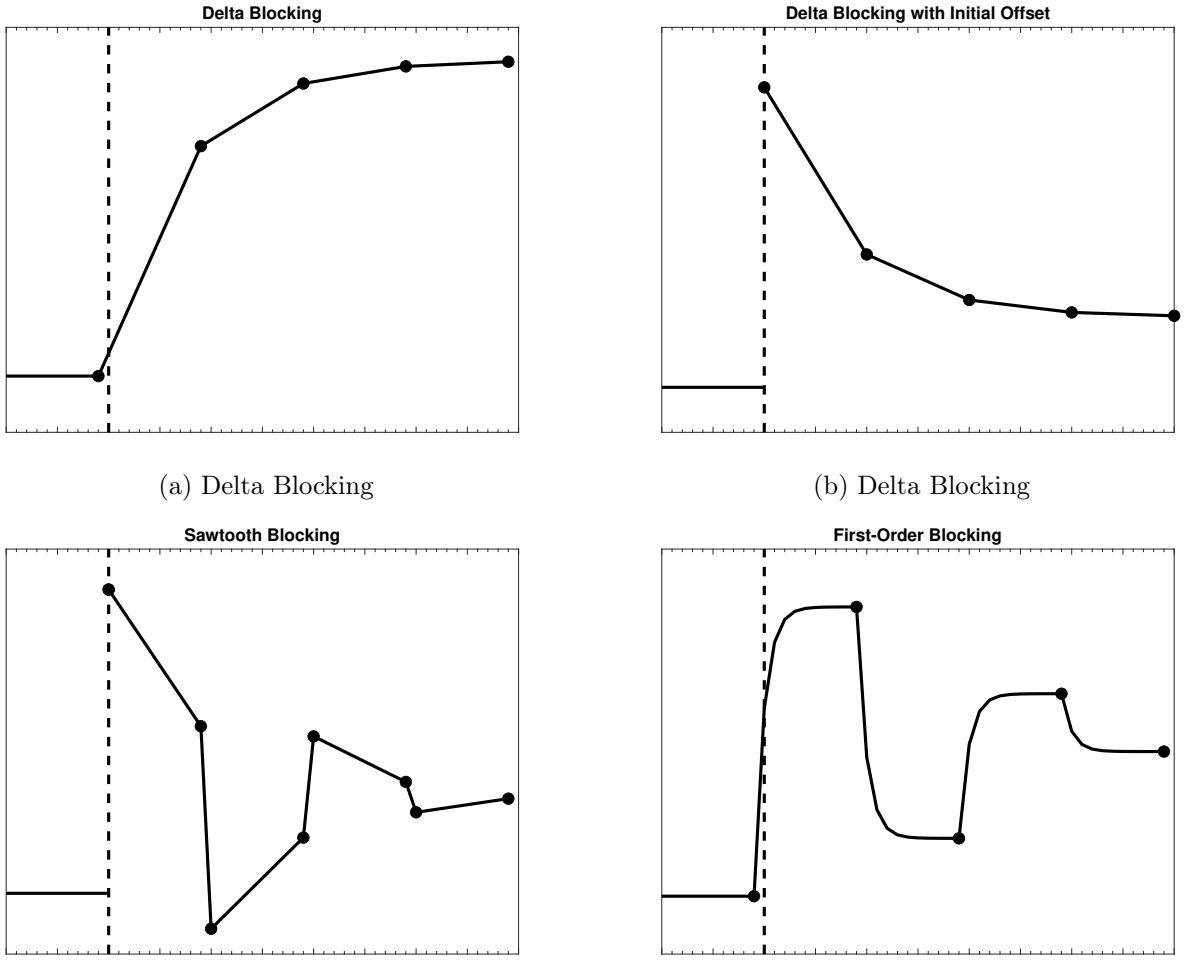
for which the only solution is $\delta\hat{\mathbb{U}}_0 = 0$, ie. not being able to move the first blocked input at all.

This causes the optimisation to lose the first decision variable completely, consequently "locking-in" the solution at whatever value was initially selected (in this case u_{max}) until the entire block passes, essentially resulting in an "open-loop (no feedback at all)" solution. It's worth mentioning that this phenomena could equally happen on blocked inputs at later stages, ie. is not restricted to happen only in the first decision variable.

By not being able to move, the optimisation is then forced to "ignore" new information that may be coming as the horizon moves forward, such as new costs or new small constraint violations and disturbances the may require small optimal corrections which otherwise could easily lead to infeasible problems. Obviously, this could have a more significant impact if larger block sizes were used, although a simple way to avoid this problem would be to use the Non-Relative framework. Indeed, this resulting in-feasibility problem is stressed out significantly in the Case Study of the Inverted Pendulum presented in section 5.5, particularly in table 5.10 where the number of in-feasibilities when using the Relative Non-Shifted framework are significantly higher than when using the Non-Relative Non-Shifted approach. Nonetheless, by using the proposed shifting-strategy, the input-structure keeps the decision variables aligned with the nominal input at all times after shifting it along with the IVE strategy, as discussed in remark 5.1, thus completely avoiding this problem.

Alternative Blocking Structures

Although the main focus of this chapter has been in blocked input structures of the form given in figure 5.2, the core methodology of fixating the input-structure in an *absolute-time frame* can be extended to other input structures such as the ones provided in figure 5.3. Each of these input trajectories may be achieved through different input structures, and may present different advantages/disadvantages as well as a different "Ideal Prediction Horizons". Ultimately, all of this strategies preserve the main advantages of allowing reduced number of degrees of freedom and reduced number of input inequality constraints. A brief discussion on each of the alternatives is provided below.



(a) Delta Blocking

(b) Delta Blocking

(c) Sawtooth Blocking

(d) First-Order Blocking

Figure 5.3: Input-Blocking Structure Alternatives

1. Delta Blocking (figure 5.3a): This type of input-parameterisation may be ideal for following smooth trajectories and can be easily achieved by embedding the ideal MWB structure proposed in this chapter (ie. equation 5.11) to the input increments, ie. $\Delta\hat{U} = \mathbb{N}_{N_{b_0}}\Delta\hat{\mathbb{U}}$. As a result, this approach would have the same Ideal Prediction Horizon. Moreover, assuming the previous input is inside the bounds ($u_{min} \leq u_{k-1} \leq u_{max}$), the entire input trajectory can be constrained by looking only at the end points of each segment. This alternative approach was used for the obstacle avoidance case study of section 5.6.

2. Delta Blocking with Initial Offset (figure 5.3b): This type of input parameterisation can be achieved by augmenting the Delta Blocking of the previous approach with one additional decision variable for the initial offset of the form:

$$\Delta\hat{U} = \begin{bmatrix} I^{n_u \times n_u} & \mathbb{O} \\ \mathbb{O} & \mathbb{N}_{N_{b_0}} \end{bmatrix} \Delta\hat{\mathbb{U}} \tag{5.25}$$

   Thus, the Ideal Prediction Horizon in this case would be a multiple integer of the block size plus 2 (ie. $N_p = nN_b + 2$) due to the extra input required for the initial offset. This initial offset would allow the system to have a fast initial input change (possibly required to quickly cancel disturbances), followed by a smooth trajectory. As with the Delta Blocking approach, the entire input trajectory can be constrained by looking only at the end points of each segment, with the only difference being the added constraint on the initial offset.

3. Sawtooth Blocking (figure 5.3c): This type of input parameterisation can also be embedded using the input increments ($\Delta\hat{U} = \mathbb{N}_{N_{b_0}}\Delta\hat{\mathbb{U}}$) with slightly different input-structure, and may offer greater responsiveness and flexibility to the user at the cost of having two decision variables per block, resulting in more degrees of freedoms and required input constraints. Because of this, the Ideal Prediction Horizon for this type of input trajectory is (given without proof) $N_p = 2nN_b+2$. An example of the resulting structures for $N_b = 3$ using an Ideal Prediction Horizon $N_p = 8$ would be given by:

$$\mathbb{N}_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow \mathbb{N}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow \mathbb{N}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.26}$$

   The entire input trajectory can be constrained by considering the first and last point of each segment, which again is twice the amount of the other approaches.

4. First Order Blocking (figure 5.3d): This type of input is perhaps one of the most natural to apply and can be achieved simply by augmenting the system with an "inner" input state that has first-order dynamics. The method would then consider the "outer" input as the decision variable for the optimisation on which the proposed Ideal MWB structure of equation (5.11) would be embedded. Once the optimisation is solved, the "inner" input state is selected as the actual input to be applied. As with the Delta Blocking approach, the entire input trajectory can be constrained by looking only at the end point.

It is important to understand that some of this structures might be better suited than others for a given application, and the purpose of introducing them is not to say which one is better but rather to generate awareness on alternatives in which the proposed strategy of this chapter can be applied.

Ultimately, all of these structures can use the core methodology of fixating the input-structure in an *absolute time-frame* to recover recursive feasibility, which overall enlarges the scope of this chapter's contribution.

5.1.3 Shifting Inequality Constraints, Shooting Points and Lagrange Multipliers: The Extended Approach

Having observed the benefits of the proposed *absolute time-frame* shifting strategy applied to the standard input-blocking approach, it was considered relevant to extend its application to the states and outputs of the system. In this case, it was observed that it was particularly useful to implement the approach in the state and output inequality constraints of the optimisation, as they represent one of the main "resource-hungry" parts when it comes to computation times of general Quadratic Programs. We have already seen an example of this approach in the first-order input-blocking alternative presented in figure 5.3d where the entire inner input/state trajectory can be constrained simply by constraining only the end point of each block. Moreover, it is well known that the number of inequality constraints that can be active at a given time will be restricted by the number of decision variables [146]. Thus, it was considered logical to also look for a way of reducing the state and output constraints as even if they were required to be active, they would be restricted by the amount of available decision variables. Additionally, this can also have an impact on the memory requirements of the constraint matrices and vectors of the optimisation which may be relevant for real-time control systems hardware which typically has reduced memory resources.

The key question to ask here is whether it is actually required to include constraints at every sampling time from a practical perspective. *What will happen if the system presents a small violation in an intermediate time-step?* Or more importantly, *what CAN happen? Is it possible that the system is restricted by its inherent dynamics as in the first order input-blocking example (fig. 5.3d)?* And also, *is it more important to have the constraints concentrated at the beginning of the horizon, eg. as in the input parameterisation of GPC, or spread around the horizon?* In all likelihood, it is quite often found that the most relevant constraints are those closest to the beginning of the horizon, but *what if they are not, and constraints at the end of the horizon are as important as the ones in the middle or in the beginning?. What if in order to be able to satisfy the constraints, they must be anticipated with a significant time ahead?* Certainly including the constraints across the entire horizon would be ideal if there were sufficient computational resources, but *what can we do when there are not enough?*

This question remains! And an example of it will be given in the obstacle avoidance case study 5.6.

The purpose of the proposed sub-strategy is therefore to provide viable alternatives for these scenarios which would allow the user to select a reduced amount of inequality constraints spread over the horizon whilst preserving nominal stability and recursive feasibility of the optimisation by using the proposed *absolute-time frame* shifting strategy. Ultimately, all discrete MPC controllers ignore, to some extent, the intermediate points, ie. those which aren't seen between discrete sampling times, thus the approach is somewhat justified from that perspective. In addition to this, it is well known that output hard-constraints generally cannot be guaranteed feasible, thus ignoring intermediate constraints allows the user to "virtually relax" the optimisation, although other approaches such as soft-constraints can also be used. This will be seen in the results of the obstacle avoidance case study of section 5.6.

On the other hand, the absolute time-frame shifting strategy can also be applied to the shooting points themselves, ie. the state and output errors of the optimisation, which can ultimately reduce computation times and memory requirements further. However, it is important to keep in mind that reducing shooting points could lead to aliasing problems where the optimisation might be ignoring intermediate information containing important frequency content. Thus, this sub-strategy is considered an optional feature.

Finally, the proposed shifting strategy can also be applied for shifting the Lagrange Multipliers of the optimisation, particularly when used for hot-starting active-set methods as in the RTI Scheme. However, as this may not always be required, eg. if Interior Point methods are used, it is also considered an optional feature.

Definition 5.2. *Shifting Shooting Points and Inequality Constraints*

The proposed approach can be formally represented by selecting only the state (and/or output) prediction errors ($e_k = x_{r_k} - x_k$) and inequality constraints at the end of each block of definition (5.1) to be included in the optimisation, ie. only including those given by:

$$\hat{e}_{k+nN_b|k+N_{b_0}} = x_{r_{k+nN_b|k+N_{b_0}}} - \hat{x}_{k+nN_b|k+N_{b_0}} \tag{5.27a}$$

$$x_{min} \leq \hat{x}_{k+nN_b|k+N_{b_0}} \leq x_{max} \tag{5.27b}$$

$$\forall N_{b_0} = [0, N_b - 1]$$

$$\forall n = [1, N_{E_{\mathbb{N}}} - 1]$$

where N_{b_0} is the "virtual block position indicator" related to the time steps in the *absolute time-frame* as in definition 5.1; n is related to number of shooting points, and the last point of the prediction horizon:

$$\hat{e}_{k+N_p+N_{b_0}|k+N_{b_0}} = x_{r_{k+N_p+N_{b_0}|k+N_{b_0}}} - \hat{x}_{k+N_p+N_{b_0}|k+N_{b_0}} \tag{5.28a}$$

$$x_{min} \leq \hat{x}_{k+N_p+N_{b_0}|k+N_{b_0}} \leq x_{max} \tag{5.28b}$$

is always included, representing the shooting point/inequality constraint related to the last block ($n = N_{E_{\mathbb{N}}}$).

To select the points at the end of each blocked input, the time step N_{b_0} must be "in phase" with the time step N_{b_0} used by Definition 5.1. As with definition 5.1, once the time step N_{b_0} reaches the "breaking point", it restarts the sequence, repeating infinitely ($N_{b_0} = 1 \rightarrow N_b \rightarrow 1 \rightarrow N_b \rightarrow \infty$).

Remark 5.3. *Shooting Point Selection*

Selecting a given subset of state errors or state constraints can be achieved by selecting (or computing) only the respective rows of matrices $H, G, D, X_r, \bar{X}, M, \gamma$.

Remark 5.4. *Constant Number of Inequalities and Shooting Points*

As there is only one selected shooting point and inequality constraint per block, it can be proved there will be a constant number of shooting points and inequality constraints when using the Ideal Prediction Horizon presented in theorem (5.1). This is given without proof as it is self-evident.

To provide a visual example of definition 5.2, figure 5.4 shows the application of this approach to the function ($y(k) = A\sin(\alpha k + \phi)\exp(-\tau k)$ - a trajectory commonly found in the response of dynamical systems) when using a block size of $N_b = 5$. In this figure, the shifting shooting points and inequality constraints to be included in the optimisation are represented by the large points, whereas the small points represent intermediate points that form part of the available predictions at discrete time-steps but are ignored to save computational resources. Moreover, as the selected points are aligned with the end of each block of definition (5.1), these points are essentially shifting backwards in the relative time-frame (ie. as the horizon moves forward), due to first block shrinking. This can be seen with the red arrows executing the "shifting action", moving the entire black curve 4 steps back to the blue curve, where the large blue points represent the shifted shooting points, ie. the same points of the black curve @ time-step $k+4$ - the edge of the block from definition 5.1. An important thing to notice here is that the last point of the black curve (signaled by the circle at the end of the trajectory) remains at the relative time-step N_p as required by definition 5.2. Finally, as pointed out earlier, given the strategy ignores the intermediate points, some of them can undergo slight constraint violations as signaled for the 9^{th} step ahead of the black curve (5^{th} of the blue curve). In practice, these small violations can be compensated using conservative safety limits, eg. by embedding a small gap or slack between the real hard constraint and the constraint used for the optimisation. This will be discussed further in the inverted pendulum case study of section 5.5, and the obstacle avoidance case study of section 5.6.

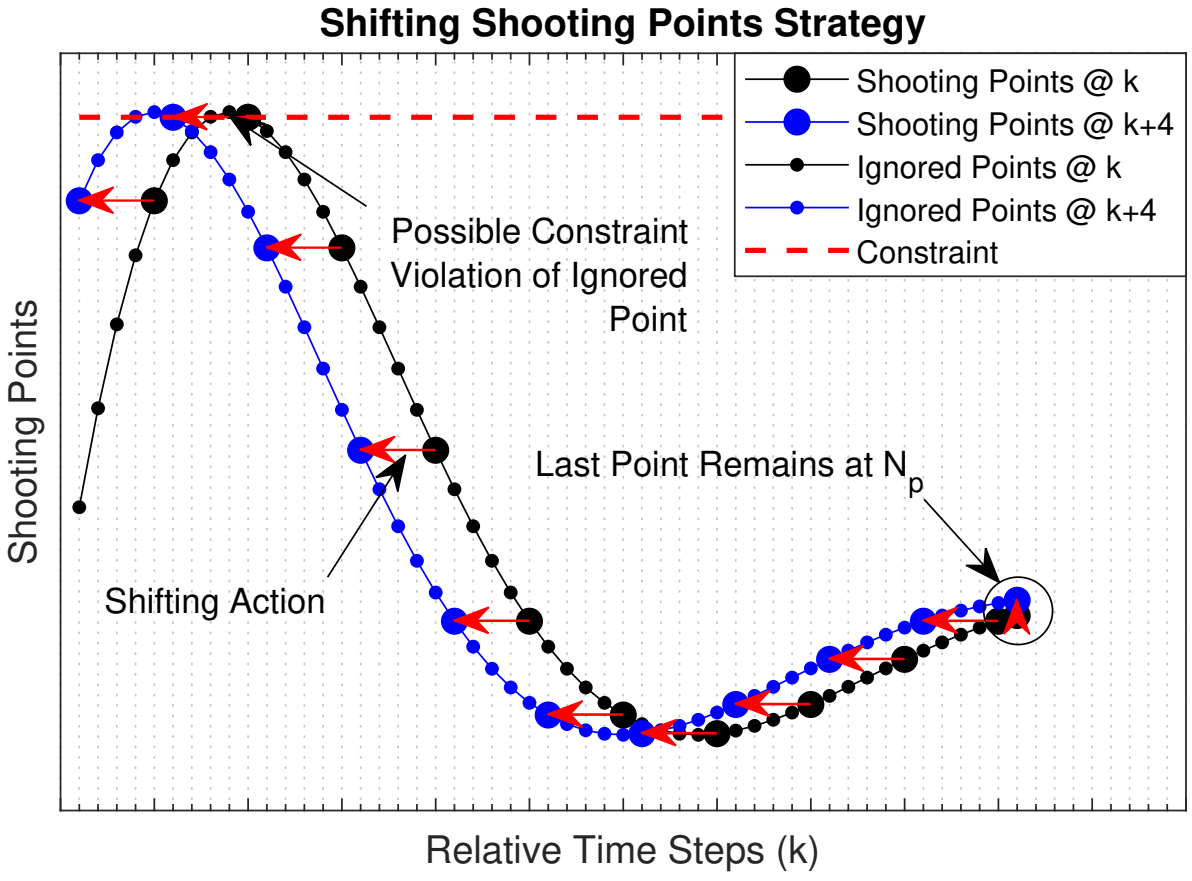


Figure 5.4: Shifting Points Strategy Example with $N_b = 5$ in a Relative Time-Frame

Alternative Shifting of Shooting Points/Inequality Constraints

Although the proposed shifting strategy focuses specifically on placing the shooting points and inequality constraints at the end of each block, one can find alternatives for the implementation of this where as long as the variables in question are fixed in an *absolute time-frame*, one will be able to find nominal stability and recursive feasibility properties similar to the ones provided in section 5.1.5. This can ultimately extend the scope of the proposed approach presented in this chapter.

To illustrate possible extensions of the proposed approach, consider first an optimisation with a block size $N_b = 4$, and an Ideal Prediction Horizon of $N_p = 9$ which instead of including only the last shooting point at each block it includes the last two shooting points of each block resulting in a $0-0-1-1$ pattern, with the last point of the prediction horizon always included. This pattern is illustrated in table 5.1 where each cyan "one" box represent a shooting point to be included in the optimisation, expressed in the relative time-frame. The limits of each block at each of the time-steps (N_{b_0}) are signaled by the red boxes for clarity. From this table it can be appreciated that the number of points in this optimisation remains constant (5 points) at all the virtual block position indexes (N_{b_0}), thus avoiding dynamic memory allocation requirements whilst reducing the overall shooting points and inequality constraints of the original optimisation by approximately half its original size. In this scenario, the approach would allow violations at a maximum of 2 consecutive points. However, the approach could be equally be implemented as a $0-1-0-1$ pattern, ignoring a maximum of 1 point.

N_{b_0}/k	1	2	3	4	5	6	7	8	9
0	0	0	1	1	0	0	1	1	1
1	0	1	1	0	0	1	1	0	1
2	1	1	0	0	1	1	0	0	1
3	1	0	0	1	1	0	0	1	1

Table 5.1: Example of Alternative Shifting for Inequality Constraints/Shooting Points with an Ideal Prediction Horizon $N_p = 9$ and block size $N_b = 4$, expressed in a Relative Time-Frame

Note that this same type of philosophy can be applied to virtually any pattern. As a second example, consider an optimisation with a block size of $N_b = 5$ and an Ideal Horizon of $N_p = 11$ implementing a $0-1-0-1-1$ pattern, with the last point of the horizon always included. The pattern is illustrated in table 5.2 where the number of points can also be seen to remain constant (7 selected points). In this case, the approach would reduce the number of points by approximately 37%, whilst ignoring a maximum of 1 consecutive shooting point.

N_{b_0}/k	1	2	3	4	5	6	7	8	9	10	11
0	0	1	0	1	1	0	1	0	1	1	1
1	1	0	1	1	0	1	0	1	1	0	1
2	0	1	1	0	1	0	1	1	0	1	1
3	1	1	0	1	0	1	1	0	1	0	1
4	1	0	1	0	1	1	0	1	0	1	1

Table 5.2: Example of Alternative Shifting for Inequality Constraints/Shooting Points with an Ideal Prediction Horizon $N_p = 11$ and block size $N_b = 5$, expressed in a Relative Time-Frame

Shifting Lagrange Multipliers

Another key part of the proposed approach focuses on how to handle the Lagrange Multipliers (λ) of the optimisation, particularly when using them for hot-starting active-set based Quadratic Programs such as QP OASES. The important thing to notice here is that because the approach uses a reduced amount of constraints (and consequently Lagrange Multipliers) spread around the horizon, the best guess for the active-set in the next iteration is not given simply by shifting them at every time-step, as implemented in the standard RTI IVE strategy. Instead we propose the Lagrange Multipliers must be shifted in the *absolute time-frame* along with the proposed input-blocking and shifting inequality constraints of definitions 5.1 and 5.2, respectively. This is addressed in the theorem 5.2.

Theorem 5.2. *Shifting Lagrange Multipliers*

When using definitions 5.1 and 5.2 with the Ideal Prediction Horizon of theorem 5.1, the Lagrange Multipliers must be shifted ONLY when the optimisation reaches the "breaking point", ie. when resetting the virtual block position indicator (N_{b_0}), to provide the guess of the active-set in the following step.

Proof. Consider the following unblocked Lagrange Multipliers related to the positive input constraints ($\mathbb{N}\hat{\mathbb{U}} \leq U_{max}$):

$$\lambda = [\lambda_{k|k}^T, \lambda_{k+1|k}^T, \cdots, \lambda_{k+N_p-1|k}^T]^T \tag{5.29}$$

The usual shifting strategy used by the RTI is:

$$\begin{aligned} &\lambda_{k+i|k+i} > 0 \quad \text{if} \quad \lambda_{k+i|k+i-1} > 0 \quad \text{(active if previous was active)} \\ &\lambda_{k+i|k+i} = 0 \quad \text{if} \quad \lambda_{k+i|k+i-1} = 0 \quad \text{(inactive if previous was in inactive)} \end{aligned} \tag{5.30}$$

Now, for simplicity, consider an ideal prediction horizon $N_p = N_b + 1$. When using the proposed shifting input structure (5.11) that results from definition (5.1), the entire trajectory can be constrained simply by constraining $\hat{\mathbb{U}} = [\hat{\mathbb{U}}_1^T, \hat{\mathbb{U}}_2^T]^T < U_{max}$ which results in requiring only 2 Lagrange Multipliers $[\lambda_{1|k+j}, \lambda_{2|k+j}]$ for the two blocked sections where:

$$\lambda_{1|k+j} = \lambda_{k+i|k+j} = \lambda_{k+N_b-1|k+j} \qquad \forall j = [0, N_b - 1] \qquad \forall i = [j, N_b - 1] \tag{5.31}$$

for the first blocked section, and:

$$\lambda_{2|k+j} = \lambda_{k+N_b+i|k+j} = \lambda_{k+N_b-1|k+j} \qquad \forall j = [0, N_b - 1] \qquad \forall i = [0, j] \tag{5.32}$$

for the second blocked section.

By applying the standard RTI IVE shifting (5.30) combined with equalities (5.31) and (5.32), and considering the "breaking point" of definition (5.1) happens at $k + N_b$ gives:

$$\begin{aligned} &\lambda_{1|k+N_b} > 0 \quad \text{active if} \quad \lambda_{2|k+N_b-1} > 0 \\ &\lambda_{1|k+N_b} = 0 \quad \text{inactive if} \quad \lambda_{2|k+N_b-1} = 0 \end{aligned} \tag{5.33}$$

which in simple terms means, the first block would be active if the second block was active before the breaking point.

The same would hold for if more blocks were considered, resulting in:

$$\begin{array}{lll} \lambda_{i|k+N_b} > 0 & \text{active if} & \lambda_{i+1|k+N_b-1} > 0 \\ \lambda_{i|k+N_b} = 0 & \text{inactive if} & \lambda_{i+1|k+N_b-1} = 0 \end{array} \qquad \forall i = [1, N_{E_{\mathbb{N}}} - 1] \tag{5.34}$$

Moreover, a similar procedure can be followed to verify that the Lagrange Multipliers of the state inequality constraints must also be shifted in time with the breaking points. Finally, a similar approach can be derived based on the same principle of *consistent absolute time-frame* shifting for shifting the Lagrange multipliers when Alternative Shifting of Shooting Points/Inequality Constraints such as those introduced in tables 5.1 and 5.1 were to be used. □

An example comparison of the proposed Lagrange Multipliers Shifting strategy is given in case study 5.5, particularly in the computation time comparison of table 5.12 where the computation times obtained by the QP OASES solver are seen to be higher when the proposed approach is not used.

5.1.4 General Optimisation Framework

The application of any of the sub-strategies proposed in this chapter leads to a general optimisation framework where a set of N_b Quadratic Programs ($J^{[N_{b_0}]}\ \forall N_{b_0} = [0, N_b - 1]$), need to be formulated and solved sequentially, whilst restarting every time the "breaking point" ($N_{b_0} = N_b$) is reached, and repeating infinitely. As an example, the resulting Quadratic Programs when applying the Ideal MWB input structure (5.11) to the relative input-parameterised framework (4.4) are given by:

$$J^{[N_{b_0}]} = \frac{1}{2}\delta\hat{\mathbb{U}}^T E_{\mathbb{N}}^{[N_{b_0}]}\delta\hat{\mathbb{U}} + \delta\hat{\mathbb{U}} f_{\mathbb{N}}^{[N_{b_0}]} \tag{5.35a}$$

$$M_{\mathbb{N}}^{[N_{b_0}]}\delta\hat{\mathbb{U}} \leq \gamma \tag{5.35b}$$

$$E_{\mathbb{N}}^{[N_{b_0}]} = \mathbb{N}_{N_{b_0}}^T (H^T Q H + R)\mathbb{N}_{N_{b_0}} \tag{5.35c}$$

$$f_{\mathbb{N}}^{[N_{b_0}]} = -\mathbb{N}_{N_{b_0}}^T [H^T Q(X_r - \bar{X} - D - G\delta x_0) - R(\bar{U} - U_r)] \tag{5.35d}$$

$$M_{\mathbb{N}}^{[N_{b_0}]} = \begin{bmatrix} \mathbb{N}_{N_{b_0}} \\ -\mathbb{N}_{N_{b_0}} \\ H\mathbb{N}_{N_{b_0}} \\ -H\mathbb{N}_{N_{b_0}} \end{bmatrix} \qquad \gamma = \begin{bmatrix} U_{max} - \bar{U} \\ \bar{U} - U_{min} \\ X_{max} - \bar{X} - D - G\delta x_0 \\ \bar{X} + D + G\delta x_0 - X_{min} \end{bmatrix} \tag{5.35e}$$

A similar formulation can be found in the IFAC World Congress 2020 conference paper [59] (see equation 20 of the paper). Moreover, note that the proposed approach can be also applied to Linear MPC which would result in a set of N_b Quadratic Programs that can be pre-prepared and stored offline, eg. that of the compressed set of Hessians ($E_{\mathbb{N}}^{[N_{b_0}]}$), compressed set of constraint matrices ($M_{\mathbb{N}}^{[N_{b_0}]}$), as well as certain parts of the compressed set of linear terms ($f_{\mathbb{N}}^{[N_{b_0}]}$). This is a particularly simplified task when the approach uses the non-relative prediction models (3.26) for standard linear MPC discussed in theorem 3.1, which can simplify the pre-storage of linear terms ($f_{\mathbb{N}}^{[N_{b_0}]}$) and constraint vector (γ). However, we do point out that other approaches such as Explicit MPC might be more suitable for linear MPC implementation [37], and the proposed approach is given as an alternative that could be more relevant for improving the performance of NMPC instead.

On the other hand, we re-emphasise that the number of constraints required to implement the proposed approach can be significantly reduced. For example, given the nominal input ($\bar{U}$) remains aligned with the Ideal MWB input structure (5.11) when using the IVE of the RTI Scheme (as discussed in remark 5.1), the approach can constrain the entire input trajectory of (5.35) by looking at only one of the inputs of each block, eg. the last one as in the shifting inequality constraint approach, resulting in an input inequality constraint of the form $(U_{min}-\bar{U})_{selected} \leq \delta\hat{\mathbb{U}} \leq (U_{max}-\bar{U})_{selected}$ as discussed in remark 5.2. Similarly, the approach would select a reduced amount of shifting state constraints which would modify the required state inequality constraint to the form $(X_{min} - \bar{X} - D - G\delta x_0)_{selected} \leq (H\mathbb{N}_{N_{b_0}})_{selected} \leq (X_{max} - \bar{X} - D - G\delta x_0)_{selected}$. The application of both of this strategies combined would result in a modified set of constraint matrices ($M_{\mathbb{N}}^{[N_{b_0}]}$) and constraint vectors ($\gamma^{[N_{b_0}]}$) given by:

$$M_{\mathbb{N}}^{[N_{b_0}]} = \begin{bmatrix} I^{N_{E_{\mathbb{N}}}n_u \times N_{E_{\mathbb{N}}}n_u} \\ -I^{N_{E_{\mathbb{N}}}n_u \times N_{E_{\mathbb{N}}}n_u} \\ (H\mathbb{N}_{N_{b_0}})_{selected} \\ -(H\mathbb{N}_{N_{b_0}})_{selected} \end{bmatrix} \qquad \gamma_{\mathbb{N}}^{[N_{b_0}]} = \begin{bmatrix} (U_{max} - \bar{U})_{selected} \\ -(U_{min} - \bar{U})_{selected} \\ (X_{max} - \bar{X} - D - G\delta x_0)_{selected} \\ -(X_{min} - \bar{X} - D - G\delta x_0)_{selected} \end{bmatrix} \tag{5.36}$$

Once again, we reiterate that the selection of each of the points can be done simply by selecting (or computing) the relevant rows of matrices $H, G, D, X_r, \bar{X}, M, \gamma$ as discussed in remark 5.3.

To provide a more complete example of the proposed approach, consider an optimisation with a block size of $N_b = 2$, and an Ideal Prediction Horizon of $N_p = 5$. By implementing the shifted input-blocking, inequality constraints, and state-errors strategies, the approach would solve the set of Quadratic Programs ($J^{[0]}$ and $J^{[1]}$) defined in table 5.3, sequentially and repeating infinitely ($J^{[0]} \rightarrow J^{[1]} \rightarrow J^{[0]} \rightarrow \infty$), where the "selected" state-errors, input-structures, as well as input and state constraints to be included in the optimisation are presented, expressed in the Relative Time-Frame.

QPs ($J^{[N_{b_0}]}$)	$J^{[0]}$	$J^{[1]}$
Input-Errors (Included in Full)	$\hat{U} - U_r$	$\hat{U} - U_r$
State-Errors	$X_r - \hat{X} = \begin{bmatrix} x_{r_{k+2}} - \hat{x}_{k+2} \\ x_{r_{k+4}} - \hat{x}_{k+4} \\ x_{r_{k+5}} - \hat{x}_{k+5} \end{bmatrix}$	$X_r - \hat{X} = \begin{bmatrix} x_{r_{k+1}} - \hat{x}_{k+1} \\ x_{r_{k+3}} - \hat{x}_{k+3} \\ x_{r_{k+5}} - \hat{x}_{k+5} \end{bmatrix}$
Input Structure	$\mathbb{N}_0 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\mathbb{N}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$
Input Constraints	$U_{min} \leq \begin{bmatrix} \hat{u}_{k+1} \\ \hat{u}_{k+3} \\ \hat{u}_{k+4} \end{bmatrix} \leq U_{max}$	$U_{min} \leq \begin{bmatrix} \hat{u}_{k} \\ \hat{u}_{k+2} \\ \hat{u}_{k+4} \end{bmatrix} \leq U_{max}$
State Constraints	$X_{min} \leq \begin{bmatrix} \hat{x}_{k+2} \\ \hat{x}_{k+4} \\ \hat{x}_{k+5} \end{bmatrix} \leq X_{max}$	$X_{min} \leq \begin{bmatrix} \hat{x}_{k+1} \\ \hat{x}_{k+3} \\ \hat{x}_{k+5} \end{bmatrix} \leq X_{max}$

Table 5.3: Example of Overall Shifting Strategy applied to optimisation with block size of $N_b = 2$ and an Ideal Prediction Horizon of $N_p = 5$, expressed in the Relative Time-Frame

5.1.5 Stability and Recursive Feasibility

Up until this point we have discussed important details regarding the general nature as well as the possible alternatives of the proposed sub-strategies, without dealing with their essential nominal stability and recursive feasibility properties. However, we have emphasised repeatedly that the proper use of a *consistent absolute time-frame* shifting allows the "tail" (ie. the solution at the previous time-step) to be included as a possible solution to the current optimisation. This particular condition is the key to retain the aforementioned properties given it allows us to obtain guarantees for them under the infinite horizon and zero-terminal constraint assumptions discussed in chapter 3, in a relatively straight forward manner. The proof of this is addressed in theorem 5.3.

Theorem 5.3. *Stability and Recursive Feasibility of the Shifting Strategy*

The tail of the optimisation is automatically included when using definitions 5.1 and 5.2 with the Ideal Prediction Horizon 5.1 which allows the proposed approach to guarantee nominal stability and recursive feasibility for the infinite horizon and zero terminal constraint approaches.

Proof. Let us begin by observing that because the input-blocking structure of definition 5.1 with the Ideal Prediction Horizon of theorem 5.1 is fixed in an *absolute time-frame*, any optimal sequence of blocked inputs ($\hat{\mathbb{U}}^*$) at any time step ($k + N_{b_0}\ \forall N_{b_0} = [0, N_b - 1]$) can be replicated in the following time step $k + N_{b_0} + 1$. In simple terms, this means that equality (5.37) can always be satisfied.

$$\hat{\mathbb{U}}_{|k+N_{b_0}} = \begin{bmatrix} \hat{\mathbb{U}}_{1|k+N_{b_0}} \\ \hat{\mathbb{U}}_{2|k+N_{b_0}} \\ \vdots \\ \hat{\mathbb{U}}_{(N_{E_{\mathbb{N}}}-1)|k+N_{b_0}} \\ \hat{\mathbb{U}}_{(N_{E_{\mathbb{N}}})|k+N_{b_0}} \end{bmatrix} = \begin{bmatrix} \hat{\mathbb{U}}^*_{1|k+N_{b_0}-1} \\ \hat{\mathbb{U}}^*_{2|k+N_{b_0}-1} \\ \vdots \\ \hat{\mathbb{U}}^*_{(N_{E_{\mathbb{N}}}-1)|k+N_{b_0}-1} \\ \hat{\mathbb{U}}^*_{(N_{E_{\mathbb{N}}})|k+N_{b_0}-1} \end{bmatrix} = \hat{\mathbb{U}}^*_{|k+N_{b_0}-1} \quad \forall N_{b_0} = [1, N_{b_0} - 1] \tag{5.37}$$

When the solution reaches the "breaking point" ($N_{b_0} = N_b$), the first optimal blocked input ($\hat{\mathbb{U}}^*_{1|k+N_{b_0}-1}$) would "shrink" completely, thus requiring the entire blocked trajectory to be shifted, eg. the second block becoming the first one. This condition, represented by equality (5.38), can also be satisfied noting that the time step $k + N_b$ would represent the point at which N_{b_0} resets to 0, thus giving the initial blocking structure. In this case, a completely "new" blocked input would emerge (as signaled by the red box in 5.38), becoming a "free" element that would not require any constraint imposed on it.

$$\hat{\mathbb{U}}_{|k+N_b} = \begin{bmatrix} \hat{\mathbb{U}}_{1|k+N_b} \\ \hat{\mathbb{U}}_{2|k+N_b} \\ \vdots \\ \hat{\mathbb{U}}_{(N_{E_{\mathbb{N}}}-1)|k+N_b} \\ \boxed{\hat{\mathbb{U}}_{(N_{E_{\mathbb{N}}})|k+N_b}} \end{bmatrix} = \begin{bmatrix} \hat{\mathbb{U}}^*_{2|k+N_b-1} \\ \hat{\mathbb{U}}^*_{3|k+N_b-1} \\ \vdots \\ \hat{\mathbb{U}}^*_{(N_{E_{\mathbb{N}}})|k+N_b-1} \\ \boxed{\text{Free}} \end{bmatrix} = \hat{\mathbb{U}}^*_{|k+N_b-1} \tag{5.38}$$

This allows the optimisation to include the tail of the input solution which is known to be a key requirement to guarantee nominal stability and recursive feasibility. Moreover, note that this would also hold for any of the proposed alternatives of figure 5.3, as long as they are properly shifted in time.

Without loss of generality, consider now a regulation problem ($X_r = U_r = \mathbb{O}$) with a globally converged optimal set of selected state-inequality constraints ($X_{min} \leq (\hat{X}^*_{|k+N_{b_0}})_{selected} \leq X_{max}$) given by definition 5.2 at time steps $k + N_{b_0}$ $\forall N_{b_0} = [0, N_b - 1]$. The proposed approach fixes these points in an *absolute time frame*, with the only "moving" point being the last one as seen from equation (5.28a). To guarantee nominal stability and recursive feasibility between any two consecutive time-steps, the optimisation would be required to have equality (5.39) as a possible solution to the optimisation.

$$(\hat{X}_{|k+N_{b_0}})_{sel} = \begin{bmatrix} \hat{x}_{k+N_b|k+N_{b_0}} \\ \hat{x}_{k+2N_b|k+N_{b_0}} \\ \vdots \\ \hat{x}_{k+(N_{E_{\mathbb{N}}}-1)N_b|k+N_{b_0}} \\ \boxed{\hat{x}_{k+N_p+N_{b_0}|k+N_{b_0}}} \end{bmatrix} = \begin{bmatrix} \hat{x}^*_{k+N_b|k+N_{b_0}-1} \\ \hat{x}^*_{k+2N_b|k+N_{b_0}-1} \\ \vdots \\ \hat{x}^*_{k+(N_{E_{\mathbb{N}}}-1)N_b|k+N_{b_0}-1} \\ \boxed{\hat{x}^*_{k+N_p+N_{b_0}-1|k+N_{b_0}-1}} \end{bmatrix} = (\hat{X}^*_{|k+N_{b_0}-1})_{sel} \qquad (5.39)$$

$$\forall N_{b_0} = [1, N_b - 1]$$

As it can be seen, the only potential "problem" in satisfying this equality is that of the last selected points ($\hat{x}_{k+N_p+N_{b_0}|k+N_{b_0}}$ and $\hat{x}^*_{k+N_p+N_{b_0}-1|k+N_{b_0}-1}$), signaled by the red boxes. In the case where a sufficiently long (or infinite) prediction horizon is used and recursive feasibility of the input trajectory is guaranteed (as discussed previously), the last states and inputs of the optimisation would have negligible cost ($\hat{x}^*_{k+N_p+N_{b_0}-1|k+N_{b_0}-1} \approx 0$ and $u^*_{k+N_p+N_{b_0}-2|k+N_{b_0}-1} \approx 0$), thus allowing the optimisation to follow the plan in the previous time step whilst satisfying all the equalities, consequently resulting in nominal stability and recursive feasibility guarantees. It is noted, however, that given infinite horizons are not generally feasible to be solved in real-time, a more rigorous guarantee for this approach requires the use of invariant sets and terminal modes, such as the infinite horizon costing approach presented in chapter 3. This will be the topic of chapter 7 where the infinite horizon costing of the proposed approach will be derived, and the ability to obtain invariant sets will be discussed.

On the other hand, we can also use the zero-terminal constraint ($\hat{x}^*_{k+N_p+N_{b_0}-1|k+N_{b_0}-1} = 0$) approach presented in chapter 3 to prove the desired nominal stability and recursive feasibility properties. Strictly speaking, when implementing both main approaches: the shifting inequality constraints approach with the proposed Ideal MWB structure (5.11), the optimisation would be able to satisfy the requested equality (red boxes in 5.39) under a zero-terminal constraint assumption, if and only if, the last optimal blocked input obtained in the previous optimisation was also zero ($\mathbb{U}^*_{N_{E_{\mathbb{N}}}|k+N_{b_0}-1} = 0$), considering the typical assumption that the system remains at the origin if its in the origin ($0 = f(0, 0)$).

To prove this, consider an optimal feasible sequence that results from an optimisation with a block size N_b and an Ideal Prediction Horizon $N_p = N_b + 1$ at the initial time-step ($N_{b_0} = 0$), with the zero-terminal condition embedded (signaled by the red box in 5.40a), ie. given by:

$$\begin{bmatrix} \hat{x}^*_{k+1|k} & \hat{u}^*_{k|k} & \cdots & \cdots & \hat{x}^*_{k+N_b|k} & \hat{u}^*_{k+N_b-1|k} & \boxed{\hat{x}^*_{k+N_p|k} = 0} & \hat{u}^*_{k+N_p-1|k} \end{bmatrix} \qquad (5.40a)$$

$$s.t. \qquad \hat{\mathbb{U}}^*_{1|k} = \hat{u}^*_{k|k} = \cdots = \hat{u}^*_{k+N_b-1|k} \qquad (5.40b)$$

$$\hat{\mathbb{U}}^*_{2|k} = \hat{u}^*_{k+N_p-1|k} \qquad (5.40c)$$

The only optimal solution with guaranteed feasibility for the zero-terminal constraint at the following time-step $\hat{x}_{k+N_p+1|k+1} = 0$ is the tail of the solution, ie. the optimal inputs from (5.40a) at time steps $k+1 \rightarrow k+N_p-1$, with the last input being zero ($\hat{u}_{k+N_p|k+1} = 0$). Note that this is not possible given the equality $\hat{\mathbb{U}}_{2|k+1} = \hat{u}_{k+N_p-1|k+1} = \hat{u}_{k+N_p|k+1}$ would be embedded. The only time-step (N_{b_0}) at which the last input could take the required zero value would be at the "breaking point" ($N_{b_0} = N_b$), as discussed for the "free" value of equation (5.38).

Hence, guaranteeing nominal stability under the zero-terminal framework can be done by embedding the zero-terminal constraint on the last blocked input as well. However, its practical implementation would most certainly be a limitation as it essentially removes the last decision variable entirely, restricting its value for the entire sequence until reaching the "breaking point", as discussed in the *lock-in condition* example 5.3, thus preventing the solution from improving that particular block as the horizon moves forward. A possible alternative to this would be to add an additional non-blocked input at the end of the overall input-structure with its respective state inequality, similar to the Delta-Blocking with Initial Offset approach of equation (5.25), resulting in an input structure of the form:

$$\hat{U} = \begin{bmatrix} \mathbb{N}_{N_{b_0}} & \mathbb{O} \\ \mathbb{O} & I^{n_u \times n_u} \end{bmatrix} \hat{\mathbb{U}} \tag{5.41}$$

This relatively small modification would give the optimisation the flexibility it needs on the last input value to be able to assign a zero value if it needs to satisfy the constraint (5.39) but have the freedom to choose otherwise, whilst embedding the proposed Ideal MWB structure in the rest of terms and guaranteeing nominal stability and recursive feasibility for this scenario.

What this proves is that nominal stability under the zero-terminal constraint scenario CAN be obtained with very small modifications of the proposed methodologies. Nevertheless, the infinite horizon costing approach (which will be the focus of chapter 7) is preferred as zero-terminal constraint approaches are known to be rather strict without necessarily resulting in good closed-loop performance as discussed in chapter 3, particularly due to the ill-posedness/inconsistency that they can present, as seen in the example comparison of figure 3.3a. Regardless, the proposed approach presented excellent results even without embedding strict nominal stability or recursive feasibility guarantees which in some cases can prove to be unnecessary from a practical perspective (eg. see comparison of figure 3.3).

As a side note, its worth mentioning that, like in the input-blocking case, when the solution reaches the "breaking point" ($N_{b_0} = N_b$), the optimisation would have "gone through" the first selected state ($\hat{x}^*_{k+N_b|k+N_{b_0}}$), thus requiring the entire trajectory of selected points to be shifted, ie. second point becoming first and so on, fundamentally changing the equality that needs to be satisfied as in (5.38), but overall maintaining the feasibility of the selected points. This particular feature or "mechanism" was the foundation of the proposed Shifting Lagrange Multipliers strategy of theorem 5.2.

Finally, a similar procedure can be used to prove the nominal stability and recursive feasibility of the proposed (optional) shifting shooting points approach where it can be derived that under zero-terminal constraint conditions, the approach would reduce the optimal cost by AT LEAST $J_{k+1} \leq J_k - {u^*_{k|k}}^T R u^*_{k|k}$ whenever the time step is not at a breaking point ($N_{b_0} \leq N_b$), and give the standard decrease of AT LEAST $J_{k+1} \leq J_k - {x^*_{k+1|k}}^T Q x^*_{k+1|k} - {u^*_{k|k}}^T R u^*_{k|k}$ when at the breaking point.

This is given without proof not to distract from the main methodologies. □

Remark 5.5. *Alternative Stability and Recursive Feasibility Proofs*

We reiterate that the fundamental understanding of the proposed absolute time-frame shifting strategy would allow the derivation of similar nominal stability and recursive feasibility proofs of a wide range of alternatives, such as those described in figure 5.3, and tables 5.1 and 5.2, independently of whether the proposed sub-strategies are applied in combination or not.

5.2 Algorithm Details and Auto-generation

Having defined the overall theory and methodologies that support that proposed Shifting Strategy, we can now proceed to introduce the key algorithms that are required for its efficient implementation using the RTI NMPC framework. Thus, this section will present a set of algorithms and methods that were used for developing an auto-generation toolkit that allowed us to benchmark the proposed approach against the solution obtained by the ACADO toolkit. The section focuses specifically on the main two methodologies: the Ideal MWB, and the Shifting Constraints approach, with the understanding the similar procedures can be followed to derive the required algorithms for any of the alternatives discussed in the previous section.

The section is organised as follows: Given the inherent complexity of the overall approach, subsection 5.2.1 introduces a general method for handling the Shifting Strategy from an algorithmic perspective which was used for developing all the algorithms presented. Moreover, subsection 5.2.2 discusses how the proposed Ideal MWB approach can exploit the sparsity of matrix (5.11) to obtain the compressed prediction matrix ($H_{\mathbb{N}}$) whilst reducing both computation times and memory usage. Furthermore, subsection 5.2.3 includes an extension to the $O(N)$ and $O(N^2)$ algorithms from PhD thesis [8] presented in the re-derivation of chapter 3, subsection 3.3.1 which is considered one of the key contributions of this thesis. Additionally, subsection 5.2.4 presents 3 "core" algorithms that were used for developing the preparation and feedback phases algorithms of the RTI Scheme introduced in subsection 5.2.5. The section concludes with a generic computation analysis in subsection 5.2.6 comparing the proposed algorithms with the standard approach of chapter 3.

5.2.1 General Method for Handling the Shifting Strategy

The overall strategy requires us to keep track of all the moving parts of the optimisation, such as the limits and/or positions of each block and the selected points for the shifting strategy along the prediction horizon. Although one can use the expressions of the definitions 5.1 and 5.2, as well as the Ideal MWB structure (5.11) in some fashion, this proved to be quite impractical from an algorithmic perspective. Thus, a method for handling the proposed shifting strategy was developed.

The proposed method relies on using an "inner block position index" (N_{b_i}), and a block counter (n), for the algorithm to keep track of the block's position and number as it moves along the prediction horizon. The idea is to initialise the index (N_{b_i}) with the current "virtual block position indicator" (N_{b_0}) and iterate it forward for all the prediction horizon ($k = 1 \rightarrow N_p$) steps, resetting at the block limits $N_{b_i} = [N_{b_0} \rightarrow N_b - 1, 0 \rightarrow, N_b - 1, \cdots, N_{b_0}]$. Moreover, every time the block reaches the breaking point ($N_{b_i} = N_b$), it increases the block counter ($n = n + 1$), and resets the inner block position index value to the "zero" block position ($N_{b_i} = 0$), meaning a new block "emerged" along the prediction.

To illustrate this method, consider a system with a block size of $N_b = 4$, and an ideal prediction horizon of $N_p = 13$. Table 5.4 presents an example of all the possible sequences of N_{b_i} and n for all the initial blocks (N_{b_0}). To provide a visual aid, each coloured cell represent a different block, each of which relates to a different block number (n) and can be seen to be moving backwards (diagonally) as the initial block position moves forward $N_{b_0} = [0, \rightarrow N_b - 1]$. The points at which the inner block position index (N_{b_i}) "reset" to indicate a new block "emerged" along the prediction are signaled in red boxes, found at the start of every block. An important thing to notice is that the ending value of the inner block index (N_{b_i}) will always be the same as the initial value as signaled by the black circles (or ellipses), found at the start and end of each N_{b_i} sequence. This will be key for developing the $O(N)$ and $O(N^2)$ algorithms (5.1 and 5.2) as they must be run backwards.

N_{b_0}	k	1	2	3	4	5	6	7	8	9	10	11	12	13
0	N_{b_i}	(0)	1	2	3	[0]	1	2	3	[0]	1	2	3	(0)
	n	1	1	1	1	2	2	2	2	3	3	3	3	4
1	N_{b_i}	(1)	2	3	[0]	1	2	3	[0]	1	2	3	[0]	(1)
	n	1	1	1	2	2	2	2	3	3	3	3	4	4
2	N_{b_i}	(2)	3	[0]	1	2	3	[0]	1	2	3	[0]	1	(2)
	n	1	1	2	2	2	2	3	3	3	3	4	4	4
3	N_{b_i}	(3)	[0]	1	2	3	[0]	1	2	3	[0]	1	2	(3)
	n	1	2	2	2	2	3	3	3	3	4	4	4	4

Table 5.4: Example of all possible sequences of inner block index (N_{b_i}), and block counters (n), for all initial blocks positions $N_{b_0} = [0 \rightarrow 3]$ with a prediction horizon of $N_p = 13$ with a block size $N_b = 4$.

The algorithms will then make decisions based on the values of n and N_{b_i} using "if-then-else" conditionals. As an example, all last points of each block segment of table 5.4, ie. the constraints that would be selected for definition 5.2, can be found by the conditional: if ($N_{b_i} = N_b - 1$ OR $k = N_p$), then "select", else "ignore". Similarly, the initial point of each block segment can be found by the conditional: if ($N_{b_i} = 0$ OR $k = 1$) then, "initial point of block". On the other hand, the algorithm uses the block counter (n) to define the relevant limits of the required for-loops. This overall method can be seen explicitly in algorithms (5.1, 5.2, 5.3, 5.4 and 5.5) at various lines.

Remark 5.6. *Range of N_{b_0} and N_{b_i}*

The actual range of N_{b_0} and N_{b_i} that was implemented for the developed algorithms was $N_{b_0} = [1, N_b]$, meaning it resets to 1 whenever $N_{b_i} > N_b \rightarrow N_{b_i} = 1$. The range is equivalent, ie. having segments of size N_b in the absolute time-frame, but the required conditionals change slightly.

5.2.2 The Compressed Ideal Moving Window Blocking Prediction Matrix $H_{\mathbb{N}}$

In order to obtain an efficient implementation of the proposed approach, we must first understand the fundamental structures of all the relevant parts. One of its key parts is the compressed prediction matrix ($H_{\mathbb{N}} = H\mathbb{N}_{N_{b_0}}$) which can then be used to compute the compressed Hessian ($E_{\mathbb{N}}$), compressed linear term ($f_{\mathbb{N}}$), and compressed constraints matrix ($M_{\mathbb{N}}$), as discussed in chapter 4. It is straight forward to show that embedding the Ideal MWB matrix (5.11) results in a time-varying structure where the number of non-zero elements of each column varies according to the time-step (N_{b_0}) by:

$$H_{\mathbb{N}} = \underbrace{\begin{bmatrix} h_{1,1} & \mathbb{O} & \mathbb{O} & \\ h_{1,1} & h_{2,2} & \mathbb{O} & \\ \vdots & \ddots & \ddots & \mathbb{O} \\ h_{N_p,1} & h_{N_p,2} & \cdots & h_{N_p,1} \end{bmatrix}}_{H} \underbrace{\begin{bmatrix} \boldsymbol{n}_{N_b-N_{b_0}} & \mathbb{O}_{N_b-N_{b_0}} & \cdots & \cdots & \mathbb{O}_{N_b-N_{b_0}} \\ \mathbb{O}_{N_b} & \boldsymbol{n}_{N_b} & \mathbb{O}_{N_b} & \cdots & \cdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \cdots & \cdots & \mathbb{O}_{N_b} & \boldsymbol{n}_{N_b} & \mathbb{O}_{N_b} \\ \mathbb{O}_{1+N_{b_0}} & \cdots & \cdots & \mathbb{O}_{1+N_{b_0}} & \boldsymbol{n}_{1+N_{b_0}} \end{bmatrix}}_{\mathbb{N}_{N_{b_0}}}$$

$$= \begin{bmatrix} (h_{\mathbb{N}})_{1,1} & \mathbb{O} & \cdots & \mathbb{O} \\ \vdots & \vdots & \cdots & \vdots \\ (h_{\mathbb{N}})_{N_b-N_{b_0},1} & \mathbb{O} & \cdots & \mathbb{O} \\ (h_{\mathbb{N}})_{N_b-N_{b_0}+1,1} & (h_{\mathbb{N}})_{N_b-N_{b_0}+1,2} & \mathbb{O} & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ (h_{\mathbb{N}})_{2N_b-N_{b_0},1} & (h_{\mathbb{N}})_{2N_b-N_{b_0},2} & \mathbb{O} & \vdots \\ \vdots & \vdots & \ddots & \mathbb{O} \\ (h_{\mathbb{N}})_{(N_{E_{\mathbb{N}}}-1)N_b-N_{b_0}+1,1} & (h_{\mathbb{N}})_{(N_{E_{\mathbb{N}}}-1)N_b-N_{b_0}+1,2} & \cdots & h_{(N_{E_{\mathbb{N}}}-1)N_b-N_{b_0}+1,N_{E_{\mathbb{N}}}} \\ \vdots & \vdots & \vdots & \vdots \\ (h_{\mathbb{N}})_{N_p,1} & (h_{\mathbb{N}})_{N_p,2} & \cdots & (h_{\mathbb{N}})_{N_p,N_{E_{\mathbb{N}}}} \end{bmatrix} \quad \forall N_{b_0} = [0, N_b - 1] \tag{5.42}$$

where $(h_{\mathbb{N}})_{i,n}$ is defined as:

$$(h_{\mathbb{N}})_{k,n} = \sum_{j=j_0}^{j_f} h_{k,j} \qquad \begin{matrix} \forall k = [1, N_p] \\ \forall n = [1, N_{E_{\mathbb{N}}}] \end{matrix} \tag{5.43}$$

with the range $j_0 \to j_f$ defined as:

$$j_0 \to j_f = \begin{cases} 1 & \to & \min(k, N_b - N_{b_0}) & \text{For } n = 1 & \text{(First Block)} \\ (n-1)N_b - N_{b_0} + 1 & \to & \min(k, nN_b - N_{b_0}) & \text{For } 1 < n < N_{E_{\mathbb{N}}} & \text{(Inner Blocks)} \\ (n-1)N_b - N_{b_0} + 1 & \to & \min(k, N_p) & \text{For } n = N_{E_{\mathbb{N}}} & \text{(Last Block)} \end{cases} \tag{5.44}$$

Note that the fundamental operation in question is the summation of the entire column-elements of H related to each block segment, depending on the time-step (N_{b_0}) at which the Ideal MWB input-structure is applied. Although the direct implementation of these expressions/summatories would be correct, and indeed for simulation purposes might be easier to implement, eg. forming $\mathbb{N}_{N_{b_0}}$ and then computing $H_{\mathbb{N}} = H\mathbb{N}_{N_{b_0}}$ explicitly, it would require the calculation of the entire matrix H which would require additional memory, and potentially higher computation times as we will demonstrate shortly. This brings the question of whether we can directly compute $H_{\mathbb{N}}$ as it was done in chapter 4, section 4.2. To address this, let us consider an example that would allow its calculation through recursive expressions whilst allowing us to illustrate the potential computational benefits.

Consider the first 5 elements of the first column of (5.42) when using a block size of $N_b = 3$, with a Ideal Prediction Horizon of $N_p = 7$ at the time-step $N_{b_0} = 0$, given by:

$$\begin{bmatrix} (h_{\mathbb{N}})_{1,1} \\ (h_{\mathbb{N}})_{2,1} \\ (h_{\mathbb{N}})_{3,1} \\ (h_{\mathbb{N}})_{4,1} \\ (h_{\mathbb{N}})_{5,1} \end{bmatrix} = \begin{bmatrix} B_0 \\ A_1B_0 + B_1 \\ A_2A_1B_0 + A_2B_1 + B_2 \\ A_3A_2A_1B_0 + A_3A_2B_1 + A_3B_2 \\ A_4A_3A_2A_1B_0 + A_4A_3A_2B_1 + A_4A_3B_2 \end{bmatrix} = \begin{bmatrix} B_0 \\ A_1(h_{\mathbb{N}})_{1,1} + B_1 \\ A_2(h_{\mathbb{N}})_{2,1} + B_2 \\ A_3(h_{\mathbb{N}})_{3,1} \\ A_4(h_{\mathbb{N}})_{4,1} \end{bmatrix} \tag{5.45}$$

Let us begin by observing that the cyan and pink parts correspond to the state predictions of H at time-steps ($k = 1 \rightarrow N_b$) over which the initial block segment is embedded (ie. $\mathbb{U}_0 = u_0 = u_1 = u_2$), and the yellow elements correspond to the elements "beyond" that initial block, related to the effects of propagating that first block segment on later steps ($k = N_b + 1 \rightarrow N_p$). Note that other columns would have similar sections, ie. elements related to the segment over which the block is embedded, and elements beyond that block. What we can derive from this is that each column-block segment will have up to 3 main parts, namely: an initial value (eg. B_0), depicted by the cyan cell, representing the first non-zero value of the column; an inner block-segment value, represented by the pink cell which can be calculated with the use of recursive expression $(h_{\mathbb{N}})_{k,i} = A_{k-1}(h_{\mathbb{N}})_{k-1,i} + B_{k-1}$ as in the red box of 5.45; and outer block-segment value, represented by the yellow cell, which can be calculated with the use of recursive expression $(h_{\mathbb{N}})_{k,i} = A_{k-1}(h_{\mathbb{N}})_{k-1,i}$ as in the blue box of 5.45, similar to the recursive expression of H (3.19c). The use of these 3 main parts and recursive expressions was key for the development of the algorithm 5.3 and can be seen explicitly in lines 7, 10 and 13 of the algorithm, where it selects which expression to use depending on the position of the block segment, supported by the general method for handling the shifting strategy introduced earlier in subsection 5.2.1. Note that in some cases, the column will only have 1 or 2 parts, eg. the first and last block segments. Finally, it is straightforward to see that the use of this method will allow a substantial amount of computation reduction. As an example, the calculation of the last element $(h_{\mathbb{N}})_{5,1}$, would require 3 matrix-matrix multiplications for H and 2 matrix summations to group them into 1. In contrast, the proposed recursive expression only requires 1 matrix-matrix multiplication.

On the other hand, the time-varying nature of the non-zero elements in (5.42) makes it particularly problematic for auto-generation routines given it involves using memory spaces that can overlap with each other. This can be seen in the cyan parts of equation (5.46) where the first $N_b + 1$ rows and 2 columns for a given time-varying $H_{\mathbb{N}}$ matrix can be seen. For this reason, the algorithm requires to "clean" this specific sections before using them to avoid any potential memory problem. This can be seen in lines 17-19 of algorithm 5.3. The resulting algorithm is given in the core algorithms subsection 5.2.4.

$$\underbrace{H_{\mathbb{N}} = \begin{bmatrix} (h_{\mathbb{N}})_{1,1} & \mathbb{O} \\ \vdots & \vdots \\ (h_{\mathbb{N}})_{N_b,1} & \mathbb{O} \\ (h_{\mathbb{N}})_{N_b+1,1} & (h_{\mathbb{N}})_{N_b+1,2} \end{bmatrix}}_{\text{For } N_{b_0} = 0} \qquad \underbrace{H_{\mathbb{N}} = \begin{bmatrix} (h_{\mathbb{N}})_{1,1} & \mathbb{O} \\ (h_{\mathbb{N}})_{2,1} & (h_{\mathbb{N}})_{2,2} \\ \vdots & \vdots \\ (h_{\mathbb{N}})_{N_b+1,1} & (h_{\mathbb{N}})_{N_b+1,2} \end{bmatrix}}_{\text{For } N_{b_0} = N_b - 1} \tag{5.46}$$

5.2.3 Extension of the $O(N)$ and $O(N^2)$ Algorithms

Another key part for efficiently implementing the proposed approach is that of the compressed Hessian computation ($E_{\mathbb{N}}$), and compressed linear term ($f_{\mathbb{N}}$). To achieve this, we can use a similar approach to that of the $O(N)$ and $O(N)^2$ algorithms [8] which are known to be key for achieving fast computation times through the use of recursive expressions.

Following a similar procedure to the one presented in chapter 3, section 3.3, subsection 3.3.1: The Re-derivation of the $O(N^2)$ and $O(N)$ algorithms, let us consider the first column of the compressed Hessian term ($E_{\mathbb{N}} = H_{\mathbb{N}}^T Q H_{\mathbb{N}}$) that results from a using a short horizon of $N_p = 3$, with a block-size $N_b = 2$ in the initial time step $N_{b_0} = 0$, ie. embedding the equality on the first two blocks ($\hat{\mathbb{U}}_1 = \hat{u}_0 = \hat{u}_1$), given by:

$$\begin{bmatrix}(E_{\mathbb{N}})_{1,1}\\(E_{\mathbb{N}})_{2,1}\end{bmatrix} = \underbrace{\begin{bmatrix}B_0^T & B_0^T A_1^T + B_1^T & B_0^T A_1^T A_2^T + B_1^T A_2^T\\ 0 & 0 & B_2^T\end{bmatrix}}_{H_{\mathbb{N}}} \underbrace{\begin{bmatrix}\tilde{w}_{1,1}\\ \tilde{w}_{2,1}\\ \tilde{w}_{3,1}\end{bmatrix}}_{\tilde{W}} \tag{5.47}$$

where the dummy variable ($\tilde{W}$) captures the efficient operation of ($QH_{\mathbb{N}}$) through the use of the special diagonal-matrix multiplication available in the Eigen 3 library, as explained in equation (3.52).

The overall idea of the standard $O(N^2)$ algorithm is to use some form of backwards recursions in the dummy variables ($\tilde{W}$) which allow simple calculations of the standard Hessian terms, eg. given by $E_{k,i} = B_{k-1}^T \tilde{w}_{k,i}$. Therefore, we can look to use this method for the proposed approach.

Let us begin by calculating the last element $(E_{\mathbb{N}})_{2,1}$, simply given by:

$$(E_{\mathbb{N}})_{2,1} = B_2^T \tilde{w}_{3,1}^{[1]} \tag{5.48}$$

Looking at the standard $O(N^2)$ algorithm 3.2, the next step would be the recursive calculation of $\tilde{w}_{2,1}^{[1]} = \tilde{w}_{2,1}^{[0]} + A_2^T \tilde{w}_{3,1}^{[1]}$ for it to be used for the calculation of the following term of the Hessian as $E_{2,1} = B_1^T \tilde{w}_{2,1}^{[1]}$ (see equation 3.55). Note that implementing this in (5.47) for the element of $(E_{\mathbb{N}})_{1,1}$ does indeed capture the required expressions related to B_1^T, as signaled in the red box of (5.49), but misses the terms related to B_0^T. Similarly, the next step in the standard algorithm 3.2 would be the recursive calculation of $\tilde{w}_{1,1}^{[1]} = \tilde{w}_{1,1}^{[0]} + A_1^T \tilde{w}_{2,1}^{[1]}$ for it to be used in the calculation of the following term of the Hessian as $E_{1,1} = B_0^T \tilde{w}_{1,1}^{[1]}$ (see equation 3.56). Interestingly, this element represent the aforementioned missing elements from the previous calculation, as depicted by the blue box in (5.49).

Thus, the resulting term of the compressed Hessian $(E_{\mathbb{N}})_{1,1}$ is then composed by the summation of the two elements over which the equality was embedded, given by:

$$(E_{\mathbb{N}})_{1,1} = \boxed{B_0^T \left(\underbrace{\tilde{w}_{1,1}^{[0]} + A_1^T \overbrace{(\tilde{w}_{2,1}^{[0]} + A_2^T \tilde{w}_{3,1}^{[1]})}^{\tilde{w}_{2,1}^{[1]}}}_{\tilde{w}_{1,1}^{[1]}} \right)} + \boxed{B_1^T \overbrace{(\tilde{w}_{2,1}^{[0]} + A_2^T \tilde{w}_{3,1}^{[1]})}^{\tilde{w}_{2,1}^{[1]}}} \tag{5.49}$$

This can be done by using a recursive expression for the compressed Hessian term itself (not the dummy variable) to iteratively calculate them in the form of $(E_{\mathbb{N}})_{n,i}^{+} = (E_{\mathbb{N}})_{n,i}^{-} + B_{k-1}^T \tilde{w}_{k,i}$, eg. given by:

$$(E_{\mathbb{N}})^{[1]}_{1,1} = B_0^T \tilde{w}^{[1]}_{1,1} + \boxed{B_1^T \tilde{w}^{[1]}_{2,1}} = B_0^T \tilde{w}^{[1]}_{1,1} + \boxed{(E_{\mathbb{N}})^{[0]}_{1,1}} \tag{5.50}$$

What we can derive from this is that the total expression for all the compressed Hessian terms for any block size (N_b) on any time-step (N_{b_0}) can be calculated by the recursive addition of the standard algorithm elements 3.2 in the n_{th} block-segment to the Hessian term related to that segment $(E_{\mathbb{N}})_{n,i}$. As an example, the first column-elements of the compressed Hessian for the same horizon ($N_p = 3$), same block size ($N_b = 2$) in the following time-step ($N_{b_0} = 1$) would be given by:

$$\begin{bmatrix}(E_{\mathbb{N}})_{1,1}\\(E_{\mathbb{N}})_{2,1}\end{bmatrix} = \begin{bmatrix}B_0^T & B_0^T A_1^T & B_0^T A_1^T A_2^T\\ 0 & B_1^T & B_1^T A_2^T + B_2^T\end{bmatrix} \begin{bmatrix}\tilde{w}_{1,1}\\ \tilde{w}_{2,1}\\ \tilde{w}_{3,1}\end{bmatrix} = \begin{bmatrix}B_0^T \tilde{w}^{[1]}_{1,1}\\ B_1^T \tilde{w}^{[1]}_{2,1} + \boxed{B_2^T \tilde{w}^{[1]}_{3,1}}\end{bmatrix} = \begin{bmatrix}(E_{\mathbb{N}})^{[0]}_{1,1}\\ B_1^T \tilde{w}^{[1]}_{2,1} + \boxed{(E_{\mathbb{N}})^{[0]}_{2,1}}\end{bmatrix}$$

The same idea can be applied to the compressed linear term calculation ($f_{\mathbb{N}}$). Finally, a similar method can be used for the input-related terms. As an example, the compressed Hessian term related to the input weights ($\mathbb{N}^T R \mathbb{N}$) represent the summation of the weights of the n_{th} segment to the n_{th} diagonal term. This can be seen in lines 17 and 19 of algorithm 5.2.

The proposed extension of the $O(N)$ and $O(N^2)$ algorithms is given in algorithms 5.1 and 5.2.

Algorithm 5.1: Ideal MWB $O(N)$ Condensing Algorithm

Data: $Q, R, A_k, B_k, X_e, U_e, N_p, N_b, N_{b_0}$

1 **begin**
2 $N_{b_i} = N_{b_0}$; // Initialise inner block position index
3 $n = N_{E_{\mathbb{N}}}$; // Initialise working row of $f_{\mathbb{N}}$
4 $\tilde{w}_{N_p} = q_{N_p} X_{e_{N_p}}$; // Initial value of the dummy variable
// For loop running backwards $k = N_p, N_p - 1, \cdots, 2$
5 **for** $k = N_p$ **to** 2 **do**
6 **if** $k = N_p$ *OR* $N_{b_i} = N_b$ **then**
7 $(f_{\mathbb{N}})^0_n = -B^T_{k-1}\tilde{w}_k + r_{k-1}U_{e_{k-1}}$; // Initialise linear term component
8 **else**
9 $(f_{\mathbb{N}})^+_n = (f_{\mathbb{N}})^-_n - B^T_{k-1}\tilde{w}_k + r_{k-1}U_{e_{k-1}}$; // Add to linear term component
10 **end**
11 $\tilde{w}_{k-1} = q_{k-1}X_{e_{k-1}} + A^T_{k-1}\tilde{w}_k$; // Propagate recursively
12 $N^+_{b_i} = N^-_{b_i} - 1$; // Iterate inner block position index backwards
13 **if** $N_{b_i} < 1$ **then**
14 $n^+ = n^- - 1$; // Decrease row of $f_{\mathbb{N}}$ elements
15 $N_{b_i} = N_b$; // Reset inner block position index to N_b
16 **end**
17 **end**
18 **if** $N_{b_i} = N_b$ **then**
19 $(f_{\mathbb{N}})^0_0 = -B^T_0\tilde{w}_0 + r_0U_{e_0}$; // Initialise first component of linear term $(f_{\mathbb{N}})_0$
20 **else**
21 $(f_{\mathbb{N}})^+_0 = (f_{\mathbb{N}})^-_0 - B^T_0\tilde{w}_0 + r_0U_{e_0}$; // Add to first component of linear term $(f_{\mathbb{N}})_0$
22 **end**
23 **end**

Result: $f_{\mathbb{N}}$

Algorithm 5.2: Ideal MWB $O(N^2)$ Condensing Algorithm

Data: $H_{\mathbb{N}}, Q, R, A_k, B_k, N_p, N_b, N_{b_0}$

begin

- $N_{b_i} = N_{b_0}$; // Initialise inner block position index
- $n = N_{E_{\mathbb{N}}}$; // Initialise working row of $E_{\mathbb{N}}$ and column of $\tilde{W}$
- **for** $i = 1$ **to** n **do**
  - $\tilde{w}_{N_p,i} = q_{N_p}(h_{\mathbb{N}})_{N_p,i}$; // Initialise last row of $\tilde{W}$ ($\tilde{w}_{N_p,1\to n}$ elements)
- **end**
- // Main for loop running backwards $k = N_p, N_p - 1, \cdots, 2$
- **for** $k = N_p$ **to** 2 **do**
  - **for** $i = 1$ **to** $n - 1$ **do**
    - **if** $k = N_p$ *OR* $N_{b_i} = N_b$ **then**
      - $(E_{\mathbb{N}})^0_{n,i} = B^T_{k-1}\tilde{w}_{k,i}$; // Initialise Hessian component
    - **else**
      - $(E_{\mathbb{N}})^+_{n,i} = (E_{\mathbb{N}})^-_{n,i} + B^T_{k-1}\tilde{w}_{k,i}$; // Add to Hessian component
    - **end**
    - $\tilde{w}_{k-1,i} = q_{k-1}(h_{\mathbb{N}})_{k-1,i} + A^T_{k-1}\tilde{w}_{k,i}$; // Propagate recursively
  - **end**
  - **if** $k = N_p$ *OR* $N_{b_i} = N_b$ **then**
    - $(E_{\mathbb{N}})^0_{n,n} = B^T_{k-1}\tilde{w}_{k,n} + r_{k-1}$; // Initialise Hessian diagonal component
  - **else**
    - $(E_{\mathbb{N}})^+_{n,n} = (E_{\mathbb{N}})^-_{n,n} + B^T_{k-1}\tilde{w}_{k,n} + r_{k-1}$; // Add to Hessian diagonal component
  - **end**
  - **if** $N_{b_i} > 1$ **then**
    - $\tilde{w}_{k-1,n} = q_{k-1}(h_{\mathbb{N}})_{k-1,n} + A^T_{k-1}\tilde{w}_{k,n}$; // Propagate recursively
  - **end**
  - $N^+_{b_i} = N^-_{b_i} - 1$; // Iterate inner block position index backwards
  - **if** $N_{b_i} < 1$ **then**
    - $n^+ = n^- - 1$; // Decrease row of $E_{\mathbb{N}}$
    - $N_{b_i} = N_b$; // Reset inner block position index to N_b
  - **end**
- **end**
- // Calculate first diagonal element outside main loop
- **if** $N_{b_i} = N_b$ **then**
  - $(E_{\mathbb{N}})^0_{0,0} = B^T_0\tilde{w}_{0,0} + r_0$; // Initialise first diagonal element $(E_{\mathbb{N}})_{0,0}$
- **else**
  - $(E_{\mathbb{N}})^+_{0,0} = (E_{\mathbb{N}})^-_{0,0} + B^T_0\tilde{w}_{0,0} + r_0$; // Add to first diagonal element $(E_{\mathbb{N}})_{0,0}$
- **end**

end

Result: $E_{\mathbb{N}}$

These 2 algorithms are supported by the general method introduced in subsection 5.2.1 to keep track of the block positions (N_{b_i}) and number of segments (n) as required by the algorithms, for which the user must provide the time-step of the virtual block position indicator N_{b_0}. Interestingly, given the iterations of the dummy variable $\tilde{W}$ are required to run backwards, the algorithms require the inner block position indicator (N_{b_i}) and (n) to be decreased rather than increased as the algorithm iterates. Finally, note that the algorithm uses the range $N_{b_0} = [1, N_b]$ as discussed in remark 5.6.

5.2.4 Core Algorithms

In addition to the 2 novel extensions of the $O(N)$ and $O(N^2)$ algorithms presented in the previous subsection, the approach uses 3 additional "core" algorithms for its efficient implementation, namely: the Ideal MWB $H_{\mathbb{N}}$ Matrix Computation; a generic algorithm for selecting constraints with the Shifting Strategy; and a decompression routine for obtaining the original uncompressed variable $\delta\hat{U}$. These algorithms will then be used in combination with the $O(N)$ and $O(N^2)$ extensions, to develop the final RTI preparation and feedback algorithms introduced in the following subsection 5.2.5. All the algorithms are supported by the general method introduced in subsection 5.2.1 with the range the $N_{b_0} = [1, N_b]$ as discussed in remark 5.6.

Algorithm 5.3 calculates the $H_{\mathbb{N}}$ matrix based on the theory established in subsection 5.2.2.

Algorithm 5.3: Ideal MWB Condensing $H_{\mathbb{N}}$ Matrix Calculation

Data: $A_k, B_k, N_p, N_b, N_{b_0}$

begin
- $N_{b_i} = N_{b_0}$; // Initialise inner block position index
- $n = 1$; // Initialise columns of $H_{\mathbb{N}}$
- $j = n$; // Initialise limit for recursive loop
- **for** $k = 1$ **to** N_p **do**
  - **for** $i = 1$ **to** $j - 1$ **do**
    - $(h_{\mathbb{N}})_{k,i} = A_{k-1}(h_{\mathbb{N}})_{k-1,i}$; // Propagate recursively
  - **end**
  - **if** $k = 1$ *OR* $N_{b_i} = 1$ **then**
    - $(h_{\mathbb{N}})^0_{k,n} = B_{k-1}$; // Initialise n_{th} column element
    - $j^+ = j^- + 1$; // Increase limit for recursive loop
  - **else**
    - $(h_{\mathbb{N}})^+_{k,n} = (h_{\mathbb{N}})^-_{k,n} + B_{k-1}$; // Add to n_{th} column element
  - **end**
  - $N^+_{b_i} = N^-_{b_i} + 1$; // Iterate inner block position index forward
  - **if** $N_{b_i} > N_b$ **then**
    - **if** $n < N_{E_{\mathbb{N}}}$ **then**
      - $(h_{\mathbb{N}})_{2+(n-1)N_b \to nN_b+1, n+1} = 0$; // Clean elements of following column
    - **end**
    - $n^+ = n^- + 1$; // Increase columns row of $H_{\mathbb{N}}$
    - $N_{b_i} = 1$; // Reset inner block position index to 1
  - **end**
- **end**

end

Result: $H_{\mathbb{N}}$

On the other hand, algorithm 5.4 is provided as a "generic" algorithm that sets up the framework for implementing the selected constraints of the shifting strategy by detecting the edge of each of the blocks with the conditional statement: if ($N_{b_i} = N_b$ OR $k = N_p$), then "select", else "ignore", visible in lines 5-11 of the algorithm.

Algorithm 5.4: Generic Algorithm for Constraint Selection using Shifting Strategy

Data: $H_{\mathbb{N}}, \bar{X}, D, \bar{U}, X_{max}, X_{min}, U_{max}, U_{min}, N_p, N_b, N_{b_0}$

1 **begin**
2 $N_{b_i} = N_{b_0}$; // Initialise inner block position index
3 $n = 1$; // Initialise constraint element number
4 **for** $k = 1$ **to** N_p **do**
5 **if** $N_{b_i} = N_b$ *OR* $k = N_p$ **then**
// Select n_{th} Constraint Elements
6 $(M, \gamma)_{selected} \leftarrow (h_{\mathbb{N}})_{k,1\to n}, \bar{x}_k, \bar{d}_k, \bar{u}_{k-1}, x_{max}, x_{min}, u_{max}, u_{min}$
7 $n^+ = n^- + 1$; // Increase constraint element number
8 $N_{b_i} = 1$; // Reset inner block position index to 1
9 **else**
10 $N_{b_i}^+ = N_{b_i}^- + 1$; // Iterate inner block position index forward
11 **end**
12 **end**
13 **end**

Result: $M_{selected}, \gamma_{selected}$

Finally, algorithm 5.5 calculates the original uncompressed optimal input trajectory ($\delta\hat{U}^*$) from the blocked decision variables ($\delta\hat{\mathbb{U}}^*$). Note that this algorithm can also be used for decompressing the non-relative blocked variables ($\hat{\mathbb{U}}^*$) into the original vector ($\hat{U}^*$).

Algorithm 5.5: Ideal MWB Decompression Routine

Data: $\delta\hat{\mathbb{U}}^*, N_b, N_{b_0}$

1 **begin**
2 $N_{b_i} = N_{b_0}$; // Initialise inner block position index
3 $n = 1$; // Initialise row of $\delta\hat{\mathbb{U}}^*$
4 **for** $k = 0$ **to** $N_p - 1$ **do**
5 $\delta\hat{u}_k^* = \delta\hat{\mathbb{U}}_n^*$; // Copy n_{th} element of $\delta\hat{\mathbb{U}}^*$ to k_{th} element of $\delta\hat{U}^*$
6 $N_{b_i}^+ = N_{b_i}^- + 1$; // Iterate inner block position index forward
7 **if** $N_{b_i} > N_b$ **then**
8 $n^+ = n^- + 1$; // Increase row of $\delta\hat{\mathbb{U}}^*$
9 $N_{b_i} = 1$; // Reset inner block position index to 1
10 **end**
11 **end**
12 **end**

Result: $\delta\hat{U}^*$

5.2.5 RTI Algorithms

Having defined the algorithms required to implement the proposed Shifting Strategy with the Ideal Moving Window Blocking input-parameterisation using the RTI Scheme, the overall approach is finally provided in terms of the preparation and feedback phases given in algorithms 5.6 and 5.7, respectively. Both of this algorithms are based on the previously presented algorithms, including some of the ones introduced in chapter 3 to facilitate the verification process of each working part of the proposed approach. The key part to use these algorithms effectively is to understand the working principle of the absolute-time frame, and its relation to the virtual block position indicator (N_{b_0}).

Algorithm 5.6: Ideal MWB RTI NMPC Preparation Phase Algorithm with Shifting Strategy

Data: $\bar{X}, \bar{U}, \bar{\lambda}_{sel}, x_{-1}, u_{-1}, Q, R, N_p, N_b, N_{b_0}$

begin

- $\bar{x}_0 = f(x_{-1}, u_{-1})$; // Calculate predicted state from previous state and input
- Shift $\bar{X}$,$\bar{U}$ consistently ; // Initial Value Embedding
- **if** $N_{b_0} = 1$ **then**
  - Shift $\bar{\lambda}_{sel}$ consistently ; // Shifting Lagrange Multipliers theorem 5.2
- **end**
- $[A_k, B_k, d_k] = Forward(\bar{X}, \bar{U}, \bar{x}_0, N_p)$; // Run algorithm 3.4
- $[H_{\mathbb{N}}] = CalculateH_{\mathbb{N}}(A_k, B_k, N_p, N_b, N_{b_0})$; // Run algorithm 5.3
- $[E_{\mathbb{N}}] = CalculateE_{\mathbb{N}}(H_{\mathbb{N}}, Q, R, A_k, B_k, N_p, N_b, N_{b_0})$; // Run algorithm 5.2
- $(M_{\mathbb{N}})_{sel} = \begin{bmatrix} I & -I & (H_{\mathbb{N}})_{sel}^T & -(H_{\mathbb{N}})_{sel}^T \end{bmatrix}^T$; // Form $(M_{\mathbb{N}})_{sel}$ using algorithm 5.4

end

Result: $E_{\mathbb{N}}, (M_{\mathbb{N}})_{sel}, A_k, B_k, d_k, \bar{x}_0$

Algorithm 5.7: Ideal MWB RTI NMPC Feedback Phase Algorithm with Shifting Strategy

Data: $x_0, \bar{x}_0, \bar{X}, \bar{U}, \bar{\lambda}_{sel}, X_r, U_r, E_{\mathbb{N}}, (M_{\mathbb{N}})_{sel}, A_k, B_k, d_k, Q, R, U_{max}, U_{min}, X_{max}, X_{min}, N_p, N_b, N_{b_0}$

begin

- $\delta x_0 = x_0 - \bar{x}_0$; // Calculate state deviation from measurement
- $[D] = FormD(A_k, d_k, \delta x_0, N_p)$; // Run algorithm 3.6
- $X_e = X_r - \bar{X} - D$; // Calculate X error
- $U_e = \bar{U} - U_r$; // Calculate U error
- $[f_{\mathbb{N}}] = Calculatef_{\mathbb{N}}(H_{\mathbb{N}}, Q, R, A_k, B_k, X_e, U_e, N_p, N_b, N_{b_0})$; // Run algorithm 5.1
- $\gamma_{sel} = \begin{bmatrix} (U_{max} - \bar{U})_{sel} \\ (\bar{U} - U_{min})_{sel} \\ (X_{max} - \bar{X} - D)_{sel} \\ (\bar{X} + D - X_{min})_{sel} \end{bmatrix}$; // Form γ_{sel} vector using algorithm 5.4
- $[\delta\hat{\mathbb{U}}^*, \bar{\lambda}_{sel}] = QPSolve(E_{\mathbb{N}}, f_{\mathbb{N}}, (M_{\mathbb{N}})_{sel}, \gamma_{sel}, \bar{\lambda}_{sel})$; // Solve the Quadratic Program
- $[\delta\hat{U}^*] = Decompress(\hat{\mathbb{U}}^*, N_p, N_b, N_{b_0})$; // Run algorithm 5.5
- $\bar{U} = \bar{U} + \delta\hat{U}^*$; // Calculate new nominal input
- $[\delta\tilde{X}] = Expand(A_k, B_k, \delta\hat{U}^*, N_p)$; // Run algorithm 3.7
- $\bar{X} = \bar{X} + D + \delta\tilde{X}$; // Calculate new nominal state

end

Result: $\bar{X}, \bar{U}, \bar{\lambda}_{sel}$

5.2.6 Generic Computations Comparison

In order to evaluate how well the proposed algorithms perform, we performed a generic computation comparison of three of the main algorithms, namely: algorithm 5.3 for calculating $H_{\mathbb{N}}$ efficiently; the extended $O(N^2)$ algorithm 5.2 for calculating $E_{\mathbb{N}}$ efficiently; and the extended $O(N)$ algorithm 5.1 for calculating $f_{\mathbb{N}}$ efficiently. Note that when the block size is one ($N_b = 1$), all of these algorithms represent the exact same algorithms as their counterparts in the standard approach introduced in chapter 3, namely: algorithms 3.5, 3.2 and 3.3, with the main difference being the use of conditionals from the general method of section 5.2.1 to keep track of the relevant block positions and numbers. In practice, we observed there was little to no difference from the additional conditional checks when comparing with the standard algorithms. Thus, we are interested in comparing how well the proposed approach performs for different block sizes against the standard case ($N_b = 1$).

To perform this comparison, we selected two systems with different dimensions, namely: the Inverted Pendulum, introduced later in the case study of section 5.5, which has $n_x = 4$ states and $n_u = 1$ inputs; and the Quadrotor of case study 4.3, which has $n_x = 7$ states and $n_u = 4$ inputs. Each of these systems was evaluated using block sizes of $N_b = [1 \rightarrow 6]$ with various Ideal Prediction Horizons, namely: $N_p = [121, 241]$ for the Inverted Pendulum, and $N_p = [61, 121]$ for the Quadrotor. The algorithms were then programmed using automatically generated C++ codes for each of the cases based on the Eigen 3 library, and were tested in Ubuntu 20.04 running with Real-Time priority (ie. chrt -r 99 ./main) on a laptop with an Intel i7-8750 CPU overclocked @ 3.9 GHz, and 32 GB DDR4 RAM @ 2,666 MHz, with 120,000 runs per algorithm. The test C++ codes were compiled using the (-O3) optimisation C-flag, as well as with the fused-multiply-addition operations (-mfma) and auto-vectorisation (-mavx) flags enabled to use the Advanced Vector Instruction set available in the Intel CPU. The results of this comparison are gathered in table 5.5 where the minimum computation time obtained for each algorithm is reported, indicating the minimum time that could be achieved if a Real-Time OS would be used.

Case	$n_x = 4, n_u = 1, N_p = 121$						$n_x = 4, n_u = 1, N_p = 241$					
N_b	1	2	3	4	5	6	1	2	3	4	5	6
$H_{\mathbb{N}}$ (alg. 5.3)	9	$5_{1.8}$	3_{3}	3_{3}	$2_{4.5}$	$2_{4.5}$	35	$18_{1.9}$	$12_{2.9}$	$9_{3.9}$	7_{5}	7_{5}
$E_{\mathbb{N}}$ (alg. 5.2)	17	$9_{1.9}$	$7_{2.3}$	$6_{2.8}$	$5_{3.4}$	$5_{3.4}$	71	35_{2}	$23_{3.1}$	$17_{4.2}$	$15_{4.7}$	$13_{5.5}$
$f_{\mathbb{N}}$ (alg. 5.1)	2	2	2	2	2	2	2	2	2	2	2	2
Total	28	$16_{1.8}$	$12_{2.3}$	$11_{2.5}$	$9_{3.1}$	$9_{3.1}$	108	55_{2}	$37_{2.9}$	$28_{3.9}$	$24_{4.5}$	$22_{4.9}$
Case	$n_x = 7, n_u = 4, N_p = 61$						$n_x = 7, n_u = 4, N_p = 121$					
N_b	1	2	3	4	5	6	1	2	3	4	5	6
$H_{\mathbb{N}}$ (alg. 5.3)	50	$26_{1.9}$	$18_{2.8}$	$14_{3.6}$	$12_{4.2}$	10_{5}	199	98_{2}	$62_{3.2}$	50_{4}	40_{5}	$35_{5.7}$
$E_{\mathbb{N}}$ (alg. 5.2)	78	$41_{1.9}$	$29_{2.7}$	$23_{3.4}$	$20_{3.9}$	$17_{4.6}$	319	161_{2}	107_{3}	$84_{3.8}$	$67_{4.8}$	$59_{5.4}$
$f_{\mathbb{N}}$ (alg. 5.1)	2	2	2	1	1	1	3	3	2	2	2	2
Total	130	$69_{1.9}$	$49_{2.7}$	$38_{3.4}$	$33_{3.9}$	$28_{4.6}$	521	262_{2}	171_{3}	$136_{3.8}$	$109_{4.8}$	$96_{5.4}$

Table 5.5: Generic Computation Times (in μs) Comparison of Ideal MWB Approach for two different scenarios: 1. A system with $n_x = 4$ states and $n_u = 1$ inputs, using different block sizes $N_b = [1, 2, 3, 4, 5, 6]$ with Ideal Prediction Horizons $N_p = [121, 241]$; and 2. A system with $n_x = 7$ and $n_u = 4$ inputs, using different block sizes $N_b = [1, 2, 3, 4, 5, 6]$ with Ideal Prediction Horizons $N_p = [61, 121]$. The gain factor (α) is indicated in red.

In this table (table 5.5) each of the cases are signaled in the gray coloured cases, with the computing times being reported beneath for the $N_b = [1 \rightarrow 6]$ block sizes for each of the algorithms (alg. 5.3, 5.2 and 5.1), and the "Total" cyan coloured rows representing the summation of this 3 algorithms. Moreover, most of the table cells contain a red-coloured under-script indicating a "gain factor" (α) related to how many times faster is the specific algorithm-case than the standard counterpart ($N_b = 1$), eg. $H_{\mathbb{N}}$ calculation with $n_x = 4, n_u = 1, N_p = 121, N_b = 6$ is $\alpha \approx 4.5$ faster than $n_x = 4, n_u = 1, N_p = 121, N_b = 1$. The row of the linear term $f_{\mathbb{N}}$ is not signaled as it discussed later in table 5.6. What we can observe from this table is that the gain factor follows closely the block-size itself, ie. most of the time being proportional to the block size in the form of: $\alpha \approx kN_b$, with $k \approx 1$. This is seen more clearly in the the highest computing time cases such as the Quadrotor ($n_x = 7, n_u = 4$) with $N_p = 121$. The proposed approach can be seen to be up to 5.4 faster than the standard approach for implementing the relevant "preparation" algorithms. Regardless, note that further computational benefits will arise from solving the resulting QPs themselves given the reduced sizes of dof and constraints. We will show an example of this in table 5.12 of case study 5.5 where gains up to 54 times faster were obtained.

To corroborate these results, a secondary platform was used: the Beaglebone Blue, a robotics-oriented Linux-based embedded platform running a quasi-Real-Time Debian OS @ 1 GHz with a NEON floating-point accelerator (fpa), specifically designed to perform efficient parallel floats computations. Only two cases were evaluated in this platform: the Inverted Pendulum sizes with $N_p = 121$, and the Quadrotor sizes with $N_p = 61$. The developed C++ codes were compiled using the (-O3) optimisation C-flag, and the (-mfpu=neon) flag to enable the NEON instructions set which luckily, the Eigen 3 library is prepared to handle. The codes were tested using float precision to take advantage of the NEON fpa and were run using real-time priority (ie. chrt -r 99 ./main) for 1250 iterations per algorithm. The comparison is presented in table 5.6 where the minimum time is reported. From this table it can be appreciated that the approach maintains a similar performance to that of table 5.5. However, it is interesting to note that the $f_{\mathbb{N}}$ computation times remain practically constant. This is because the number of operations in algorithm 5.1 remains constant for all block sizes N_b, with the only difference being the selection of the $(f_{\mathbb{N}})_n$ element to which the operation is assigned, as visible in lines 7, 9, 19, 21.

Case	$n_x = 4, n_u = 1, N_p = 121$					
N_b	1	2	3	4	5	6
$H_{\mathbb{N}}$ (alg. 5.3)	644	$288_{2.2}$	$194_{3.3}$	$140_{4.6}$	$124_{5.2}$	107_{6}
$E_{\mathbb{N}}$ (alg. 5.2)	767	$432_{1.8}$	$317_{2.4}$	$248_{3.1}$	$209_{3.7}$	$184_{4.2}$
$f_{\mathbb{N}}$ (alg. 5.1)	16	16_{1}	16_{1}	$15_{1.1}$	$15_{1.1}$	$15_{1.1}$
Total	1427	$737_{1.9}$	$527_{2.7}$	$403_{3.5}$	$348_{4.1}$	$306_{4.7}$
Case	$n_x = 7, n_u = 4, N_p = 61$					
N_b	1	2	3	4	5	6
$H_{\mathbb{N}}$ (alg. 5.3)	4703	$2543_{1.9}$	$1716_{2.7}$	$1353_{3.5}$	$1137_{4.1}$	$991_{4.7}$
$E_{\mathbb{N}}$ (alg. 5.2)	5394	$3030_{1.8}$	$2108_{2.6}$	$1658_{3.3}$	$1391_{3.9}$	$1214_{4.4}$
$f_{\mathbb{N}}$ (alg. 5.1)	44	$40_{1.1}$	$38_{1.2}$	$38_{1.2}$	$38_{1.2}$	$38_{1.2}$
Total	10141	$5523_{1.8}$	$3862_{2.6}$	$3049_{3.3}$	2566_{4}	$2243_{4.5}$

Table 5.6: Generic Computation Times (in μs) Comparison of Ideal MWB Approach in the Beaglebone Blue for two different scenarios: 1. A system with $n_x = 4$ states and $n_u = 1$ inputs, and 2. A system with $n_x = 7$ and $n_u = 4$ inputs, using different block sizes $N_b = [1, 2, 3, 4, 5, 6]$ with Ideal Prediction Horizons of $N_p = 121$ and $N_p = 61$, respectively. The gain factor is indicated in red.

5.3 Case Study: The Van Der Poll Oscillator

One of the first dynamical systems encountered when revising the literature available of NMPC was the Van Der Poll Oscillator from [23]. To quote the authors of this work, "This system is known to have a limit cycle, and with actuator constraints it is difficult to stabilise." Moreover, in this work the authors give a convincing case in which a "non-predictive" fixed-state feedback controller fails to stabilise the system due to the aforementioned constraints. On the other hand, the system in question is composed of 2 states and 1 input, thus making it a perfect example in which condensing-based approaches typically outperform those of sparse representations [145]. Additionally, given the difficulty related to input-constraints, it also makes a perfect case study for a system which struggles with this basic type of constraints, which recalling from the introduction of this chapter, was the initial motivation into looking at the Moving Window Blocking approach. Thus, this system is presented as a first example on which implementing the proposed approach would be ideal.

5.3.1 Modelling, Simulation and Optimisation

The Van Der Poll Oscillator is defined by the following nonlinear differential equation:

$$\ddot{p} = \mu(1 - p^2)\dot{p} - p + u \tag{5.51}$$

with μ being a positive value, selected as $\mu = 2$ as in [23] for our simulations.

This can be translated in the standard state space ($\dot{x} = f(x, u)$) simply by:

$$\underbrace{\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix}x_2\\ \mu(1 - x_1^2)x_2 - x_1 + u\end{bmatrix}}_{f(x,u)} \tag{5.52}$$

with $x_1 = p$ and $x_2 = \dot{p}$.

The system was simulated using the Explicit Forward Euler algorithm 3.1 with a sampling time of $T_s = 0.1$ (s) as in [23], and with single Euler step ($N_s = 1$). Moreover, the optimisation was subject to $q_{k+i} = diag([1, 0.5])$ $\forall i = [1, N_p]$ weights for penalising state errors (no infinite horizon costing used), $r_{k+i} = diag([0.1])$ $\forall i = [0, N_p - 1]$ weights for penalising input errors as in [23], and subject only to input inequality constraints considered as $-1 \leq \hat{u}_{k+i} \leq 1$ $\forall i = [0, N_p - 1]$. Furthermore, all the simulations performed started from steady-state condition at $x_0 = [1, 0]^T$ as in [23], and the references (or objective) of the system was to bring the state to zero (regulation - $x_r = [0, 0]^T$). Finally, the optimisation was done using single shooting with an initial guess for the nominal input trajectory selected as $\bar{U} = [1 \cdots 1]^T$ for the initial nominal state trajectory to start at a steady steady $\bar{X} = [x_0^T, \cdots, x_0^T]^T$ by cancelling the terms $-x_1 + u$ from the differential equation of $\dot{x}_2$. This satisfies one of the basic requirements of the RTI discussed in [55] which is starting at a global optimum, that being in this case the steady state at a previous target $x_r = [1, 0]^T$ before changing the reference.

In order to compare the performance of the proposed approach with that of the standard RTI NMPC, a $T = 15$ (s) was performed for block sizes $N_b = [1 \rightarrow 6]$ using an Ideal Prediction Horizon of $N_p = 61$. Recall that the standard RTI is achieved with $N_b = 1$. The comparison is gathered in table 5.7 where the resulting costs and percentage sub-optimalities $\Delta J = (\frac{J_{N_b}}{J_{STD}} - 1) \times 100\%$ can be seen.

N_b	1	2	3	4	5	6
Cost (J)	94.8	96	93.3	99.2	93.8	90.3
% Suboptimality (ΔJ)	-	1.3	-1.5	4.7	-1.1	-4.7

Table 5.7: Van Der Poll Oscillator Comparison of the Ideal MWB approach.

What is most interesting about the results from table 5.7 is that the solutions with block sizes $N_b = [3, 5, 6]$ actually outperformed the standard RTI NMPC solution ($N_b = 1$), as signaled by the red boxes where negative sub-optimalities were obtained. Given the solution with any block-size can actually be achieved with $N_b = 1$, ie. is a subset of the possible solutions of the standard RTI NMPC, the only reason this can be happening is if the optimisation would improve further AFTER re-linearising the system in future steps, eg. finding a steeper decreasing point for the cost after "waiting" with a constant input as opposed to the standard RTI NMPC which would take the steepest there is at each time-step. This is an example demonstration of the Boxing Analogy. This is the regular-person opening its guard, and the World Champion effortlessly finding it. Obviously this not a generalisable result, but it does give insight on a potential advantage of applying this approach to non-linear systems.

To provide a visual example of the comparison, figure 5.5 presents the simulations obtained with the standard RTI NMPC ($N_b = 1$), and with the Ideal MWB using block size $N_b = 6$. From this figure, it can clearly be appreciated how the solution embeds the desired blocking in the input trajectory, and performs small corrections on each block, as visible on the inner graph of the inputs, which allows the approach to make further improvements to the initial blocked plan as the horizon moves forward.

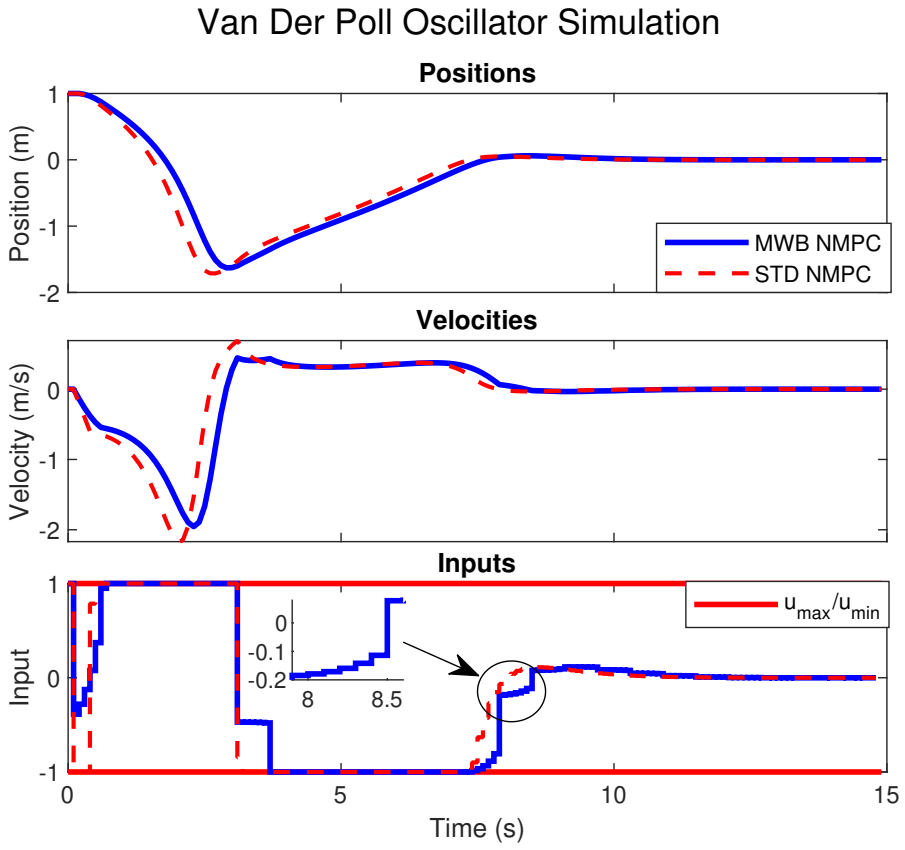


Figure 5.5: Van Der Poll Oscillator Comparison using Standard NMPC and Ideal MWB with $N_b = 6$.

5.4 Case Study: The Wave Energy Conversion Device

Another system which was considered for the implementation of the proposed MWB approach was the Wave Energy Conversion (WEC) Device which formed the main contribution of the IFAC World Congress 2020 paper [59], developed in collaboration with Ph.D. Student Juan Guerrero-Fernandez. This system was discussed briefly in the previous chapter for case study 4.5 where a set of Chebyshev polynomials were chosen as the input-parameterisation given their non-decaying properties (as opposed to eg. Laguerre Polynomials), and were able to get relatively good performance but presented a notorious disadvantage in the computation times obtained when using the "quadprog" function of Matlab for solving the optimisation, which motivation the contents of this chapter further.

In this case, we will present the application of the Non-Ideal MWB approach as it was applied in the paper, and compare it to various other strategies, including that of the Chebyshev polynomials of section 4.5. Note that the paper [59] applied the approach of extending the last block beyond the block-size (N_b) to maintain the dimensions of all the matrices constant, as discussed in the Intuitive Block Shifting example 5.2, which for practical purposes can be applied in this system given a long horizon is used ($N_p = 100$) and the objective is not to achieve nominal stability in the system, but rather to maximise energy extraction from the ocean wave energy. However, it is worth mentioning that adjusting the optimisation for it to use the closest Ideal Prediction Horizon, in this case $N_p = 101$ given $N_b = 5$ is used, made insignificant difference to the results that presented in this case study.

5.4.1 Modelling, Simulation and Optimisation

The WEC system modelling is well documented in the paper [59] and its contents are in high part thanks to expert WEC devices knowledge from Ph.D. Student Juan Guerrero-Fernandez, including the hydrodynamic modelling based on potential flow theory in which the model is decomposed into a summation of forces. Given the theory involved in obtaining the model to be used is outside the scope of this thesis, we refer to the final discrete-time model defined in equation (13) of the paper, where the system is represented by a linear state space model of the form:

$$x_{k+1} = Ax_k + B\Delta u_k + Bu_{exc_k} \tag{5.53a}$$

$$y = Cx_k \tag{5.53b}$$

where $x_k = [z_k, \dot{z}_k, x_{r_k}, u_{k-1}]^T$ is the state containing the vertical displacement of the buoy (z_k), the vertical velocity ($\dot{z}_k$), the radiation force approximation state (x_{r_k}), and the previous input (u_{k-1}); $\Delta u_k = u_k - u_{k-1}$ is the force increment in the Power Take-Off (PTO) system; u_{exc_k} is the excitation force of the wave; and the matrices A,B,C are defined as in the paper, with C being an extraction matrix that gives the output $y_k = [z_k, \dot{z}_k, u_{k-1}]^T$. The system sampling time was fixed at $T_s = 0.1$ (s).

The task is then to maximise the energy extraction by minimising the cost function:

$$\min \quad J_k = \sum_{i=1}^{N_p} \hat{u}_{k+i-1}\dot{z}_{k+i} \tag{5.54a}$$

$$s.t. \quad u_{min} \leq u_{k+i-1} \leq u_{max}, \quad \Delta u_{min} \leq \Delta u_{k+i-1} \leq \Delta u_{max} \tag{5.54b}$$

Although this is not a positive-definite cost function, it did result in a convex optimisation for our particular WEC system which can be solved by using the general output cost function (3.2) method introduced in chapter 3 with the non-relative modelling approach of equation (3.26) used by standard linear MPC, and selecting a special output weighting matrix ($q_{y_{k+i}}$) as defined in equation (16) of the paper which performs the crossed-terms multiplication. Moreover, in order to apply MWB strategy using input increments ($\Delta\hat{U}$) instead of the inputs ($\hat{U}$) directly, the paper applies a slightly different input-structure ($\mathbb{N}$), given in equation (19) of the paper. Note that the aforementioned input-structure is the shifted version of the alternative method for implementing the input-blocking scheme provided earlier in equation (5.3). On the other hand, the application of the MWB scheme to this problem results in the General Optimisation Framework discussed in section 5.1.4 as seen in equation (20) of the paper. Finally, given that the system is linear, the compressed Hessian ($E_{\mathbb{N}}$) and the compressed constraint matrix ($M_{\mathbb{N}}$) can be pre-stored offline to avoid any related calculations. This basically reduces the required computations to updating the compressed linear term $f_{\mathbb{N}}$ and constraints vector γ (both of which have negligible computation times), and solving the Quadratic Program afterwards.

Regarding the optimisation setup, a Prediction Horizon of $N_p = 100$ ($T_p = 10s$) was used with a block size of $N_b = 5$ for implementing the MWB approach. In order to evaluate how well the proposed approach performs against other strategies, several others types of solution were considered, including: 1. Standard MPC with Full Degrees of Freedom (F-DoF); 2. Unconstrained MPC with F-DoF, 3. Unconstrained MPC with F-DoF and the additional input-weighting terms $r_{k+i} = 1.12\,\forall i = [0, N_p{-}1]$ selected via a brute-force search of the penalty giving the highest energy absorption; 4. GPC with $N_u = \frac{N_p}{N_b} = 20$ decision variables; and 5. the Chebyshev Polynomials of section 4.5 with $N_{E_{\mathbb{N}}} = [20, 10, 5]$ coefficients. Other parameters such as constraint limits and parameters specific to the buoy and wave excitation are specified in the paper. These strategies were then simulated for $T = 600$ (s) and two performance metrics were captured, namely: Energy Extracted and Computation Times of constrained approaches.

5.4.2 Energy Extraction Comparison

Table 5.8 presents a comparison of the energy extracted by each of the aforementioned strategies. It is considered here that the energy obtained with the Standard MPC with F-DoF is the "maximum" that could be obtained with the selected prediction horizon, and therefore the efficiency of all the other strategies is measured relative to this performance. From this table it can be seen that the MWB approach outperforms all the other strategies, with the exception of the Standard MPC with F-DoF.

Method	Energy Extracted (MJ)	Efficiency (%)
Std MPC (F-DoF)	307	100
Unc. MPC (F-DoF)	-329	LOSS
Unc. MPC (r_k) (F-DoF)	272	88.7
MWB MPC ($N_b = 5$)	303	98.8
GPC ($N_u = 20$)	285	92.8
CHEV ($n_{\mathbb{N}} = 20$)	301	97.9
CHEV ($n_{\mathbb{N}} = 10$)	272	88.6
CHEV ($n_{\mathbb{N}} = 5$)	246	80.3

Table 5.8: Energy Extraction Comparison of Different Solutions applied to the Wave Energy Converter

This result is by no means unexpected, and can be explained with a few arguments. The first and most important argument is that the Standard MPC F-DoF solution most of the time is saturated in the input constraints u_{max}/u_{min}, with brief trajectories interconnecting them. This can be appreciated in figure 5.7 where the standard MPC solution for a section of the simulation is visible along with only 2 other solutions, namely: MWB and GPC, to avoid saturation of the figure. Recall that Chebyshev Polynomials were demonstrated to be incapable of replicating this behaviour in the optimised input trajectories for this particular system as seen in figure 4.7, even when using as high as 20 coefficients.

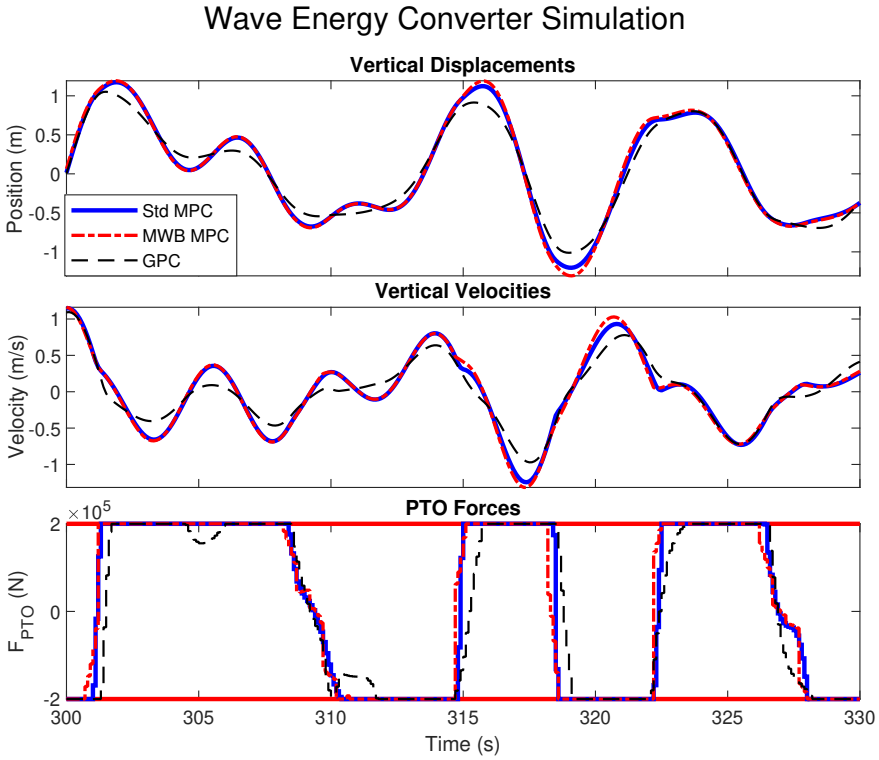


Figure 5.6: Simulation Example of Wave Energy Converter using Std. MPC, MWB MPC and GPC

On the other hand, the GPC approach suffers from ill-posedness, ie. the same problem explained in figure 5.1, where having all the decision variables congested at the beginning of the horizon is unable to replicate closely the plan of the Std. MPC F-DoF, and the optimisation is therefore making a plan that it is for all practical purposes "irrelevant" and inconsistent. This can be seen in figure 5.7 where the predicted optimal trajectories for various types of solutions at a given time-step are seen, namely: the Std MPC F-DoF prediction, depicted by the blue line; the MWB prediction, depicted by the cyan line; the GPC prediction, depicted by the dashed black line; and the Chebyshev Polynomials, depicted by the dot-dashed red line. Note that this is the same time-step used for figure 4.7 to show incapability of the Chebyshev Polynomials for replicating the original solution correctly due to the Gibbs-type phenomena. In contrast, the MWB can be seen to follow the input trajectory closely, with the displacement trajectory being virtually indistinguishable. Thus, this type of system or situation presents a perfect example in which decision variables at the end are as important as in the middle or beginning of the prediction horizon.

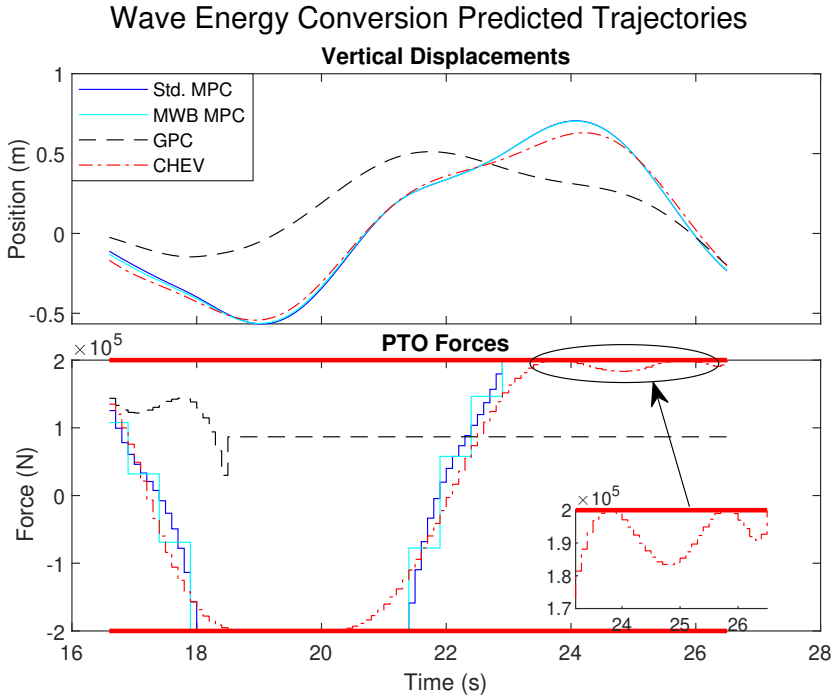


Figure 5.7: Example of Predicted Trajectories of Wave Energy Converter using Std. MPC, MWB MPC, GPC and Chebyshev

Lastly, both unconstrained solutions also naturally suffer from ill-posedness/inconsistency given that first of all they do not account for the required input saturations that may be required. Moreover, as this is an energy maximisation problem subject to an external excitation force, the unconstrained unweighted F-DoF solution tends to infinity, ie. if there were no input constraints, the input trajectories would quickly converge to infinity along with the energy produced which for all practical purposes is unrealistic and irrelevant to consider as a proper prediction. Because of this completely flawed plan-making, this type of solution results in a complete LOSS upon encountering the input constraints, as signaled in table 5.8. This phenomena is well known and was discussed in [90]. The alternative approach, which is to impose weights on the input to avoid this "exploding" phenomena, also discussed in [90], causes the optimisation targets, in this case "energy maximisation" and "input minimisation", to fight each other, ie. posing an inconsistent overall objecting, thus resulting in reduced performance.

5.4.3 Computation Times Comparison

The other metric of interest was the computation time required to perform the optimisation of the constrained approaches, namely: Std. MPC, MWB MPC, GPC, and Chebyshev Polynomials. Note that even though the system is linear, it may difficult to obtain an explicit solution given the high variability that the wave-excitation-force may present. For this reason, the WEC device may require an online solution to properly adjust the predictions according to the predicted wave-excitation-forces, thus requiring efficient methods to tackle the computational burden. Moreover, even if the Standard MPC F-DoF solution for a single WEC system may be computationally feasible, ie. not requiring the use of the proposed approach, the application of this method could be extended to wave-farms or robust multi-model approaches [20, 136] where multiple WEC models must be optimised simultaneously.

Table 5.9 presents the average computations times of each solution along with other relevant metrics such as number of QP iterations. All the resulting QPs were solved using the "quadprog" function of Matlab R2019b with the interior-point method, and the computation times involve both, the update of the linear term ($f/f_{\mathbb{N}}$) and the solution of the QP itself, noting that the other matrices/vectors required by the QP were pre-stored offline as discussed earlier. From this table we can clearly see how the MWB and GPC approaches present decent computational gains of up to 14.5 and 15.1, respectively, as signaled by the red boxes, whereas the CHEV approach presents the degraded performance discussed in case study 4.5. What is more interesting from the results of this table is that the MWB approach presents the smallest standard deviation in all the metrics, signaled by the green boxes, with the sole exception of the number of iterations of the Std MPC, signaled by the blue box. Moreover, the MWB approach presented the least amount of average iterations of the QP. Both of these indicators give evidence that the MWB approach has a clear consistency embedded in its underlying methodology.

Method	Avg. opt. time (ms)	Avg. opt. time per iter. (ms)	Avg. num. of QP Iterations	Computational Gain
Std MPC	28 ± 4.5	3.43 ± 0.5	8.19 ± 0.79	-
MWB MPC ($N_b = 5$)	1.94 ± 0.5	0.292 ± 0.086	6.75 ± 0.98	14.5
GPC ($N_u = 20$)	1.85 ± 0.74	0.257 ± 0.137	7.37 ± 1.17	15.1
CHEV ($n_{\mathbb{N}} = 20$)	93.1 ± 24.3	9.88 ± 0.97	9.44 ± 2.67	0.3
CHEV ($n_{\mathbb{N}} = 10$)	82.5 ± 10.8	9.13 ± 0.79	9.05 ± 1.06	0.326
CHEV ($n_{\mathbb{N}} = 5$)	82.2 ± 38.7	9.39 ± 0.83	8.8 ± 4.6	0.333

Table 5.9: WEC problem Computation Times comparison of different Methods using the "quadprog" QP solver of Matlab R2019b. Note: Computational Gain defined as t_{std}/t_{method} using avg. opt. time.

5.5 Case Study: The Inverted Pendulum

As discussed in the introduction chapter 1, the Inverted Pendulum was of special interest to the author of this thesis given prior experience with it for the physical implementation of advanced control methodologies which ultimately resulted in the experimental implementation of NMPC for a Double Inverted Pendulum presented later in case study 6.6. Indeed, it is well known that the inverted pendulum is a complex nonlinear system that was key for developing advanced rocket guidance systems [155]. Nowadays it is widely used by academics given it presents several control challenges such as non-linear and non-minimum phase dynamics, physical constraints and under-actuation (multiple outputs - single input), where the task is to drive the pendulum to its upright position, and simultaneously control its position in a rail. This makes it an interesting and challenging benchmark for NMPC.

For reference, part of the results of this case study were published in IET Journal [50].

5.5.1 System Modeling

Several variations of the mathematical model of an inverted pendulum have been used, some of which are more complex than others. For our simulation, we used the mathematical model presented in [5] which contains two main non-linearities, namely, the gravitational effect, $g\sin(\theta)$, and the non-linear torque-relationship $\cos(\theta)u$ of the bar-link with the input u (or car acceleration $\ddot{p} = u$).

The model is given by the following differential equations:

$$\begin{bmatrix} \ddot{p} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} ku \\ -b\dot{\theta} + g_0 \sin\theta + \cos\theta u \end{bmatrix} \tag{5.55}$$

were θ is the angle of the pendulum; p is the position of the cart; u is the acceleration of the cart which is considered the input of the system; $b = 0.3$ is a friction coefficient; $k = 1$ is an input-to-acceleration gain coefficient; and $g_0 = 9.81$ (m/s^2) is the gravitational constant.

Assuming the state $x = \begin{bmatrix} \dot{p} & \dot{\theta} & p & \theta \end{bmatrix}^T = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}^T$, a standard nonlinear state space of the form $\dot{x} = f(x, u)$ can be obtained, given by:

$$\underbrace{\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix} ku \\ -bx_2 + g_0 \sin(x_4) + \cos(x_4)u \\ x_1 \\ x_2 \end{bmatrix}}_{f(x,u)} \tag{5.56}$$

This system was simulated using the Explicit Euler integration method of algorithm 3.1 with a sampling time of $T_s = 0.025$ (s) and using a single integration step $(N_s = 1)$.

5.5.2 Simulation and Optimisation Setup

Regarding the simulation setup, all the simulations were done in the nominal case (no noise, no disturbances, no uncertainty) given that the prime interest is in the "inner" nominal stability and recursive feasibility properties and computational efficiency. Disturbance rejection and noise cancellation can be addressed separately using offset-free optimisations [67, 69] and observer/estimator design or filters [19, 121, 122, 146] respectively. The simulation was initialised at the lower equilibrium $x_0 = \begin{bmatrix} 0 & 0 & 0 & \pi \end{bmatrix}$, and was run for $T = 8$ (s), allowing the system to swing up in "one shot" or "two shots" (see figure 5.9).

On the other hand, the optimisation was solved using single-shooting and was initialised with the "free-response" of the system simply by using a nominal input guess trajectory of zeros $\bar{U} = \mathbb{O}$. The references of the optimisation where selected as $x_{r_{k+i}} = [0, 0, 0, 0]^T$ $\forall i = [1, N_p]$ and $u_{r_{k+i}} = 0$ $\forall i = [0, N_p - 1]$ which satisfy the unbiased prediction models and costs requirements for offset-free control as discussed in section 3.5 of chapter 3. A "desired" prediction horizon of $N_{p_{des}} = 50$ $(T_p = 1.25$ $(s))$ was selected and the closest upper Ideal Prediction Horizon required for implementing the proposed approach was then calculated as $N_p = \left\lceil \frac{N_{p_{des}}}{N_b} \right\rceil N_b + 1$ depending on the selected block size (N_b). For reference, the selected Ideal Horizon is displayed in the parenthesis next to the block size on the results presented in tables 5.10 and 5.11. Regarding the tuning parameters, the optimisation was done using $q_{k+i} = diag([0, 0, 1, 1])$ $\forall i = [1, N_p - 1]$ for penalising the state errors, and $r_{k+i} = 0.1$ $\forall i = [0, N_p - 1]$ for penalising input errors. Moreover, a terminal cost of $q_{k+N_p} = diag([0, 0, 500, 500])$ was used as a "soft" zero-terminal constraint by penalising heavily the terminal "outputs/states" (angle and position) to improve the stability characteristics of the optimisation. Finally, the constraints of the system were imposed in the input and the position as $-10 \leq u \leq 10$ (m/s^2) and $-1 \leq p \leq 1$ (m), respectively.

Shifting Strategy Example

Let us begin by giving comprehensive visualisation of the proposed shifting strategy applied to this system as depicted in figure 5.8 where the predicted trajectories of three subsequent optimisation problems are presented in the *relative time frame* when using a block size of $N_b = 4$ starting from time-step $N_{b_0} = 1$ of definition 5.1. From this figure, it is noticeable how the shooting points at times approximately $0.2 \leq t \leq 0.4$ (s) are kept precisely at the constraint limits, with the intermediate/ignored steps practically satisfying the constraints despite them not being included in the optimisation. Moreover, it is interesting to note how most of the time the shooting points and the blocked inputs are moving left horizontally, indicating that the resulting optimisation at all three subsequent time-steps were nearly identical in the *absolute time-frame*.

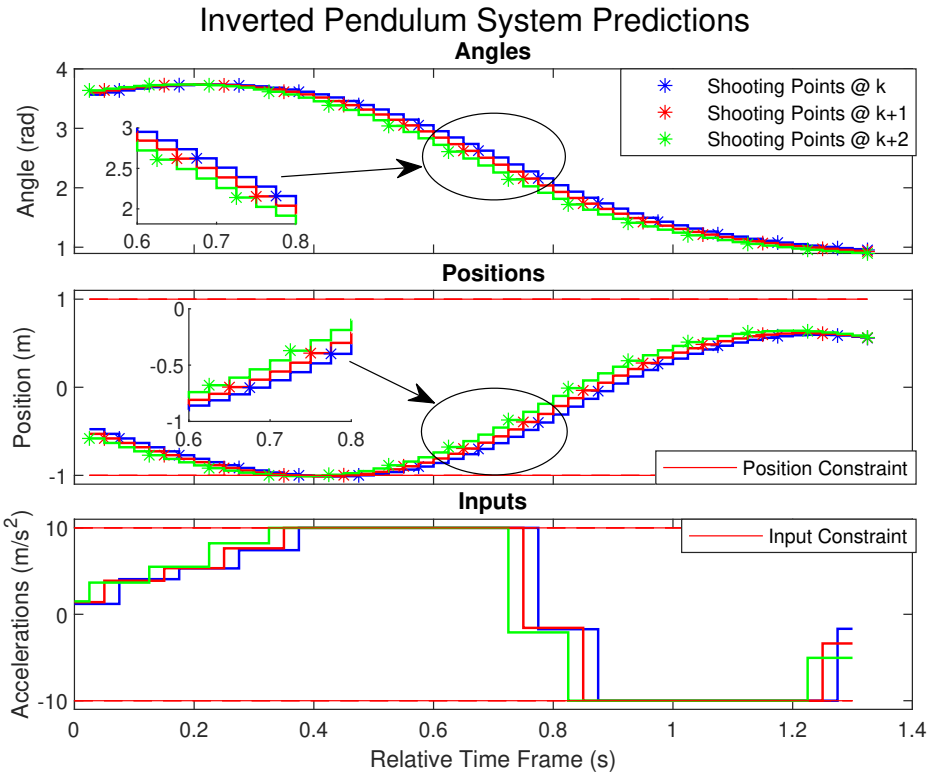


Figure 5.8: Shifting Example with $N_b = 4$ starting from $N_{b_0} = 1$ of Definition 5.1

Performance Comparison

On the other hand, to assess the performance and recursive feasibility properties of the proposed approach, the system was tested with four possible types of solution, namely: using the relative/non-relative input parameterised frameworks (4.4 and 4.5), with/without the proposed shifting strategy. Each of these approaches was tested for different block sizes ($N_b = [1 \rightarrow 12]$) with their corresponding Ideal Prediction Horizon as discussed earlier. The resulting optimisation of each case was solved using "quadprog" function of Matlab R2018a with the interior point method. Moreover, given the miss-alignment with the blocked decision variables discussed in remark 5.1, the relative/non-shifted solution kept the constraints in the full sized vector (no selection) which allowed the demonstration of

the resulting in-feasibilities discussed in the *lock-in condition* example 5.3. To demonstrate this more precisely, every time the "quadprog" function returned an in-feasibility flag on any of the methods, it was counted and the previous solution was used explicitly, essentially representing open-loop control (no feedback) as discussed in the aforementioned example.

Table 5.10 gathers the comparison of the costs for all the different types of solutions where $J_{Rel-Shift}$ represents the cost of the relative based solution with the proposed shifting strategy, $J_{Non-Rel-Shift}$ the cost of the non-relative based solution, and so on. For reference, costs less than 2000 swung up the system in "one shot", costs greater than 2000 but less than 3000 swung up the system in "two shots", and costs above 3000 means the optimisation was not able to stabilise the system at all (see green line of figure 5.9). The number of resulting in-feasibilities of each type of solution is reported in brackets the table. The following summarises the main results:

1. The results from using the relative/non-relative frameworks with the proposed shifting strategy are exactly the same (columns 1-2 are equal), as expected from the discussion of remark 5.1. This is a direct result of what has been said repeatedly throughout the chapter: *consistency*. Moreover, no in-feasibilities were recorded for both types of solutions of the proposed approach.

2. The solution based on the relative framework without shifting (J_{Rel}) presented a large number of in-feasibilities (239 total), most likely due to the "lock-in condition" discussed in example 5.3.

3. Overall, the best performance were given by the proposed shifting strategy as seen from the "Total" costs. In contrast, the non-relative/non-shifted formulation gave the worse results.

4. For block sizes $N_b = [10, 11]$, non of the solutions was able to swing up the system in "one shot". This is unsurprising and linked to the obvious observation that there may be more sensible block sizes for a specific situation/initial condition.

5. All the "one shot" solutions of the proposed approach resulted in suboptimalities of $\Delta J = (J_{N_b}/J_1 - 1) \times 100 < 13.26\%$ which give acceptable performance such as the ones given in both figures (5.9) and (5.10).

6. Notice block sizes $N_b = [2, 7]$ presented even better performance than the standard NMPC approach ($N_b = 1$). This is because in the linearisation process, the optimisation might take a different "branch" of the solution that improved further AFTER re-linearisation, as discussed in the case study of the Van-Der Poll Oscillator 5.3. Moreover, allowing the intermediate constraints to be violated may relax the solution and lead to better performance at the cost of having intermediate steps violations. Finally, the optimisation is done in a finite horizon where both block sizes have slightly longer prediction horizon which could result in better overall predictions.

Figure 5.9 shows an example response with block size $N_b = 6$ where it can be seen that both solutions using the proposed shifting strategy, namely: the relative/non-relative, depicted by the blue and red dashed lines, were able to swing up and stabilise the system in "one shot", with both giving the exact same result using. In contrast, it took "two shots" for the non-shifted relative-based solution, depicted by the cyan dotted line, and the optimisation failing completely in the case of the non-shifted non-relative solution, depicted by the green dot-dashed line.

N_b (N_p)	$J_{Rel-Shift}$	$J_{Non-Rel-Shift}$	J_{Rel}	$J_{Non-Rel}$
1 (50)	1101	1101	1101	1101
2 (51)	1099	1099	1105 [17]	1119[8]
3 (52)	1104	1104	2732 [18]	1239
4 (53)	1138	1138	2758 [32]	2475
5 (51)	1194	1194	1224 [14]	4118
6 (55)	1168	1168	2436 [18]	4002
7 (57)	1098	1098	2438 [26]	2197
8 (57)	1183	1183	2533 [38]	2173
9 (55)	1207	1207	2361 [20]	3514
10 (51)	2524	2524	2544 [27]	3475
11 (56)	2776	2776	2284 [15]	3448
12 (61)	1244	1244	4432 [14]	3397
Total	16835	16835	27948 [239]	32258 [8]

Table 5.10: Cost comparison for different block sizes N_b using Shifting and Non-Shifting Strategies

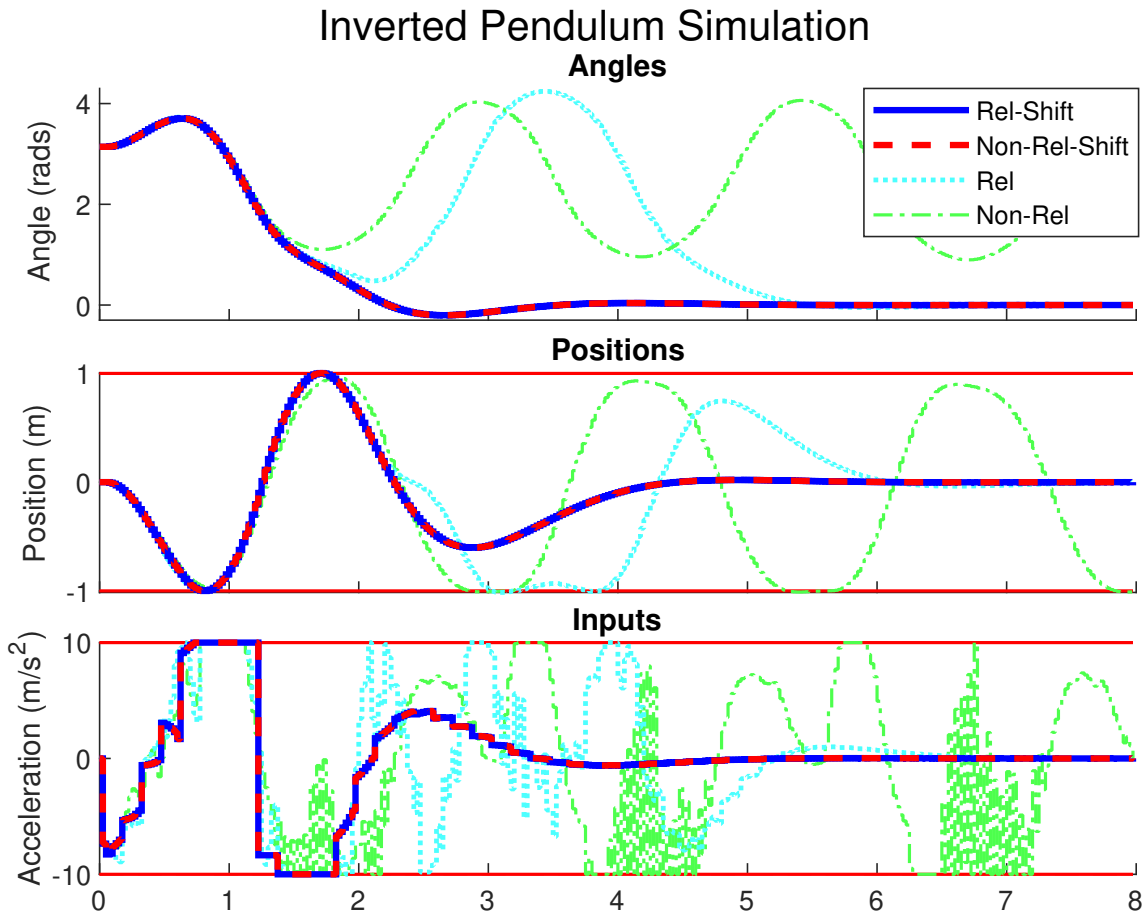


Figure 5.9: Example Performance Comparison with $N_b = 6$

Violations Comparison

Another value of interest that was compared was the summation of the absolute violation to the position constraints for different block sizes (N_b). In other words:

$$\sum_{\forall k} v_k \qquad \text{where} \qquad v_k = \begin{cases} |p_k| - 1 \ \text{ if } \ |p_k| > 1 \\ 0 \ \text{ else} \end{cases} \tag{5.57}$$

The results of this are gathered in table 5.11 where $V_{T-Shift}$ represents the total violation of the proposed shifting strategy (relative and non-relative being the same), and V_{Rel} and $V_{Non-Rel}$ the total violation of the relative and non-relative non-shifting solutions, respectively. Additionally, given that the proposed shifting strategy is only supposed to enforce the constraints in the shooting points, the summation of the constraint violation at the selected "shifted shooting points (Shifted-SP)" was stored separately and is represented by $V_{Shifted-SP}$ in the table. From this table we can clearly see how the proposed approach resulted in no violations of the constraints at the selected points ($V_{Shifted-SP} = 0$) when using the proposed strategy with all block sizes $N_b = [1 \rightarrow 12]$. Moreover, although the non-relative non-shifted solution ($V_{Non-Rel}$) did not present as much violations as the total shifted solution ($V_{T-Shift}$), it resulted in significantly degraded performance as seen in figure 5.9 and table 5.10.

N_b (N_p)	$V_{T-Shift}$	$V_{Shifted-SP}$	V_{Rel}	$V_{Non-Rel}$
1 (50)	0	0	0	0
2 (51)	0.002	0	0.004	0.006
3 (52)	0.009	0	0.017	0
4 (53)	0.002	0	0.064	0
5 (51)	0	0	0.044	0.106
6 (55)	0.003	0	0.094	0.178
7 (57)	0.171	0	0.300	0
8 (57)	0.114	0	0.423	0
9 (55)	0	0	0.406	0.059
10 (51)	0.032	0	0.686	0
11 (56)	0.069	0	0.313	0
12 (61)	1.055	0	0.986	0
Total	1.457	0	3.337	0.349

Table 5.11: Constraint violation comparison for different block sizes N_b, with and without Shifting

Finally, to illustrate the concept of satisfying the constraints in the selected shooting points, figure 5.10 shows the response of the system with block size $N_b = 12$ where it can clearly be seen that the solution satisfies the constraints of the selected shooting points at the very limits as signaled in the inner graph/ellipse, which in this case are at times $t = [0.925, 1.225] = [3N_B + 1, 4N_B + 1]T$. The "extra step" in both shooting points is due to the computation separation strategy of the RTI which uses a predicted state, thus always optimising relative to "one step ahead" and applying the feedback phase in the next sampling time when the measurement of the state is available. This can clearly be seen in the input response where the first decision is at $t = 0.025$ (s) instead of $t = 0$ (s), as seen in the inner graph. Another important thing to notice is that the solution clearly exhibits the blocking structure, in particular, after $t > 3$ when the system is stabilised within the prediction horizon.

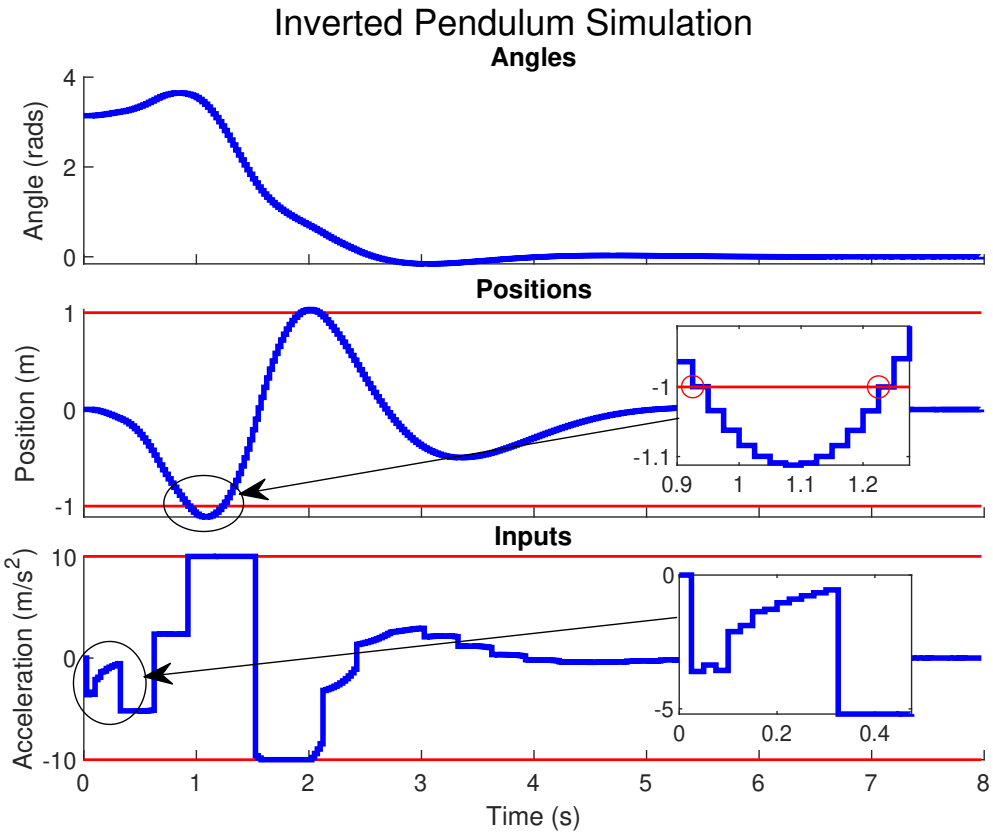


Figure 5.10: Constraint Satisfaction Example for Shifting Strategy with $N_b = 12$

5.5.3 Computation Times Comparison

In order to evaluate the computational performance of the proposed approach in this system, we developed a set of auto-generated C++ codes based on the Eigen 3 library using the RTI algorithms 5.6 and 5.7 which implemented the approach in the Inverted Pendulum system. For normalisation and performance related reasons, three main differences were implemented in this particular test, namely: a different model parameter ($k = 10$), different input constraints ($-1 \leq u \leq 1$ (m/s^2)) and different weights ($q_{k+i} = diag([0.01, 0.01, 1, 1])$ $\forall i = [1, N_p - 1]$, $q_{k+N_p} = 10q_{k+1}$, $r_{k+i} = 10$ $\forall i = [0, N_p - 1]$). Moreover, the simulation used the same initial condition at the lower equilibrium, however, in this case, the reference was introduced at the end of the prediction horizon to allow for a smooth transition between the two operating points, thus satisfying common requirements of the RTI Scheme as discussed in [55]. Furthermore, to evaluate a more complete version of the approach, the optimisation was done using the multiple-shooting approach instead. The algorithms were automatically generated for different block sizes $N_b = [1 \rightarrow 6]$ using an Ideal Prediction Horizon $N_p = 121$. For comparison, the solution with the standard RTI algorithms 3.9 and 3.8, as well as via the ACADO toolkit was also obtained. The solutions of the resulting QPs were obtained using QP OASES [37, 38], all of which were verified to match in all cases from Matlab simulations, to developed C++ codes, to the ACADO toolkit. Finally, the algorithms were tested both with and without applying theorem 5.2 where the Lagrange multipliers were shifted either correctly, every time the absolute time-step (N_{b_0}) reached the limit, or incorrectly by shifting them at every time step as in the standard IVE.

Each of the aforementioned cases was run for 1000 simulations of $T = 10\ (s)$ giving a total of $400,000$ optimisations per case. The codes were run using the same conditions as in the generic computations comparison of section 5.2.6, ie. same laptop running the codes with real-time priority (ie. chrt -r 99 ./main) with the same optimisation flags (-O3,-mavx,-mfma) for the compilation. The resulting average computation times of the constrained iterations of each of these approaches is presented in table 5.12. For reference, the resulting gain factors are indicated in the red under-scripts.

Solution/N_b	ACADO	Std	1	2	3	4	5	6
QP OASES (with thrm. 5.2)	2040	2000	2014	$312_{6.4}$	$114_{17.7}$	$64_{31.5}$	42_{48}	37_{54}
QP OASES (without thrm. 5.2)	-	-	-	337_{6}	$136_{14.8}$	$79_{25.5}$	$57_{35.3}$	$44_{45.7}$

Table 5.12: Average constrained computation times in μs for the Inverted Pendulum using different block sizes $N_b = [1 \rightarrow 6]$ with Ideal Prediction Horizon $N_p = 121$. The gain factor is indicated in red.

From this table (table 5.12), it can clearly be appreciated how the proposed approach resulted in substantial computational gains, being up to 54 times faster than the Standard approach when using $N_b = 6$, increasing rapidly as the block-size increases. Moreover, it can be appreciated how the approach resulted in slightly faster computation times when using the proposed theorem 5.2 for shifting the resulting Lagrange multipliers. Although this may be a small difference, it does indicate the method has improved hot-starting capabilities when using proposed method of theorem 5.2.

On the other hand, note that the method presents a very slight increase in the computation time obtained with a block size of $N_b = 1$ when compared to the standard NMPC solution. This is inevitably related to the additional computations of the conditional "if-then-else" statements required by most of the algorithms introduced in section 5.2 for detection/tracking of the required block position and count. This however is of minimum impact when considering the gains obtained with any block size $N_b > 1$. Finally, it is noteworthy to mention that the solution obtained with QP OASES resulted in slightly higher computation times, even when compared with the Standard NMPC solution obtained via the auto-generation routines based on the Eigen 3 library developed as part of this Ph.D. thesis. Recalling that all of the evaluated methods were validated by comparing the results, eg. comparing ACADO with both Matlab simulations and with auto-generated C++ codes, we take this as an opportunity to demonstrate that the approaches of this Ph.D. thesis, including the multiple-shooting framework presented in chapter 3, along with the implementation of the $O(N)/O(N^2)$, and its required extensions for the proposed approach were all implemented correctly and are absolutely essential for obtaining an efficient real-time solution.

Thus, this comparison presents an example application in which the proposed approach was observed to result in significantly reduced computation times. Moreover, the approach demonstrated excellent closed loop performance and was observed to maintain the expected recursive feasibility properties on the selected constraints of the optimisation, as per design of the proposed Shifting Strategy. All of this features combined give an indication that the proposed approach may be a key method for implementing advanced NMPC strategies in real-time control systems.

5.6 Case Study: The Obstacle Avoidance Problem

The last case study that was considered for implementation of the proposed approach was on the topic of obstacle avoidance, which is a powerful application of MPC methods given its innate ability to anticipate and take corrective actions for future events, eg. collisions. In this case, we were interested in its application in Quadrotors given their requirement for real-time solutions with reduced computational resources which makes them a perfect candidate. However, the method of obstacle avoidance that we will present in this case study can equally be applied to similar systems such as autonomous aircrafts, cars, trucks, or even satellites, given it is based on a "high-level" optimisation approach which focuses on defining trajectories to be followed by the inner control systems through the use of the so called "virtual-forces" used in "control-allocation" schemes, as discussed in [72]. Moreover, although there exist a wide range of nonlinear controllers which can be used for the inner control systems of Quadrotors such as nonlinear dynamic inversion [22, 27], differential-flatness based control [35, 119], and so on, this particular case study uses a discrete back-stepping controlled based on the methodologies described in [96, 151] which will be introduced briefly without proof on section 5.6.3 as the derivation of the required control laws are outside the scope of this thesis. Finally, this case study will use the alternative Delta blocking structure of figure 5.3a, along with alternative shifting shooting points strategies as described in tables 5.1 and 5.2 to provide an example of the potential for extension of this chapter's contribution.

5.6.1 UAV Modelling

Let us begin by considering the positioning model a Quadrotor of unitary mass ($m = 1$) under low-speed/negligible drag scenarios, given by:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} Z(2q_0q_2 + 2q_1q_3) \\ -Z(2q_0q_1 - 2q_2q_3) \\ Z(q_0^2 - q_1^2 - q_2^2 + q_3^2) - g_0 \end{bmatrix} \tag{5.58}$$

where Z is the overall thrust of the system, g_0 is the gravity constant, and $\overrightarrow{q} = \begin{bmatrix} q_0, q_1, q_2, q_3 \end{bmatrix}$ is a unitary quaternion vector that defines the orientation of the vehicle.

Likewise, the Quaternion dynamics, defining the attitude/orientation of the vehicle are given by:

$$\dot{\overrightarrow{q}} = \frac{1}{2} E(\overrightarrow{q})\overrightarrow{\omega} \tag{5.59a}$$

$$s.t \quad \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1 \tag{5.59b}$$

where $\overrightarrow{\omega} = \begin{bmatrix} \omega_x, \omega_y, \omega_z \end{bmatrix}$ is the angular velocity vector in body-axis, which for the simulations of this case study was limited to $||\overrightarrow{\omega}|| \leq 300$ (deg/s), and where $E(\overrightarrow{q})$ is defined as:

$$E(\overrightarrow{q}) = \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{bmatrix} \tag{5.60}$$

Finally, we consider the angular velocity has a generic model given by:

$$\dot{\overrightarrow{\omega}} = f(\overrightarrow{\omega}, \omega^2_{motors}) \tag{5.61}$$

which is typically defined using the inertias of the vehicle, the thrust/torque constants, motor arrangement (depending on the type of vehicle eg. Hexacopters), etc. However, this case study will consider the angular acceleration vector as the "lowest" control signal limited $||\dot{\overrightarrow{\omega}}|| < 2500(deg/s^2)$ with the understanding that a lower level controller could be easily developed. This will allow us to focus the attention on the high-level planning of the controller.

In order to simulate the system, the positioning model (5.58) was discretised using a 2nd order Taylor Polynomial series expansion, given by:

$$\begin{bmatrix} x \\ v_x \\ y \\ v_y \\ z \\ v_z \end{bmatrix}_{k+1} = \begin{bmatrix} x + T_s v_x \\ v_x \\ y + T_s v_y \\ v_y \\ z + T_s v_z \\ v_z \end{bmatrix}_k + \begin{bmatrix} \frac{T_s}{2} & 0 & 0 \\ T_s & 0 & 0 \\ 0 & -\frac{T_s}{2} & 0 \\ 0 & -T_s & 0 \\ 0 & 0 & \frac{T_s}{2} \\ 0 & 0 & T_s \end{bmatrix} (\overrightarrow{F}_{k+1} - \overrightarrow{F}_G) \tag{5.62}$$

where T_s is the sampling time, $\overrightarrow{F} = \begin{bmatrix} F_x, F_y, F_z \end{bmatrix}^T$ is the "virtual force" vector defined as:

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} Z(2q_0q_2 + 2q_1q_3) \\ -Z(2q_0q_1 - 2q_2q_3) \\ Z(q_0^2 - q_1^2 - q_2^2 + q_3^2) \end{bmatrix} \tag{5.63}$$

and $\overrightarrow{F}_G$ being the gravity vector $\overrightarrow{F}_G = [0, 0, g_0]$.

A very important assumption about this model is that the force vector ($\overrightarrow{F}$) is evaluated in step $(k+1)$ which could be seen as a "backward-forward" euler scheme. This was required by the discrete backstepping low-level controller presented in section 5.6.3 to be able to have the model in "strict feedback form" [96], which will allow the trajectories of the high-level NMPC planner to be tracked "perfectly" when ever possible in the nominal case. This again will allow us to focus on the relevant properties of the proposed approach. Moreover, given a fast sampling time of $T_s = 0.02$ (s) was chosen for the simulation, this model could approximate the real trajectories with relatively small error.

On the other hand, the Quaternion dynamics were discretised for the simulation by:

$$\overrightarrow{q}_{k+1} = \overrightarrow{q}_k + B_q \overrightarrow{\omega}_{k+1} \tag{5.64}$$

where $B_q = \frac{T_s}{2} E(\overrightarrow{q}_k)$, T_s being the sampling time as before.

Once again, notice the angular velocity vector ($\overrightarrow{\omega}$) is evaluated in step $(k+1)$. However, in this case, this model actually represents the standard backward Euler approach which has been used for developing Extended Kalman Filters for Quaternion estimation [126] which take the "latest" gyroscope measurement ($\overrightarrow{\omega}_k$) to simulate and correct the "latest" Quaternion ($\overrightarrow{q}_k$) using the "latest" accelerometer measurement ($\overrightarrow{a}_k$). Thus, this discrete model is justified.

Lastly, the generic angular velocity model was discretised via a simple forward integration scheme, given by:

$$\overrightarrow{\omega}_{k+1} = \overrightarrow{\omega}_k + B_\omega \dot{\overrightarrow{\omega}}_k \tag{5.65}$$

where $B_\omega = T_s I_{3x3}$, T_s being the sampling time, and $\dot{\overrightarrow{\omega}}_k$ being the lowest-level decision variable which will be taken into account for the simulations presented in this case study.

Note that because of the required use of values at $k+1$ steps, the discrete models must be used in a specific order to obtain the simulation, ie. $\dot{\overrightarrow{\omega}}_k \rightarrow \overrightarrow{\omega}_{k+1} \rightarrow \overrightarrow{q}_{k+1} \rightarrow \overrightarrow{F}_{k+1} \rightarrow [x, v_x, y, v_y, z, v_z]^T_{k+1}$. This again was specifically required by the discrete back-stepping procedure to allow us to obtain "perfect" tracking in the nominal case whenever possible. We will discuss this further in subsection 5.6.3 where the backstepping procedure will be introduced.

5.6.2 High-Level NMPC Planner

One of the methods that is being used to reduce the complexity of the NMPC is the use of virtual variables, in this case forces, in the same way Control Allocation is used for distributing the control signals to the motors on low-level controllers of Quadrotors as discussed in [72]. This is not an uncommon strategy for Quadrotors, and in fact, standard flight controllers such as "Ardupilot" use this kind of strategy for their control loops, although they don't use it in the same way, eg. using normal PIDs and Transformation matrices, rather than combining NMPC and backstepping together with Control Allocation concepts. The reason why they are used is in this case study is because it allows us to simplify the control system by using modularity rather having one single master control law. Moreover, in this particular case, it also allows the use of linear model (5.62) relating the forces present in the system with the positional variables which are then handled by using control allocation [72] and/or transformation matrices. The use of this linear model (5.62) in the context of NMPC framework would allow some of the required matrices to be time-invariant, which in practice could prove to be key for reducing the computational burden further.

The task of the NMPC High Planner is then to design trajectories using the virtual force vectors ($\overrightarrow{F}$) as decision variables which must then be transformed into references for the low-level controllers. To achieve this transformation, let us begin by noting that we can express the force vector in terms of Euler angles (without singularities) using the Direction Cosine Matrix (DCM) as:

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} Z(\sin\Phi\sin\Psi + \cos\Phi\sin\Theta\cos\Psi) \\ Z(-\sin\Phi\cos\Psi + \cos\Phi\sin\Theta\sin\Psi) \\ Z(\cos\Phi\cos\Theta) \end{bmatrix} \tag{5.66}$$

where Z is the thrust, Φ is the roll angle, Θ is the pitch angle, and Ψ is the yaw angle.

Looking at these equations, it can be clearly observed there are 4 independent variables that can be used to control the 3 variables of the force vector, which clearly wouldn't have a unique solution. As an example, for any yaw-heading (Ψ) there is a combination of roll (Φ), pitch (Θ) and thrust (Z) that would give the required force vector.

To avoid this "non-uniqueness" problem, a simple solution is to fix the yaw-heading (Ψ) to a reference value (Ψ_{ref}), and sort out what are the other variables required to achieve the force vector. Moreover, because the motors of a Quadrotor typically can only produce thrust effectively in one direction (unless, eg. variable pitch propellers are used), the thrust vector (Z) is constrained/saturated to be positive and limited, eg. less than 90% of maximum thrust (Z_{max}) to allow for orientation movements at all times. Based on this, the control allocation solution can be obtained as follows:

Noting that the total summation of forces must be equal the thrust vector:

$$Z_{ref} = \sqrt{F_{x_{ref}}^2 + F_{y_{ref}}^2 + F_{z_{ref}}} \tag{5.67a}$$

$$s.t. \quad 0 < ||Z_{ref}|| < 0.9 Z_{max} \tag{5.67b}$$

the following system of 2 equations can be formed from the equations of F_x and F_y as:

$$\begin{bmatrix} \frac{F_{x_{ref}}}{Z_{ref}} \\ \frac{F_{y_{ref}}}{Z_{ref}} \end{bmatrix} = \begin{bmatrix} \sin\Psi_{ref} & \cos\Psi_{ref} \\ -\cos\Psi_{ref} & \sin\Psi_{ref} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{5.68}$$

After solving for α and β, the Roll (Φ) and Pitch (Θ) reference angles are defined as:

$$\Phi_{ref} = \sin^{-1}(\alpha) \tag{5.69a}$$

$$\Theta_{ref} = \sin^{-1}\left(\frac{\beta}{\cos\Phi}\right) \tag{5.69b}$$

This angles are then transformed to the equivalent Quaternion and given as a reference to the back-stepping controller presented in section 5.6.3.

Smoother Delta Blocking Trajectories

When implementing this type of control allocation, it is desirable to penalise the force increments ($\Delta\overrightarrow{F}_k = \overrightarrow{F}_k - \overrightarrow{F}_{k-1}$) which would result in smooth trajectories on the required references angles, thus improving the feasibility properties of the optimised trajectories in the low-level controllers. Moreover, given the approach of this case study will be applying the general concept of "blocking" to reduce the degrees of freedom of the optimisation, it is more reasonable to have the increment of the force vector ($\Delta\overrightarrow{F}_k$) as the blocked decision variable, rather than the force vector itself. A simple way to explain this is to consider blocking the force vector as having a desired thrust/orientation for some time (where the time is defined by the block size N_b), and then INSTANTANEOUSLY changing to another force/orientation without taking into account any previous force or orientation. In contrast, by optimising using the increment of the force, ie. the force derivative, results in smoother control actions where the reference thrust and orientation of the control allocation is always "connected" with its previous value in some form, which in general was seen to improve the feasibility of the virtual forces to be achieved by the low-level discrete back-stepping controller of section 5.6.3. Note that this can be done simply by augmenting position dynamics model (5.62) with the force vector ($\overrightarrow{F}_{k+1}$) and embedding the Ideal Delta Moving Window Blocking approach ($\Delta\hat{U} = \mathbb{N}_{N_{b_0}}\Delta\hat{\mathbb{U}}$) in the force increments as depicted in figure 5.3a.

An example of the resulting smoothness that can be obtained with the Ideal Delta MWB approach when using a block size of $N_b = 6$ and an Ideal Prediction Horizon of $N_p = 79$ for the obstacle avoidance simulation conditions provided later in section 5.6.5 is given in figure 5.11 where the trajectory of the Y axis required to successfully avoid an obstacle is presented. For comparison, this figure also presents the solution using two other type of optimisations, namely: the Standard NMPC solution, depicted by the red dashed line; and the GPC solution with the same decision variables ($N_u = 14$), depicted by the green dot-dashed line, similar to the comparison of figure 5.1. From this it can be appreciated how the Ideal Delta MWB approach results in nearly indistinguishable trajectories in the force and position when compared to the standard NMPC approach, whereas the GPC approach clearly results in completely different trajectories. Thus, the Obstacle Avoidance problem gives another example application where having decision variables spread around the horizon results in much better performance when compared to having them congested at the beginning of the horizon as in GPC.

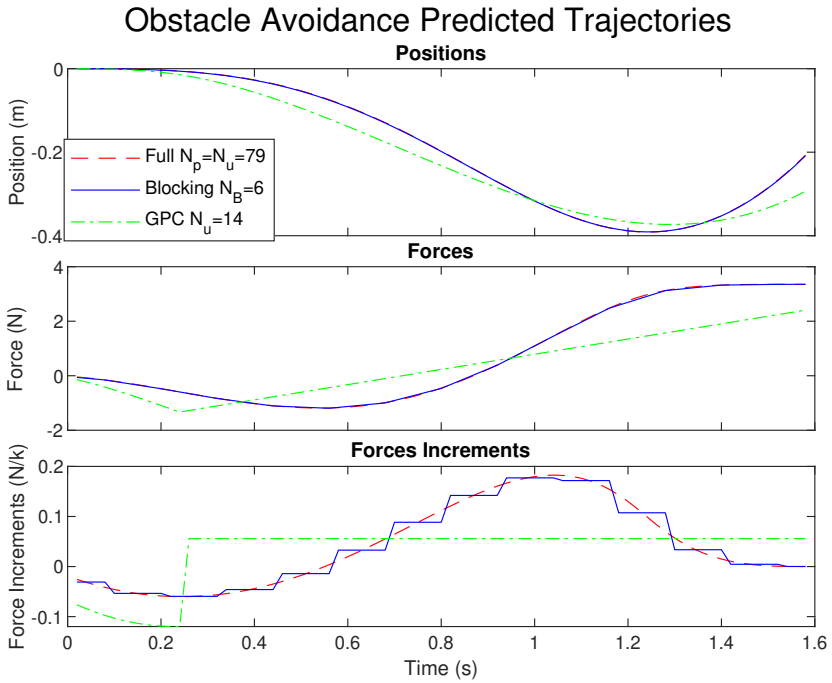


Figure 5.11: Delta Blocking vs GPC Comparison in the Obstacle Avoidance Problem with $N_b = 6$

5.6.3 Discrete Back-stepping Control

Back-stepping control is among the most popular nonlinear control methods for Quadrotors given its simplicity and its adequacy for the models and control architectures that are typically used, as well as its Lyapunov-derived stability guarantees [97, 139]. The derivation of the control laws can be done both in continuous time and discrete time, however, for our simulation we choose to use it in discrete time given its excellent tracking properties which would allow us, in an ideal nominal-case scenario, to track the desired virtual control signals of the high-level planner with zero steady state error, thus allowing us to focus on the expected performance and recursive feasibility properties of the proposed approach. Note that in this case, the high-level NMPC planner is essentially optimising the first (or highest) "virtual control signal" or references of the general back-stepping procedure.

The back-stepping procedure used for this simulation consists of only 2 steps; the first step controlling the Quaternion to a desired reference using the angular velocity vector as a virtual signal; and the second step controlling the angular velocity vector using angular acceleration as an actual control vector. The derivation of the control laws is out of the scope of this thesis but can be obtained following the methodology given in [151]. The procedure is given by:

Step 1: Stabilise the Quaternion using Angular Velocities

$$\overrightarrow{\omega}_{k+1_{ref}} = B_q^{-1}\left(\overrightarrow{q}_{k+1_{ref}} - \overrightarrow{q}_k - (I - K_q)(\overrightarrow{q}_{k_{ref}} - \overrightarrow{q}_k)\right) \tag{5.70}$$

where $B_q^{-1} = (B_q^T B_q)^{-1} B_q^T$ and $K_q = \alpha_q I_{4x4}$ is a diagonal tuning matrix with $0 \leq \alpha_q \leq$ where $\alpha_q = 0$ would indicate "no correction", basically leading to an open-loop feed-forward control law, and $\alpha_q = 1$ would result in an "instantaneous correction" leading to a potentially unstable one-step ahead dynamic inversion control law.

Step 2: Stabilise the Angular Velocities using Angular Accelerations

$$\dot{\overrightarrow{\omega}}_k = B_\omega^{-1}\left(\overrightarrow{\omega}_{k+1_{ref}} - \overrightarrow{\omega}_k - (I - K_\omega)(\overrightarrow{\omega}_{k_{ref}} - \overrightarrow{\omega}_k)\right) \tag{5.71}$$

where $K_\omega = \alpha_\omega I_{3x3}$ is again a diagonal tuning matrix where the tuning variable $0 \leq \alpha_\omega \leq 1$ follows the exact same behaviour as described for step 1.

Note that both steps require the references given at two steps (k and $k+1$). In essence, if the system reached the desired reference for the current step, eg. $q_{k_{ref}} - q_k = 0$ or $\omega_{k_{ref}} - \omega_k = 0$, then the solution is a simple one-step ahead dynamic inversion. Otherwise, it applies a correction which slows the one-step dynamic inversion into a more smooth trajectory.

An important thing to mention is that both virtual control signals norm $\left|\left|\overrightarrow{\omega}_{k+1_{ref}}\right|\right| < 300\ (deg/s)$ and $\left|\left|\dot{\overrightarrow{\omega}}_k\right|\right| < 2500\ (deg/s^2)$ were always saturated to be within the constraint values for the simulation. Whenever this happened, the Quadrotor went through a short transition phase before recovering track of the requested trajectory. An example of this behaviour is given in figure 5.12 where the excellent tracking performance of the resulting discrete back-stepping controller can be seen, and the approach can be seen to recover quickly from a momentary thrust saturation, as signaled by the inner elipses.

5.6.4 Dynamic Obstacles and Nonlinear Constraints

Lastly, we have the topic of obstacle avoidance itself. In this case we are interested in dynamic obstacles represented by "spheres" which follow 3D parabolic trajectories subject to gravity determined by the discrete position dynamics of (5.62), with the only difference of not being "actuated" by the force vector. Although there exist some different variants on how to achieve obstacle avoidance for this scenario, eg. using potential fields as in [1, 17, 106], this case study implemented the nonlinear constraints approach as in [57]. Considering the Quadrotor could also be represented by a sphere, the objective is then to maintain a minimum squared distance to N_{obs} obstacles, eg. by imposing the nonlinear constraints:

$$d_{k,i}^2 = (x_k - x_{k,i})^2 + (y_k - y_{k,i})^2 + (z_k - z_{k,i})^2 \geq d_{min}^2 \tag{5.72}$$
$$\forall k = [1, N_p] \qquad \forall i = [1, N_{obs}]$$

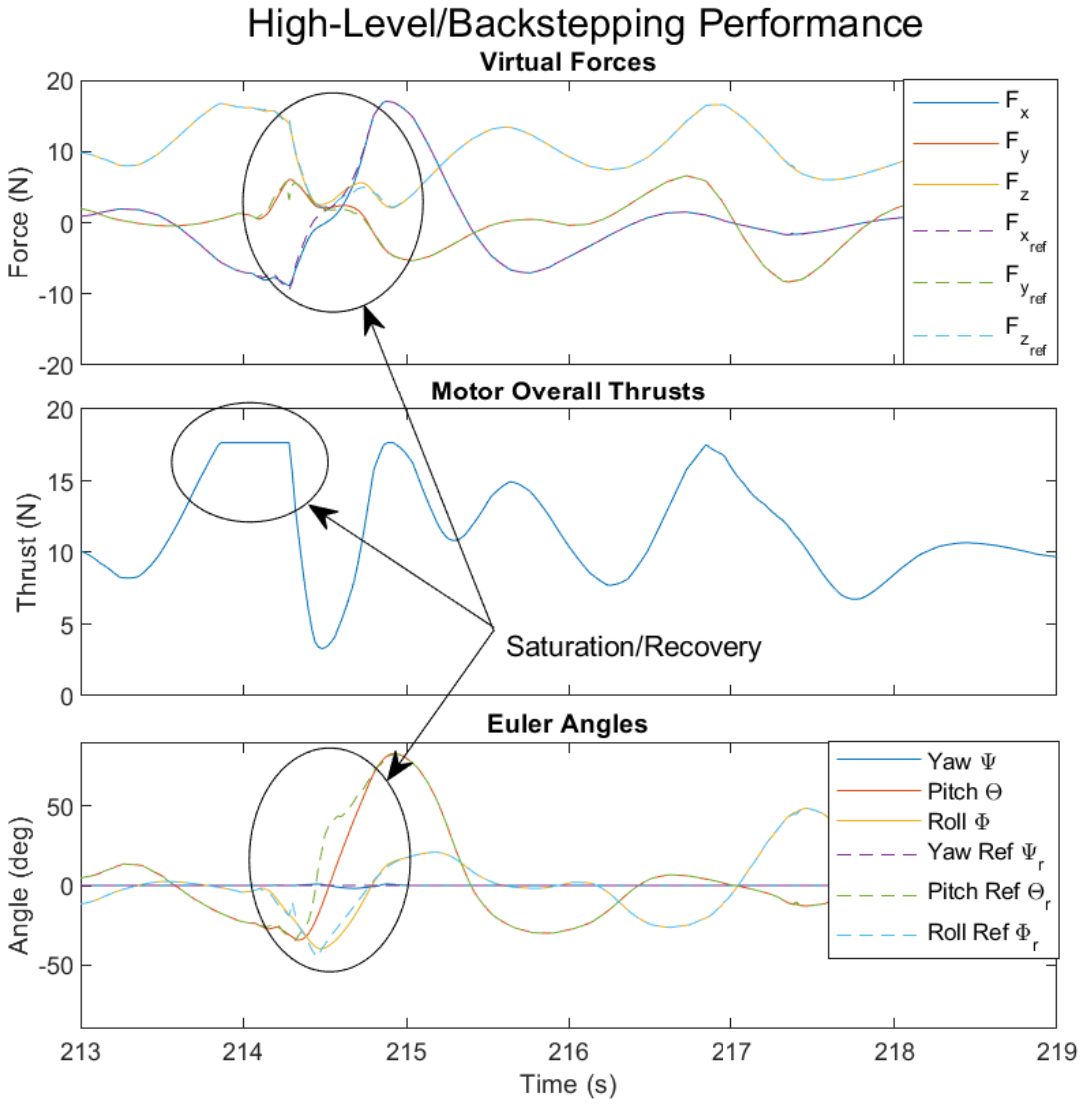


Figure 5.12: Example Discrete Back-stepping Tracking Performance with Recovery

These nonlinear constraints (5.72) can be imposed into the cost function simply by forming the N_{obs} linearised output prediction models of the squared distances to each obstacle, eg. given by:

$$\hat{Y}^i_{obs} = \bar{Y}^i_{obs} + G^i_{obs}\delta x_0 + H^i_{obs}\mathbb{N}_{N_{b_0}}\Delta\hat{\mathbb{U}} \qquad \forall i = [1, N_{obs}] \tag{5.73}$$

and including them into the optimisation inequality constraints as:

$$M_{\mathbb{N}}\Delta\hat{\mathbb{U}} = \begin{bmatrix} -H^1_{obs}\mathbb{N}_{N_{b_0}} \\ \vdots \\ -H^{N_{obs}}_{obs}\mathbb{N}_{N_{b_0}} \end{bmatrix} \Delta\hat{\mathbb{U}} \leq \begin{bmatrix} -(D^2_{min} - \bar{Y}^1_{obs} - G^1_{obs}\delta x_0) \\ \vdots \\ -(D^2_{min} - \bar{Y}^{N_{obs}}_{obs} - G^{N_{obs}}_{obs}\delta x_0) \end{bmatrix} = \gamma \tag{5.74}$$

The shifting strategy approach would then select a subset of these constraints to be included in the optimisation.

5.6.5 Performance Comparison

In order to evaluate the performance of the proposed approach applied to this problem, we performed a series of simulations for different optimisation versions, including: block sizes $N_b = [1 \rightarrow 6]$, and apart from the basic shooting point selection of $N_{con} = 1$ constraint selected at the end of each block, alternative shooting point selection methods were implemented where the optimisation selected $N_{con} = 2$ constraints per block, similar to those described in tables 5.1 and 5.2. Following the same notation as in the aforementioned tables, the alternative $N_{con} = 2$ selection patterns were: $1-1$ for $N_b = 2$ (fully constrained); $0-1-1$ for $N_b = 3$; $0-1-0-1$ for $N_b = 4$; $0-0-1-0-1$ for $N_b = 5$; and $0-0-1-0-0-1$ for $N_b = 6$. These alternative cases will be referred to as $N_{con} = 2$. Each of this cases were simulated for different number of obstacles $N_{obs} = [1, 3, 5, 7, 10]$ simultaneously being thrown to the vehicle. Two performance metrics were captured from these simulations, namely: overall cost and collision probability, which will be discussed later in table 5.13.

The conditions of the simulation are quite involved, but can be briefly summarised as follows. All the simulations were done for a total of $T = 400$ (s) with the Quadrotor starting at the origin in steady state. A sampling time of $T_s = 0.02$ (s) was used for all the discrete models of section 5.6.1. Regarding the controller setup, a base prediction horizon of $N_p = 75$ $(T_p = 1.5)$ was used for the NMPC optimisation and was modified to select the Ideal Prediction Horizon depending on the block size (N_b) as in case study 5.5. The costs of the optimisation were selected as $q_{k+i} = diag([1, 0, 1, 0, 1, 0, 0, 0, 0])$ $\forall i = [1, N_p]$ for penalising the augmented state errors $[x, v_x, y, v_y, z, v_z, F_x, F_y, F_z]$, and $r_{k+i} = diag([1, 1, 1])$ $\forall i = [0, N_p - 1]$ for penalising the force increments $(\Delta\overrightarrow{F})$. The reference of the simulation was selected at the origin for half the simulation time, and was modified to a co-sinusoidal trajectory on all three axis for the rest of the simulation, with each axis reference selected as: $x_r = 2\sin(0.4\pi t)$, $y_r = 2\sin(0.4\pi t)$ and $z_r = 3\sin(0.2\pi t)$. The obstacle's trajectories were randomised based on the current and future vehicle references ($[x_r, y_r, z_r]_k$ and $[x_r, y_r, z_r]_{k+t_{imp}/T_s}$), considering random initial positions on a sphere of radius $5 < r_{obs-ini} < 25(m)$ relative to the position reference of the vehicle at the current time $[x_r, y_r, z_r]_k$, and random collision positions around the reference at a future time of impact $[x_r, y_r, z_r]_{k+t_{imp}/T_s} \pm 0.5$ (m), with the time of impact randomised between $(1.5 < t_{imp} < 2.5)$, allowing for statistical variability in the resulting trajectories of the obstacles. The time impact was selected such that the obstacles were smoothly introduced at the end of the prediction horizon of the optimisation, allowing for the IVE conditions of the RTI Scheme to be valid. The trajectory of each obstacle was "reset" (given a new trajectory) when its vertical position was $z_{obs} < -5$ (m) and its vertical velocity was $v_{z_{obs}} < 0$ (m/s), ie. falling and 5 (m) below the origin. The exact same obstacle trajectories were used for all the simulations where the total number of different obstacles thrown to the vehicle (after resetting) averaged on 160 obstacles per N_{obs}.

Regarding the nonlinear constraints, considering a common $m = 1$ (kg) Quadrotor frame has a spherical safe radius of about $r_{safe} = 0.45$ (m) (including 10in propeller tips), and considering spherical obstacles of radius $r_{obs} = 0.15$ (m) (similar to basket-balls) are thrown to the vehicle, all simulations were subject to two distance constraints; a "collision constraint" $d^2_{col} \geq 0.6^2$ which, if violated, indicated the obstacle collided with the vehicle; and a "warning constraint" using a fixed slack $d^2_{warn} = (0.6+0.2)^2$ (referred to as soft-constraint in figure 5.13) which indicated the obstacle got extremely close to the vehicle. This warning constraint (d^2_{warn}) was used as the constraint to be satisfied by the nonlinear

constraints (5.72) in order to allow for the small violations of the intermediate constraints which are ignored by the proposed Shifting Strategy. Given the obstacles were smoothly introduced at the end of the prediction horizon, the optimisation was solved using the Hildreth's QP algorithm from [146] with only 5 iterations per obstacle. Moreover, given the potential in-feasibility of the obstacles being thrown from all directions, the optimisation was augmented with a slack soft-constrained variable (ρ) added to the cost as $J_\rho = J + 100000\rho^2$, that was used to automatically adjust the selected constraints limits which allowed the optimisation to adjust its plan with small violations on the selected constraints in case the Quadrotor required manoeuvres which were too difficult to avoid the obstacles. This was observed to result in a more stable optimisation by always being able to find an optimal trajectory. Finally, the constants for the back-stepping controller were both selected as $\alpha_q = \alpha_\omega = 0.5$.

As it is understandably hard to picture the aforementioned conditions, a video showing an example animation can be found in (https://www.youtube.com/watch?v=Jsl6dRCSGNI) where the resulting performance with $N_b = 4$ and $N_{con} = 2$ can be seen.

Selected Nonlinear Constraints Satisfaction

One of the main properties of interest from the proposed Shifting Strategy is the satisfaction of the selected nonlinear constraints for the optimisation. This property was seen consistently throughout all the results that were analysed. To illustrate this, figure 5.13 shows an example performance of the distance maintained for up to $N_{obs} = 10$ obstacles when using the proposed approach with $N_b = 3$, $N_{con} = 1$. In this case, the proposed Shifting Strategy is expected to ignore 2 intermediate constraints, something which can be clearly appreciated from the signaled ellipses of the figure.

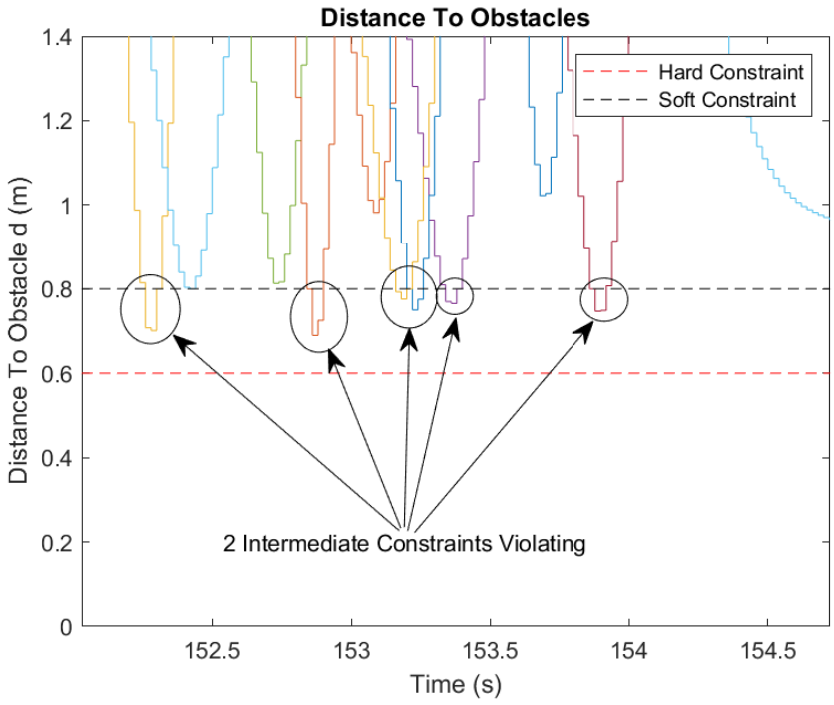


Figure 5.13: Example Distance to Obstacles when using $N_b = 3$ with $N_{con} = 1$ constraint per block

Costs and Collision Probability

As discussed earlier, the approach was tested using various number of obstacles $N_{obs} = [1, 3, 5, 7, 10]$, with various block sizes $N_b = [1 \rightarrow 6]$, and different number of constraints per block $N_{con} = [1, 2]$ for block sizes $N_b > 1$. The resulting performance with each of this variations was then measured in terms of resulting costs (J) and collision probability, where the latter was the calculated based on the number of times the solution violated the collision constraint $d_{col}^2 = 0.6^2$. Both of this measures are captured in table 5.13 where each case is clearly signaled in terms of Ideal Prediction Horizon, block size, number of constraints and number of obstacles. As it can be seen, in terms of the resulting costs, the proposed approach resulted in improved costs for the majority of the solutions when compared to the standard NMPC approach ($N_p = 75, N_b = 1, N_{con} = 1$), most likely given the relaxation of the intermediate constraints as well as the small increments in the Ideal Prediction Horizons that were used. However, this improvement comes at the cost of having a potential increase in the collision probability given the aforementioned relaxation. This can be seen clearly by observing that the solution ignoring the most constraints ($N_p = 79, N_b = 6, N_{con} = 1$) results in the best overall costs, but proves insufficient in successfully avoiding/detecting the obstacles with the reduced amount of selected constraints, thus having the highest collision probabilities for all the different number of obstacles tested. On the other hand, it can be appreciated how by using the alternative shifting constraints approach ($N_{con} = 2$) on block sizes $N_b > 2$, the optimisation fixes the collision probabilities resulting in as low as 0.1% (1 in 1000 obstacles colliding) when using a block size of $N_b = 5$ with $N_{con} = 2$ constraints per block.

Metric	Cost					Collision Probability				
Case/N_{obs}	1	3	5	7	10	1	3	5	7	10
$N_p = 75, N_b = 1, N_{con} = 1$	574	1354	2250	3243	4988	0	0	0	0	0
$N_p = 77, N_b = 2, N_{con} = 1$	527	1211	2171	2989	4730	0	0	0	0	0
$N_p = 76, N_b = 3, N_{con} = 1$	511	1258	1865	2605	4211	0	0	0.3	0.3	0.1
$N_p = 77, N_b = 4, N_{con} = 1$	430	992	1839	2434	4234	4.4	3.4	3.3	3.2	3.0
$N_p = 76, N_b = 5, N_{con} = 1$	446	952	1465	1936	3018	12.5	9.2	8.1	6.8	6.8
$N_p = 79, N_b = 6, N_{con} = 1$	376	743	1081	1705	2620	12.8	13.0	13.1	10.8	10.8
$N_p = 77, N_b = 2, N_{con} = 2$	552	1278	2298	3226	4874	0	0	0	0	0
$N_p = 76, N_b = 3, N_{con} = 2$	564	1372	2098	2977	4554	0	0	0	0	0
$N_p = 77, N_b = 4, N_{con} = 2$	528	1214	2196	3094	4779	0	0	0	0	0
$N_p = 76, N_b = 5, N_{con} = 2$	533	1286	1950	2691	4266	0	0	0.1	0.1	0
$N_p = 79, N_b = 6, N_{con} = 2$	518	1154	1786	2530	4378	0.6	0	0.1	0.4	0.1

Table 5.13: Comparison of costs (J) and collision probability (in %) for different type of optimisations $N_b = [1 \rightarrow 6]/N_{con} = [1, 2]$ and different number of obstacles $N_{obs} = [1, 3, 5, 7, 10]$

Thus, this case study shows another example where the proposed Shifting Strategy could be applied to reduce the computational burden of the optimisation whilst resulting in excellent performance. Although no computation time comparison was performed, it is evident that the proposed approach would result in significantly reduced computation times. As an example, the standard optimisation ($N_p = 75, N_b = 1, N_{con} = 1$) for $N_{obs} = 10$ case would result in 225 decision variables with 750 nonlinear constraints. In contrast, the solution with ($N_p = 76, N_b = 5, N_{con} = 2$) would result in 48 decision variables with only 310 nonlinear constraints.

5.7 Summary

This chapter presented the proposed Shifting Strategy which is considered one of the most important contributions of this thesis. The proposed method was published in a number of articles, namely: an initial conference (abstract-only) paper in IFAC NMPC 2018 [48]; a journal paper in IET Control Theory and Applications 2020 [50]; and the IFAC World Congress 2020 conference paper [59].

The method is based on shifting a set of input-blocking equality constraints in an *absolute time-frame* as established in definition 5.1. Moreover, to avoid having dimensional related problems that potentially require dynamic memory allocation for implementing the approach, the proposed method uses the novel concept of the "Ideal Prediction Horizon" supported by theorem 5.1 which proves that, although counter intuitive, the selected prediction horizon must be an *integer multiple of the block-size plus one* in order to have a constant Hessian dimension, ie. a constant number of decision variables. The application of this concept results in a novel input-blocking parameterisation which the author of this thesis calls "The Ideal Moving Window Blocking (MWB)" approach, defined specifically by equation (5.11). Furthermore, to reduce the computational burden further, the proposed approach uses a reduced set of shifting shooting points and inequality constraints as established by definition 5.2 which allow the optimisation to keep focus on the same overall targets in the *absolute time-frame*. All of these features combined allows the method to preserve nominal stability and recursive feasibility properties when considered in infinite horizons or zero-terminal constraints as proven in theorem 5.3. Finally, one of the benefits of the proposed approach is that by applying the basic concept of *consistent absolute time-frame* shifting, the method can be extended to a wide range of alternatives, both in terms in alternative input-parameterisation as given in figure 5.3, as well as in terms of alternative shifting of shooting points as described by tables 5.1 and 5.2. This gives additional flexibility to the user for its implementation, as well as extends the scope of the contribution.

On the other hand, the chapter includes a set of algorithms and methods provided in section 5.2 which allow an efficient handling and implementation of the overall method. One of the key contributions of this approach is the extension of the $O(N)$ and $O(N^2)$ algorithms from Ph.D. thesis [8] provided in section 5.2.3. Additionally, the section includes the "core" and RTI algorithms required to efficiently implement the approach, as well as a generic computations comparison where the proposed method can be seen to present substantial computational benefits.

The chapter ends with 4 case studies which allow the demonstration of the various properties, advantages and disadvantages of the proposed method, as well as serve as motivation for its usage. These include: the Van Der Poll Oscillator from section 5.3 which gives an example application of the Ideal MWB approach for a system "difficult to stabilise in the presence of actuator constraints"; the WEC device, presented in section 5.4 which presents the results from the publication [59] in which the Non-Ideal MWB approach resulted in substantial performance and computational gains when compared to standard approaches; the Inverted Pendulum, presented in section 5.5 where the performance and recursive feasibility properties of the Shifting Strategy were presented along with a computational comparison where significant computational gains were observed, giving solutions up to 54 times faster than the standard NMPC, including the solution of the ACADO toolkit; and finally, the Obstacle Avoidance Problem from section 5.6, which allowed the demonstration of potential alternatives of the proposed approach applied to a complex problem in which an efficient NMPC solution is required.

A general drawback of the proposed method is that it only guarantees recursive feasibility for the selected shooting points by fixating all attention to them. Therefore, a small slack is required to protect the ignored points from small constraint violations as it was done in the Obstacle Avoidance case study. The selection of the slack size itself is a non-trivial task but could be selected based on Monte Carlo simulations, analysing the system from a variety of conditions, and obviously would be required to increase as the block size increases. However, this problem is also found in most MPC methods because of the discretisation process, and thus is not unique to the proposed method of this chapter.

Thus, based on this evidence, we believe the proposed approach has enough benefits and enough reason for it to be considered relevant for an actual implementation on real systems.

Chapter 6

Closed Loop Dual Mode Nonlinear Model Predictive Control: The Numeric Conditioning Problem

Up until this point, the previous chapters have all been based on the "open-loop" paradigm described in [122] where the predicted evolution of the system is essentially represented by an "open-loop" simulation that includes decision variables for the optimisation at each of the prediction steps, and the resulting optimal feedback control action ($\hat{u}_0$) is applied "after" the optimisation is solved, ie. in an "a-posteriori" manner. This type of approach is generally sufficient for common systems like the Inverted Pendulum presented in the previous chapter 5, or the Quadrotor presented in chapter 4, which suffer from "mild" unstable (or conditionally stable) dynamics (sometimes called unstable modes as they are related to the poles of the system) that do not typically present a problem when using small to medium horizons. There are however, other systems such as the Ball Plate System of [18] or the Triple Inverted Pendulum System from [45, 120] which suffer from severe unstable dynamics which are well known to cause numeric conditioning problems arising from the growing nature of the state-transition dynamics recursion required by condensing-based optimisation frameworks (ie. $A_{N_p} \cdots A_1 A_0$). This causes the Hessian term (E) to become ill-conditioned which makes the numeric calculation of its inverse very inaccurate (in some cases impossible), thus affecting the quality of the optimal solution, or preventing the optimisation to be solved all-together.

A simple solution to this is to use the so called sparse solvers such as QP DUNES [40] or QP FORCES [66] which implement the "simultaneous" approach where the states are kept as decision variables resulting in a large dimensional Quadratic Programs (QPs) that present a clear sparse structure which can be exploited by allowing the solution of each prediction stage separately and connecting them in the final solution, similar to the "principle of optimality" of Dynamic Programming, described in the re-derivation of the Discrete Algebraic Riccati Equation (DARE) presented in chapter 3, section 3.3. Indeed, it is shown in theorem 6 of [40] that the unconstrained solution obtained by QP DUNES has the exact same structure as the dynamic recursion of DARE. However, it is important to note that these methods have been shown to give better performance for medium to large scale Optimal Control Problems (OCPs) [145], and one can also find small OCPs which present this problem.

An alternative approach to this is the well-known "closed-loop" paradigm described in [122], commonly used to tackle Robust Control problems which use a "pre-stabilisation" procedure that imposes feedback "before" the optimisation is done, ie. "a-priori', which results in closed loop state-transition dynamics of the form $\Phi_k = A_k - B_k K$ that contains stable modes and possibly cancels any disturbance, thus solving the underlying problem. This approach has been studied by many authors which have proposed different variants to its implementation, both in linear MPC and Nonlinear MPC, although mostly for linear MPC with most of them based on using a fixed feedback gain K. Moreover, when combined with terminal weights/terminal modes, this approach is typically called "closed loop dual-mode", and was originally proposed for state-space linear MPC in [123]. Other works such as [147] have used the closed-loop paradigm for NMPC with a single locally stabilizing gain K across the entire prediction to stabilize the system around a given steady state target/reference. Finally, work by [29, 30] used a linear time-varying controller calculated offline, to stabilise a nonlinear system around a pre-defined periodic trajectory, which arguably could be refered to as "linear" MPC [55].

A key issue that has not yet been addressed for condensing based NMPC is: *how can we pre-stabilise the system around ANY trajectory that emerge from the nonlinear optimisation, irrespective of the trajectory or prediction horizon requirements?*. The answer to this is relevant bearing in mind that one could be interested in solving optimisations with heavily unstable nonlinear systems on short horizons and/or using a preferred optimised/efficient dense QP solver such as QP OASES [38] which would otherwise not be available for an implementation. At this point it is important to highlight that it is possible the trajectory presents highly unstable dynamics BEFORE even getting close to a steady state target/reference or terminal region, and none of the currently available methods and toolkits, such as the ACADO toolkit, offers a "generic" methodology or "option" for prestabilizing the system to allow condensing approaches to be used in this scenarios. What we must understand is that the application of a single feedback gain K, eg. obtained via the LQR, might be completely inappropriate for parts of the system evolution which have not yet entered the terminal region. A clear example of this is the inverted pendulum (or its bigger brother, the triple inverted pendulum [45]) which undergo a complete input-reversal phenomena where the inputs in the lower-side of the pendulum cause a moment in the complete opposite direction of that in the upper-side, ie. 180° in the frequency domain, which ultimately means what is stabilising for one side, its de-stabilising for the other. On the other hand, the improvement of the numeric conditioning of the optimisation would allow us to obtain more accurate solutions as well as to use reduced numeric accuracy (eg. floats instead of doubles) to obtain faster solutions, something which cannot commonly be exploited using the available condensing methods for highly unstable nonlinear systems.

The aforementioned reasons motivated the contents of this chapter in the hopes of establishing a "generic" condensing-based methodology that could tackle a wide range of unstable nonlinear systems, which has not yet been achieved. Moreover, the method was developed as a "pre-requisite" step towards the final approach presented in chapter 8 which combines pre-stabilisation with the shifting strategy of chapter 5 and the terminal weight methodology of chapter 7. The contents of this chapter resulted in the IET publication [47] related to an experimental validation in a physical double inverted pendulum system, and a submission to the IEEE TAC journal detailing the overall procedure, along with important theorems, novel algorithms and interesting case studies that demonstrate the advantages and disadvantages of the method and allow its efficient implementation using the RTI Scheme.

The chapter is organised as follows: Section 6.1 introduces the proposed closed loop dual mode approach where subsection 6.1.1 presents the pre-stabilisation scheme; subsection 6.1.2 presents the dual mode optimisation framework; subsection 6.1.3 discusses and proves the nominal stability, recursive feasibility and convergence properties of the proposed approach, supported by novel theorem 6.2; and subsection 6.1.3 discusses the implementation of the overall approach in the RTI framework. Following this, section 6.2 presents a set of algorithms that allow the efficient implementation of the approach using the RTI Scheme, including an extension of the $O(N^2)$ and $O(N)$ algorithms of chapter 3 along with a generic computations analysis of the resulting algorithms against those used by the standard approach. Section 6.3 presents a brief case study of a nonlinear ball plate system from [18] which has severe unstable dynamics that were unable to be handled by the condensing-based methodology of the ACADO toolkit as an initial example to demonstrate the potential of the proposed approach. Moreover, section 6.4 presents the application of this methodology to the standard inverted pendulum system where the auto-generation toolkit was tested against the standard condensing-based approach and discusses certain interesting properties, advantages and disadvantages which might be overlooked. Section 6.5 presents the extension of the classical inverted pendulum to a significantly more challenging problem: the Triple Inverted Pendulum system, a problem which was unable to be solved using the standard condensing-based methodology all together given the heavily unstable dynamics that arise from the triple link in the system. Finally, section 6.6 presents experimental validation of the proposed approach in a double inverted pendulum system which formed the main contribution of the IET publication [47]. The chapter ends with a summary of the contents and contributions in section 6.7.

6.1 Closed Loop Dual Mode Prediction Models and Optimisation

As with all the other methodologies of this thesis, the basic/fundamental problems of interest are that of the "state-only" cost given by (3.1), or in the case of nonlinear outputs, the "state-and-output" cost function (3.2). We will see in this section that in order to implement the proposed approach we must modify the overall linearised prediction models of interest, particularly those related to the state and input dynamics, ie. (3.17a) and (3.17b). This will allow us to embed the proposed prestabilisation methodology into the predictions which will ultimately result in a different Quadratic Program that will have the desired properties. Based on this, we will demonstrate how we can implement the Real Time Iteration (RTI) Scheme using this approach.

6.1.1 Prestabilised Prediction Models

In contrast to a linear system where a single linear gain K, typically obtained from LQR, can be used to prestabilise the predictions by imposing it into the cost function (3.1) as an unconstrained control law of the form $u_k = -Kx_k + c_k$, where c_k would become the additional variables of the optimisation as in [122], a nonlinear system may require a time-varying, possibly nonlinear unconstrained control law of the form $u_k = f(x_k) + c_k$ that would stabilise the system to the desired reference, eg. the origin. The development of such a nonlinear control law would likely vary from system to system, thus making its generalisation a hard process. Moreover, the resulting control law may be based on nonlinear strategies such as back-stepping or nonlinear dynamic inversion which completely ignore any desired performance

index of the optimisation as well as any desired constraint handling. On the other hand, one could use the Dynamic Programming (DP) procedure to quickly find the unconstrained solution of the original cost function (3.1) and embed it into the solution in the form of $\hat{u}_k = \bar{u}_k + \bar{u}^*_{unc_k} - K\delta x_k + \delta c_k$. Alternatively, one could avoid solving the DP online by having it calculated off-line for a known trajectory, eg. a periodic trajectory as in [30], or a steady state reference, ie. LQR as in [66], although this would limit the overall application and generalisation of the procedure.

An Underlying Problem and The Proposed Solution

A potential problem of the typical approach where the unconstrained solution is embedded into the predictions as in [122] is that it can potentially place the "free-response" ($\delta\hat{c}_k = 0$ of the pre-stabilised prediction models of the optimisation) on paths that completely ignore the constraints, thus making the solution of feasible initial points more difficult for QP solvers. To solve this problem we propose an alternative approach where rather than seeking to solve the original optimisation (3.1) which focuses on stabilising the system around the original target references (X_r, U_r), we will focus on stabilising the system at whatever nominal trajectory $(\bar{X}, \bar{U})$ is used by the optimisation. By implementing this modification we naturally embed a pre-stabilisation scheme that focuses on maintaining the current optimal constrained plan, rather than finding a new unconstrained plan and re-optimising around it.

The proposed approach can be achieved by formulating a "secondary/inner" unconstrained optimisation with the objective:

$$\min_{\delta\hat{u}_k} \quad J = \sum_{k=0}^{N_p-1} \left[\delta\hat{x}_{k+1}^T q_{k+1}\delta\hat{x}_{k+1} + \delta\hat{u}_k^T r_k \delta\hat{u}_k\right] \tag{6.1a}$$

$$\delta\hat{x}_{k+1} = A_k\delta\hat{x}_k + B_k\delta\hat{u}_k \qquad \forall k = [0 \to N_p - 1] \tag{6.1b}$$

which can be efficiently solved using the Dynamic Programming procedure presented in chapter 3, section 3.4 resulting in the well known Time-Varying Discrete Algebraic Riccati Equation (DARE):

$$P_k = q_k + A_k^T P_{k+1} A_k - A_k^T P_{k+1} B_k K_k \tag{6.2a}$$

$$K_k = (B_k^T P_{k+1} B_k + r_k)^{-1} B_k^T P_{k+1} A_k \tag{6.2b}$$

The method can then obtain a set of optimal gains (K_k) by iterating the dynamic recursion (P_k) backwards $k = [N_p - 1 \to 0]$ starting from $P_{N_p} = q_{k+N_p}$, and using the penalisation terms q_{k+i} $\forall i = [1, N_p - 1]$ and r_{k+i} $\forall i = [0, N_p - 1]$ defined by in the original cost function (3.1).

The proposed scheme results in a control law of the form:

$$\begin{aligned} \hat{u}_k &= \bar{u} + \delta\hat{u}_k \\ &= \bar{u} \underbrace{-K_k\delta\hat{x}_k + \delta\hat{c}_k}_{\delta\hat{u}_k} \end{aligned} \tag{6.3}$$

where at each step, the input automatically reacts to deviations of the nominal trajectory $(\bar{X})$ with the term $(-K_k\delta\hat{x}_k)$ whilst having additional decision variables $(\delta\hat{c}_k)$ for the "main/outer" optimisation.

Before we proceed any further, there are a few important remarks to be made about this methodology. The points below are key remarks that set up the general expectations as well as allow the user to understand the overall objectives, alternatives and limitations of proposed approach.

Remark 6.1. *Inner Optimisation Stability Considerations*

Despite the word "pre-stabilisation" being used to describe the proposed methodology (as typically used in the literature for similar approaches), it is important to clarify that the purpose of the inner optimisation is NOT to achieve nominal stability by itself (that task is solely handled by the outer controller), but rather to "pre-condition" the outer optimisation such that it has improved numeric conditioning, as well as improved hot-starting capabilities. Both of these tasks are achieved by the "inner" optimisation by minimising the predicted deviations from the previous optimal trajectory, thus preventing them from "growing freely", in case the system presents unstable dynamics along the trajectory. By doing this, the columns of the resulting input-to-state/input-to-input prediction matrices (ie. H and F from equations 6.10a and 6.10b) won't be allowed grow freely, therefore resulting in a properly conditioned optimisation. Provided the optimisation can actually minimise/react to the predicted deviations, ie. it has proper q_k, r_k matrices, and the nominal trajectory is stabilisable/controllable (see remark 6.2), the proposed method will typically give good results. Nonetheless, the method does not guarantee (nor requires) nominal stability of the inner controller, neither it guarantees that a proper numeric conditioning will be obtained, but it allows the user to have the freedom to design as required (see remark 6.3).

Remark 6.2. *A stabilisiable trajectory - Not Point-wise Stabilisable A_k and B_k*

Another important point to mention is that it is NOT required to have "point-wise" stabilisable A_k and B_k matrices at each time-step, although having it would ensure that the proposed method can indeed "react/minimise" to the deviations at each time-step. Nonetheless, there may be nonlinear systems where the nominal trajectory momentarily passes through segments with unstable/uncontrollable dynamics, thus causing the inner optimisation to be unable to "react" at those points. However, even if this happens, feedback gains at future stages can still minimise any deviations that emerge from that segment, thus preventing them from "growing freely". Because of this, we refer to the "trajectory" itself being stabilisable/controllable in remark 6.1, not the individual points/time-steps.

Remark 6.3. *Freedom in Weighting Matrices and Design of K_k*

Finally, note that different weighting matrices (q_{k+1} and r_k) can be used in case stronger pre-stabilisation is necessary without affecting the main overall target of cost function (3.1), as it will be proved in theorem 6.2. However, we propose to use the same weights as in the main optimisation to keep things simple, although this freedom on the selection of the pre-stabilisation weights might be advantageous for heavily unstable systems. In general, this gives the user the freedom for designing the feedback gains (K_k) as appropriate, eg. by using those obtained from the infinite-horizon optimisation (K_∞) for the linearised models at each time-step (A_k, B_k), or using pole-placement techniques. Nonetheless, it is important to mention that because of the time-varying nature of the linearisation matrices (A_k, B_k), the matrix-matrix multiplication between two different matrices with stable poles doesn't guarantee a matrix with stable poles. Moreover, the use of infinite horizon gains may not even be possible for A_k, B_k pairs that are not stabilisable, as discussed in remark 6.2, and may take significantly longer time, thus it is not recommended. A short comparison discussing these points further is presented in appendix B.

The Inner Optimisation Equivalency

On the other hand, it is important to clarify that this pre-stabilisation scheme does not penalise or include the multiple shooting offsets (d_{k+1}), ie. it follows a single shooting modelling approach, given that they represent the actual predictions of each segment as depicted in figure 3.1 of chapter 3, and preventing the them from going to this points as it would with $H\delta\hat{U} = \mathbb{O}$ was considered counter-productive by the author of this thesis. This essentially means the inner objective (6.1a) has an overall cost given by $J = ||G\delta x_k + H\delta\hat{U}||^2_Q + ||\delta\hat{U}||^2_R$, ie. $D = \mathbb{O}$. However, even if they were to be included into the overall "inner" objective (6.1a), they would simply result in an additional offset in the overall input model in the form of $\hat{u}_k = \bar{u}_k - K_k\delta\hat{x}_k + \boxed{\bar{c}_k} + \delta\hat{c}_k$ and would become part of the state offsets in the form of $\boxed{d_{k+1} + B_k\bar{c}_k}$ when considered in the state dynamics, but they would not affect neither the recursion of P_k (eq. 6.2a), neither the feedback gain K_k (eq. 6.2b), as proved in theorem 6.1. As a result, the approach would result in the same pre-stabilised matrices (H and F) of the final prediction models (6.10a) and (6.10b) which consequently would result in the optimisation having the same numeric conditioning. The main difference would be the additional computations related to the additional term $\boxed{\bar{c}_k}$ which justifies its neglection, as well as different offset vectors D and S of models (6.10a) and (6.10b), which do not affect the validity of theorem 6.2 proof which is the foundation for proving the nominal stability and in general, relevance of the whole approach.

Theorem 6.1. *The Inner Optimisation Equivalency*

The solution of the proposed "secondary/inner" optimisation (6.1a) would result in the same Time-Varying DARE recursion of P_k, and the same feedback gain K_k with or without the including the offsets (d_{k+1}), and even with or without including the references of the outer optimisation ($x_{r_{k+1}}, u_{r_k}$), ie. when looking to obtain the unconstrained solution.

Note: For the purpose of this theorem, only the offsets case will be presented.

Proof. Consider the cost of two subsequent predictions stages given by:

$$J = \underbrace{\delta x_k^T q_k \delta x_k + 2\delta x_k^T g_k + \delta u_{k-1}^T r_{k-1}\delta u_{k-1}}_{J_k} + \underbrace{\delta x_{k+1}^T P_{k+1}\delta x_{k+1} + 2\delta x_{k+1}^T \tilde{g}_{k+1} + \delta u_k^T r_k \delta u_k}_{J_{k+1}} \tag{6.4a}$$

$$s.t. \quad \delta x_{k+1} = A_k\delta x_k + B_k\delta u_k + d_{k+1} \tag{6.4b}$$

where g_k and $\tilde{g}_{k+1}$ are linear terms w.r.t. δx_k and δx_{k+1}, respectively, that emerge from the including offsets, as it will be seen.

As discussed in chapter 3, the objective of the Dynamic Programming approach is to express the everything that is in the second stage (J_{k+1}) in terms of the first stage (J_k), ie. to solve the optimisation using Bellman's "principle of optimality", solving each of the stages separately and then connecting them.

By substituting the state dynamics (6.4b) and reorganising the terms to form a standard QP w.r.t. δu_k we obtain:

$$J_{k+1} = \delta u_k^T \underbrace{(B_k^T P_{k+1} B_k + r_k)}_{E_k}\delta u_k + 2\delta u_k^T \underbrace{B_k^T\left[P_{k+1}(A_k\delta x_k + d_{k+1}) + \tilde{g}_{k+1}\right]}_{f_k} \tag{6.5}$$

The optimal unconstrained solution to this QP would be given by:

$$\begin{aligned}\delta u_k^* &= -E_k^{-1} f_k \\ &= -K_k \delta x_k + \boxed{\bar{c}_k} \\ &= -\underbrace{E_k^{-1} B_k^T P_{k+1} A_k}_{K_k} \delta x_k \underbrace{-E_k^{-1} B_k^T (P_{k+1} d_{k+1} + \tilde{g}_{k+1})}_{\bar{c}_k}\end{aligned} \tag{6.6}$$

By substituting (6.6) in J_{k+1} of the original cost (6.4a), rearranging the state dynamics in the closed loop form ($\delta x_{k+1} = \Phi_k \delta x_k + \tilde{d}_{k+1}$ with $\Phi_k = A_k - B_k K_k$ and $\tilde{d}_{k+1} = \boxed{d_{k+1} + B_k \bar{c}_k}$) and reorganising the terms w.r.t. δx_k, results in:

$$J_{k+1} = \delta x_k^T (\Phi_k^T P_{k+1} \Phi_k + K_k^T r_k K_k) \delta x_k + 2\delta x_k^T \left[\Phi_k^T (P_{k+1} \tilde{d}_{k+1} + \tilde{g}_{k+1}) - K_k^T r_k \bar{c}_k\right] \tag{6.7}$$

Finally, by substituting (6.7) in (6.4a), and grouping common terms w.r.t. δx_k:

$$J_k = \delta x_k^T \underbrace{(q_k + \Phi_k^T P_{k+1} \Phi_k + K_k^T r_k K_k)}_{P_k} \delta x_k + 2\delta x_k^T \underbrace{\left[g_k + \Phi_k^T (P_{k+1} \tilde{d}_{k+1} + \tilde{g}_{k+1}) - K_k^T r_k \bar{c}_k\right]}_{\tilde{g}_k} \tag{6.8}$$

resulting in the same recursions for P_k, and same feedback gains K_k which concludes the proof. □

In order to obtain the pre-stabilised state prediction model we can substitute the input deviation model ($\delta \hat{u}_k = -K_k \delta \hat{x}_k + \delta \hat{c}_k$) from the proposed pre-stabilisation control law (6.3) into the general multiple-shooting prediction model (3.1) presented in chapter 3, resulting in:

$$\begin{aligned}\hat{x}_{k+1} &= \bar{x}_{k+1} + A_k \delta \hat{x}_k + B_k \overbrace{(-K_k \delta \hat{x}_k + \delta c_k)}^{\delta \hat{u}_k} + d_{k+1} \\ &= \bar{x}_{k+1} + \underbrace{(A_k - B_k K_k)}_{\Phi_k} \delta \hat{x}_k + B_k \delta \hat{c}_k + d_{k+1} \\ &= \bar{x}_{k+1} + \underbrace{\Phi_k \delta \hat{x}_k + B_k \delta c_k + d_{k+1}}_{\delta \hat{x}_{k+1}}\end{aligned} \tag{6.9}$$

where $d_{k+1} = f(\bar{x}_k, \bar{u}_k) - \bar{x}_{k+1}$ is defined as in the general model (3.15b), and $\Phi_k = A_k - B_k K_k$ are the closed-loop state-transition dynamics. Note that Φ_k can be calculated as equation (6.2a), ie. P_k, is iterated backwards and the optimal gains (K_k) are obtained.

Before formulating the final linearised prediction models, note that the input predictions are required to be adjusted to include the feedback terms propagated through the closed-loop state dynamics (Φ_k). By propagating both state and input linearised prediction models (6.9) and (6.3) N_p steps ahead starting from an initial state deviation (δx_0) and reorganising in matrix-vector formats, the predictions of all future states and inputs are then given by:

$$\hat{X} = \bar{X} + \delta \hat{X} = \bar{X} + \underbrace{D + G \delta x_0 + H \delta \hat{C}}_{\delta \hat{X}} \tag{6.10a}$$

$$\hat{U} = \bar{U} + \delta \hat{U} = \bar{U} + \underbrace{S + W \delta x_0 + F \delta \hat{C}}_{\delta \hat{U}} \tag{6.10b}$$

where as before, $\delta x_0 = x_0 - \bar{x}_0$ is an initial condition mismatch that forms part of the RTI Scheme, $\delta\hat{C} = [\delta\hat{c}_k^T, \delta\hat{c}_{k+1}^T, \cdots, \delta\hat{c}_{k+N_p-1}^T]^T \in \mathbb{R}^{N_p n_u}$ is a column-vector containing the new decision variables or inputs of system, and $D \in \mathbb{R}^{N_p n_x}$, $G \in \mathbb{R}^{N_p n_x \times n_x}$, $H \in \mathbb{R}^{N_p n_x \times N_p n_u}$, $S \in \mathbb{R}^{N_p n_u}$, $W \in \mathbb{R}^{N_p n_u \times n_x}$, $F \in \mathbb{R}^{N_p n_u \times N_p n_u}$ are defined as:

$$D = \begin{bmatrix} \bar{d}_1 \\ \bar{d}_2 \\ \vdots \\ \bar{d}_{N_p} \end{bmatrix} \qquad G = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_{N_p} \end{bmatrix} \qquad H = \begin{bmatrix} h_{1,1} & 0 & \cdots & 0 \\ h_{2,1} & h_{2,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ h_{N_p,1} & h_{N_p,2} & \dots & h_{N_p,N_p} \end{bmatrix} \tag{6.11a}$$

$$S = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_{N_p} \end{bmatrix} \qquad W = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{N_p} \end{bmatrix} \qquad F = \begin{bmatrix} f_{1,1} & 0 & \cdots & 0 \\ f_{2,1} & f_{2,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ f_{N_p,1} & f_{N_p,2} & \dots & f_{N_p,N_p} \end{bmatrix} \tag{6.11b}$$

and by dropping the k notation (ie. $\Phi_{k+1} = \Phi_1$), the inner matrix-vector terms are defined through the following recursions as:

$$\overbrace{\bar{d}_k}^{\text{States}} = \begin{cases} d_k & k = 1 \\ d_k + \Phi_{k-1}\bar{d}_{k-1} & k > 1 \end{cases} \qquad \overbrace{s_k}^{\text{Inputs}} = \begin{cases} \mathbb{O}^{n_u} & k = 1 \\ -K_{k-1}\bar{d}_{k-1} & k > 1 \end{cases} \tag{6.12a}$$

$$g_k = \begin{cases} \Phi_{k-1} & k = 1 \\ \Phi_{k-1} g_{k-1} & k > 1 \end{cases} \qquad w_k = \begin{cases} -K_{k-1} & k = 1 \\ -K_{k-1} g_{k-1} & k > 1 \end{cases} \tag{6.12b}$$

$$h_{k,j} = \begin{cases} B_{k-1} & k = j \\ \Phi_{k-1} h_{k-1,j} & k > j \end{cases} \qquad f_{k,j} = \begin{cases} I^{n_u \times n_u} & k = j \\ -K_{k-1} h_{k-1,j} & k > j \end{cases} \tag{6.12c}$$

$$\forall k = [1 \to N_p] \quad \forall j = [1 \to N_p]$$

Remark 6.4. *For modelling of nonlinear outputs, the same linearised prediction model (3.21) can be used considering the new expressions of the pre-stabilised state prediction model (6.10a).*

6.1.2 Condensing-based Optimisation

Having established the linearised state, input, and if required, output prediction models, we can proceed to implement the condensing procedure based on the relative framework by directly substituting them into the original cost function (3.1b), resulting in:

$$\begin{aligned} \min_{\delta\hat{C}} \quad J = {} & \frac{1}{2}(X_r - \bar{X} - D - G\delta x_0 - H\delta\hat{C})^T Q (X_r - \bar{X} - G\delta x_0 - H\delta\hat{C}) \\ & + \frac{1}{2}(\bar{U} + S + W\delta x_0 + F\delta\hat{C} - U_r)^T R (\bar{U} + S + W\delta x_0 + F\delta\hat{C} - U_r) \end{aligned} \tag{6.13a}$$

$$U_{min} \leq \bar{U} + S + W\delta x_0 + F\delta\hat{C} \leq U_{max} \tag{6.13b}$$

$$X_{min} \leq \bar{X} + D + G\delta x_0 + H\delta\hat{C} \leq X_{max} \tag{6.13c}$$

By grouping terms w.r.t. the decision variables ($\delta\hat{C}$), disregarding constant terms and rearranging the inequalities results in a standard Quadratic Program (QP) given by:

$$\min_{\delta\hat{C}} \quad J = \frac{1}{2}\delta\hat{C}^T E\delta\hat{C} + \delta\hat{C}^T f \tag{6.14a}$$

$$E = H^T Q H + F^T R F \tag{6.14b}$$

$$f = -\left[H^T Q(X_r - \bar{X} - D - G\delta x_0) - F^T R(\bar{U} + S + W\delta x_0 - U_r)\right] \tag{6.14c}$$

$$M\delta\hat{C} \leq \gamma \tag{6.14d}$$

$$M = \begin{bmatrix} F \\ -F \\ H \\ -H \end{bmatrix} \quad \gamma = \begin{bmatrix} U_{max} - \bar{U} - S - W\delta x_0 \\ -(U_{min} - \bar{U} - S - W\delta x_0) \\ X_{max} - \bar{X} - D - G\delta x_0 \\ -(X_{min} - \bar{X} - D - G\delta x_0) \end{bmatrix} \tag{6.14e}$$

which can be solved any general purpose solver as QP OASES [38] or quadprog function of Matlab, and will have the same general form discussed in chapter 3, section 3.1.3 where the optimal solution will always be given by:

$$\delta\hat{C}^* = \delta\hat{C}^*_{unc} + \delta\hat{C}^*_{\lambda} \tag{6.15}$$

Once the solution is found, the "expansion step" from the standard approach of the multiple shooting procedure can be implemented to obtain the nominal states and inputs for the following step as:

$$\hat{X} = \bar{X} + D + G\delta x_0 + H\delta\hat{C}^* \tag{6.16}$$

$$\hat{U} = \bar{U} + S + W\delta x_0 + F\delta\hat{C}^* \tag{6.17}$$

Only the first input ($\hat{u}_0$) is applied into the system and the process is repeated in the following step as in the standard receding horizon method.

6.1.3 Stability, Recursive Feasibility and Convergence

In order to prove the nominal stability, recursive feasibility and convergence properties of the proposed approach we could first seek to understand how it relates to the standard methodology which would allow us to derive certain conclusions about its underlying properties.

The Equality of The Pre-stabilised Solution

Based on this interest, we derived theorem 6.2 which states and proves that the solution obtained with this approach will always be exactly the same as the original methodology.

Theorem 6.2. *The Equality of the Pre-stabilised Solution*

The optimal solution with the proposed pre-stabilised prediction models will always be exactly the same as the standard optimal solution which consequently will result in the same nominal stability, recursive feasibility and convergence properties of the standard approach.

Proof. The standard solution, ie. the one that uses the predictions with $K_k = \mathbb{O}$, results in $F = I$, $S = W = \mathbb{O}$, and therefore $\delta\hat{C} = \delta\hat{U}$ which results in the standard prediction models:

$$\hat{X} = \bar{X} + D_1 + G_1\delta x_0 + H_1\delta\hat{U} \tag{6.18a}$$
$$\hat{U} = \bar{U} + \delta\hat{U} \tag{6.18b}$$

In contrast, the proposed approach uses the pre-stabilised prediction models:

$$\hat{X} = \bar{X} + D_2 + G_2\delta x_0 + H_2\delta\hat{C} \tag{6.19a}$$
$$\hat{U} = \bar{U} + \underbrace{S + W\delta x_0 + F\delta\hat{C}}_{\delta\hat{U}} \tag{6.19b}$$

Notice the $D_1/D_2 - G_1/G_2 - H_1/H_2$ notation has been used to distinguish the two state prediction models. Similar notation will be used throughout the proof where relevant, eg. as E_1 in equation 6.22.

Although we have different prediction models, we can expect that both models would result in the exact same predictions for a given "total" input deviation ($\delta\hat{U}$). Thus, by replacing the pre-stabilised prediction of the input deviations ($\delta\hat{U} = S + W\delta x_0 + F\delta\hat{C}$) in equation (6.18a) we obtain:

$$\begin{aligned}\hat{X}_1 &= \bar{X} + D_1 + G_1\delta x_0 + H_1(S + W\delta x_0 + F\delta\hat{C}) \\ &= \bar{X} + \underbrace{(D_1 + H_1S)}_{D_2} + \underbrace{(G_1 + H_1W)}_{G_2}\delta x_0 + \underbrace{H_1F}_{H_2}\delta\hat{C}\end{aligned} \tag{6.20}$$

which allows us to derive the following relationships:

$$D_2 = D_1 + H_1S \tag{6.21a}$$
$$G_2 = G_1 + H_1W \tag{6.21b}$$
$$H_2 = H_1F \tag{6.21c}$$

Following the procedure of theorem 3.2 presented in chapter 3, we can begin by first proving unconstrained solutions ($\delta\hat{U}^*_{unc}$) are the same, and proving the equality of constrained solutions afterwards.

In the standard case, we have an optimal unconstrained solution of the form:

$$\begin{aligned}\delta\hat{U}^*_{unc} &= \overbrace{\left(H_1^TQH_1 + R\right)}^{E_1}{}^{-1}\overbrace{\left[H_1^TQ(X_r - \bar{X} - D_1 - G_1\delta x_0) - R(\bar{U} - U_r)\right]}^{-f_1} \\ &= -E_1^{-1}f_1\end{aligned} \tag{6.22}$$

In contrast, the proposed approach has an optimal unconstrained solution of the form:

$$\begin{aligned}\delta\hat{U}^*_{unc} = S + W\delta x_0 + F\overbrace{\left(H_2^TQH_2 + F^TRF\right)}^{E_2}{}^{-1}\big[&H_2^TQ(X_r \\ &- \bar{X} - D_2 - G_2\delta x_0) - F^TR(\bar{U} + S + W\delta x_0 - U_r)\big]\end{aligned} \tag{6.23}$$

By substituting equation (6.21c) in (6.23) and rearranging it noting that the Hessian of the proposed approach (E_2) can be expressed in terms of the standard Hessian (E_1) as $E_2 = F^T E_1 F$ gives:

$$\delta\hat{U}^*_{unc} = S + W\delta x_0 + \boxed{F\left(F^T E_1 F\right)^{-1} F^T}\left[H_1^T Q(X_r - \bar{X} - D_2 - G_2\delta x_0) - R(\bar{U} + S + W\delta x_0 - U_r)\right] \tag{6.24}$$

Given F is always invertible because of the identity matrix in the diagonal, and E_1 is always invertible because is positive definite, the terms related to the inverse of the inner product signaled by the red box above are given by:

$$\begin{aligned} F\left(F^T E_1 F\right)^{-1} F^T &= F(F^{-1}E_1^{-1}F^{T^{-1}})F^T \\ &= E_1^{-1} \end{aligned} \tag{6.25}$$

Substituting equations (6.21b), (6.21a) and (6.25) in (6.24) gives:

$$\delta\hat{U}^*_{unc} = S + W\delta x_0 + E_1^{-1}\left[H_1^T Q(X_r - \bar{X} - D_1 - H_1 S - G_1\delta x_0 - H_1 W\delta x_0) - R(\bar{U} + S + W\delta x_0 - U_r)\right] \tag{6.26}$$

Rearranging terms:

$$\begin{aligned} \delta\hat{U}^*_{unc} &= S + W\delta x_0 - \cancel{E_1^{-1}}\cancel{E_1}(S + W\delta x_0) \\ &\quad + E_1^{-1}\left[H_1^T Q(X_r - \bar{X} - D_1 - G_1\delta x_0) - R(\bar{U} - U_r)\right] \\ &= E_1^{-1}\left[H_1^T Q(X_r - \bar{X} - D_1 - G_1\delta x_0) - R(\bar{U} - U_r)\right] \\ &= -E_1^{-1} f_1 \end{aligned} \tag{6.27}$$

Thus, the equality of the unconstrained solutions (6.27) and (6.22) holds.

Having proved the equality of the unconstrained solutions, the equality of constrained solutions reduces to proving the optimal correction terms related to the Lagrange Multipliers ($\delta\hat{U}^*_\lambda = F\delta\hat{C}^*_\lambda$) are exactly the same. Following the procedure for optimal constrained solutions established in chapter 3 and used by theorem 3.2, the optimal correction term for both solutions is known to be given by:

$$\delta\hat{U}^*_\lambda = -E_1^{-1} M_1^T \lambda_1^* \qquad \text{Standard} \tag{6.28a}$$

$$\delta\hat{U}^*_\lambda = -F E_2^{-1} M_2^T \lambda_2^* \qquad \text{Dual Mode} \tag{6.28b}$$

with the optimal vectors of Lagrange Multipliers (λ_1^* and λ_2^*) containing only positive or zero values that satisfy the Karush-Kush-Tucker (KKT) conditions.

Noting that $M_2 = M_1 F$, substituting it in equation (6.28b) and using the equivalency between the Hessians (6.25), we obtain:

$$\begin{aligned} \delta\hat{U}^*_\lambda &= -\boxed{F(F^T E_1 F)^{-1} F^T} M_1^T \lambda_2^* \\ &= -E_1 M_1^T \lambda_2^* \end{aligned} \tag{6.29}$$

This reduces to proving both optimal vectors of Lagrange Multipliers will be the same. ie. $\lambda_1^* = \lambda_2^*$. To prove this, we can then apply the active-set approach used for theorem 3.2 where for any given active set, the optimal Lagrange Multipliers of each approach would be given by:

$$\lambda^*_{act_1} = -(M_{act_1} E_1^{-1} M_{act_1}^T)^{-1}(\gamma_{act_1} - M_{act_1}\delta\hat{U}^*_{unc}) \tag{6.30a}$$
$$\lambda^*_{act_2} = -(M_{act_2} E_2^{-1} M_{act_2}^T)^{-1}(\gamma_{act_2} - M_{act_2}\delta\hat{C}^*_{unc}) \tag{6.30b}$$

where M_{act_1}/γ_{act_1} and M_{act_2}/γ_{act_2} are the active-set constraints matrix/vector of each approach.

Note that because, $M_2 = M_1 F$, then $M_{act_2} = M_{act_1} F$. Substituting this along with the hessian equivalency $E_2 = F^T E_1 F$ from (6.25) in (6.30b) results in:

$$\begin{aligned}\lambda^*_{act_2} &= -(M_{act_1} \boxed{F(F^T E_1 F)^{-1} F^T} M_{act_1}^T)^{-1}(\gamma_{act_2} - M_{act_2}\delta\hat{C}^*_{unc}) \\ &= -(M_{act_1} E_1^{-1} M_{act_1}^T)^{-1}(\gamma_{act_2} - M_{act_2}\delta\hat{C}^*_{unc})\end{aligned} \tag{6.31}$$

Thus, by equating (6.30a and 6.30b) the inverse related term cancels which results in:

$$\gamma_{act_1} - M_{act_1}\delta\hat{U}^*_{unc} = \gamma_{act_2} - M_{act_2}\delta\hat{C}^*_{unc} \tag{6.32}$$

For the strict purpose of the proof, consider the active-set composed by the entire set, something which as stated in theorem 3.2 can not be done in practice given the requirement of linear independence of the active sets for invertibility of the matrix $(ME^{-1}M^T)^{-1}$, and the restriction of the number of active-sets being less than the number of decision variables, but is equally valid for the proof.

Given we have proved the unconstrained solutions to be the same, ie. $\delta\hat{U}^*_{unc} = S + W\delta x_0 + F\delta\hat{C}^*_{unc}$, we can substitute this in (6.32) along with the equivalence expressions of (6.21) resulting in:

$$\begin{aligned}\gamma_1 - M_1\delta\hat{U}^*_{unc} &= \overbrace{\begin{bmatrix} U_{max} - \bar{U} \\ \bar{U} - U_{min} \\ X_{max} - \bar{X} - D_1 - G_1\delta x_0 \\ \bar{X} + D_1 + G_1\delta x_0 - X_{min} \end{bmatrix}}^{\gamma_1} - \overbrace{\begin{bmatrix} I \\ -I \\ H_1 \\ -H_1 \end{bmatrix}}^{M_1} \overbrace{(S + W\delta x_0 + F\delta\hat{C}^*_{unc})}^{\delta\hat{U}^*_{unc}} \\ &= \underbrace{\begin{bmatrix} U_{max} - \bar{U} - S - W\delta x_0 \\ \bar{U} + S + W\delta x_0 - U_{min} \\ X_{max} - \bar{X} - D_2 - G_2\delta x_0 \\ \bar{X} + D_2 + G_2\delta x_0 - X_{min} \end{bmatrix}}_{\gamma_2} - \underbrace{\begin{bmatrix} F \\ -F \\ H_2 \\ -H_2 \end{bmatrix}}_{M_2} \delta\hat{C}^*_{unc} = \gamma_2 - M_2\delta\hat{C}^*_{unc}\end{aligned} \tag{6.33}$$

which concludes the proof. □

Note that this proof also holds for single-shooting scenarios where the system is linearized along the nominal state trajectory ($\bar{X}$) obtained with the nominal input ($\bar{U}$) resulting in $d_k = \mathbb{0}^{n_x}\ \forall\ k = [1, N_p]$, and consequently in $S = D = \mathbb{0}^{N_p n_x}$. Moreover, the proof holds independently of the selection of the pre-stabilisation weights, ie. independently of the feedback gains used, as discussed earlier in the formulation of the inner objective function (6.1a), given F will always be invertible which essentially means that if the new decision variables ($\delta\hat{C}$) can replicate the original variables ($\delta\hat{U}$), they will.

Advantages

Because the solution of the proposed approach will always be exactly the same as the standard one, we can use the same methodologies dicussed in chapter 3, section 3.4 such as zero terminal constraints, terminal weights, invariant sets and convergence improvement methods which ultimately result in the method having exactly the same nominal stability, recursive feasibility and convergence properties. However, the use of the proposed pre-stabilisation scheme will provide appropriate numeric conditioning to the problem which will lead to a numerically robust Hessian inversion required by the optimisation. This will allow the optimisation of a wide range of unstable nonlinear systems based on the condensing methodology on any prediction horizon without sacrificing numerical robustness of the solution. Moreover, the method could allow the use of less accurate inverse solutions and weaker numeric representations such as floats which could ultimately speed up the overall solution. Examples of the potential advantages of the proposed method will be presented in the case studies of sections 6.3, 6.4 and 6.5.

Disadvantages

On the other hand, the method has some inherent disadvantages, particularly due to the additional computations required for the computation of DARE (ie. the calculation of P_k, K_k, Φ_k), as well as the additional matrices S, W, F, and the fact that input constraints essentially have the form of output constraints which prevents the special case available in standard QPs where the inputs are constrained through an identity matrix (I) rather than dense matrix (F), all of which will ultimately increase the overall computation times when compared to the standard approach. Moreover, assuming the system was linearised around a feasible input trajectory ($\bar{U}$), an initial feasible point for ($\hat{U}$) is not necessarily achieved by $\delta\hat{C} = \mathbb{O}^{N_p n_u}$ as opposed to the standard methodology where imposing $\delta\hat{U} = \mathbb{O}^{N_p n_u}$ directly would be feasible. This can ultimately make finding a feasible initial point for what are typically the "easiest" constraints a hard process. However, note that a straightforward solution to this is to compute $\delta\hat{C} = F^{-1}(\mathbb{O}^{N_p n_u} - S - W\delta x_0)$ which is the required $\delta\hat{C}$ to obtain $\delta\hat{U} = \mathbb{O}^{N_p n_u}$. This brings the question of whether it would be possible develop QPs specially designed to handle the pre-stabilised input structure F better.

6.1.4 Real Time Iterations

To implement this approach using the Real Time Iteration (RTI) Scheme the user can simply follow the guidelines established chapter 3, section 3.2 and apply the standard steps, ie. performing the Initial Value Embedding (IVE) shifting approach, performing a single full-step SQP iteration and separating the computations in Preparation and Feedback Phases by using the predicted state obtained from the previous state and input ($\bar{x}_0 = \hat{x}_{k|k-1} = f(x_{k-1|k-1}, u_{k-1|k-1})$), calculating the operations related to the initial offset ($\delta x_0 = x_0 - \bar{x}_0$) as soon as the measurement arrives and solving the QP.

On the other hand, if the user requires to have stricter deterministic timings, an alternative is to implement the approach discussed in chapter 3 to completely remove the time-delay related to the QP solution in the feedback phase by implementing equation (6.34) which as discussed, is based on solving the problem with the assumption of $\delta x_0 = 0$, and then re-including the feedback term to provide an optimal "unconstrained correction" to the "approximated constrained solution".

$$\hat{U} = \bar{U} + \overbrace{\underbrace{S - FE^{-1}\left[-H^TQ(X_r - \bar{X} - D) - F^TR(U_r - \bar{U} - S)\right]}^{\text{Unconstrained}} - \underbrace{FE^{-1}M^T\bar{\lambda}}_{\text{Constrained}}}^{\text{Preparation Phase}} - \underbrace{\left[FE^{-1}(H^TQG + F^TRW) - W\right]}_{\text{Feedback Phase}}\delta x_0 \tag{6.34}$$

This particular approach was used in the experiment validation process of the double inverted pendulum case study presented in section 6.6, and was published in [47] along with the overall pre-stabilisation methodology and a hybrid switching scheme that will be discussed in the case study's section.

6.2 Algorithm Details and Auto-generation

Having defined the theory of the overall methodology, we can proceed to define the key algorithms that are required to implement the proposed approach using the developed auto-generation toolkit based on the RTI framework which will ultimately allow us to benchmark the proposal against the solution obtained by the ACADO toolkit. Thus, this section will provide a set of algorithms that allow its efficient implementation including an extension of the $O(N^2)$ and $O(N)$ algorithms from PhD thesis [8] presented in the re-derivation of chapter 3, section 3.3.1, which is considered one of the key contributions of this thesis, along with 4 "core" algorithms that will be used by the Preparation and Feedback algorithms of the RTI approach.

6.2.1 Extension of the $O(N^2)$ and $O(N)$ Algorithms

Apart from S and W of the pre-stabilised input prediction model (6.10b) which result in simple vectors (with the understanding that the effect of δx_0 can be included in S as it was included in D in algorithm 3.6), the main and most important difference of the whole approach proposed in this chapter comes from matrix F, particularly from the Hessian term F^TRF (previously simply R), and the linear term $F^TR(\bar{U} + S + W\delta x_0 - U_r)$. In this subsection we will see that we can use the philosophy of the $O(N^2)/O(N)$ algorithms to calculate this terms efficiently, thus resulting in an "extension" to the fundamental algorithms provided in [8].

Following a similar procedure as in the re-derivation of the original algorithms presented in section 3.3.1, let us begin by expanding the first $N_p = 3$ terms of the first column of the total Hessian considering dummy matrices $\tilde{W}$ and $\tilde{V}$ as:

$$\begin{bmatrix} E_{1,1} \\ E_{2,1} \\ E_{3,1} \end{bmatrix} = \underbrace{\begin{bmatrix} B_0^T & B_0^T\Phi_1^T & B_0^T\Phi_1^T\Phi_2^T \\ 0 & B_1^T & B_1^T\Phi_2^T \\ 0 & 0 & B_2^T \end{bmatrix}}_{H_1}\underbrace{\begin{bmatrix} \tilde{w}_{1,1} \\ \tilde{w}_{2,1} \\ \tilde{w}_{3,1} \end{bmatrix}}_{\tilde{W}} + \underbrace{\begin{bmatrix} I & -B_0^TK_1^T & -B_0^T\Phi_1^TK_2^T \\ 0 & I & -B_1^TK_2^T \\ 0 & 0 & I \end{bmatrix}}_{F}\underbrace{\begin{bmatrix} \tilde{v}_{1,1} \\ \tilde{v}_{2,1} \\ \tilde{v}_{3,1} \end{bmatrix}}_{\tilde{V}} \tag{6.35}$$

where the dummy matrices $\tilde{W}$ and $\tilde{V}$ will eventually represent columns of the operations $\tilde{W} = QH$, $\tilde{W} = QX_e$, $\tilde{V} = RF$ or $\tilde{V} = RU_e$, but are not particularly required to find the underlying pattern of the operations.

From equation (6.35) we can easily obtain the last value given by:

$$E_{3,1} = \tilde{v}_{3,1} + B_2^T \tilde{w}_{3,1}^{[1]} \tag{6.36}$$

where as before, the notation $\tilde{w}_{3,1}^{[0]}/\tilde{w}_{3,1}^{[1]}$ represents the initial/final value of the algorithm. Note that this particular variable starts in the final value as it does not require any modification.

The following expression can be expressed in terms of the previous dummy variable ($\tilde{w}_{3,1}^{[1]}$) as:

$$E_{2,1} = \tilde{v}_{2,1} + B_1^T \underbrace{(\tilde{w}_{2,1}^{[0]} + \Phi_2^T \boxed{\tilde{w}_{3,1}^{[1]}} - K_2^T \tilde{v}_{3,1})}_{\tilde{w}_{2,1}^{[1]}} \tag{6.37}$$

Moving on to the last term, we can express it in terms of the previous dummy variable ($\tilde{w}_{2,1}^{[1]}$) as:

$$\begin{aligned} E_{1,1} &= \tilde{v}_{1,1} + B_0^T (\tilde{w}_{1,1}^{[0]} + \Phi_1^T \overbrace{(\tilde{w}_{2,1}^{[0]} + \Phi_2^T \tilde{w}_{3,1}^{[1]} - K_2^T \tilde{v}_{3,1})}^{\boxed{\tilde{w}_{2,1}^{[1]}}} - K_1^T \tilde{v}_{2,1}) \\ &= \tilde{v}_{1,1} + B_0^T \underbrace{(\tilde{w}_{1,1}^{[0]} + \Phi_1^T \boxed{\tilde{w}_{2,1}^{[1]}} - K_1^T \tilde{v}_{2,1})}_{\tilde{w}_{1,1}^{[1]}} \end{aligned} \tag{6.38}$$

Thus a clear static pattern can be seen where the Hessian can be calculated in terms of the modified dummy variable $\tilde{w}_{k,j}^{[1]}$ as: $E_{k,j} = \tilde{v}_{k,j} + B_{k-1}^T \tilde{w}_{k,j}^{[1]}$, and the dummy variable is defined as a recursive expression given by: $\tilde{w}_{k,j}^{[1]} = \tilde{w}_{k,j}^{[0]} + \Phi_k^T \tilde{w}_{k+1,j}^{[1]} - K_k^T \tilde{v}_{k+1,j}$.

This solution is also valid for the calculation of the linear term as well as any column of the Hessian, however, as with the standard $O(N^2)$ algorithm, only the diagonal terms are calculated and the rest are duplicated. Based on this and the understanding that the operations related to the dummy variables are $\tilde{W} = QH$, $\tilde{W} = QX_e = Q(X_r - \bar{X} - D - G\delta x_0)$, $\tilde{V} = RF$ and $\tilde{V} = RU_e = R(U_r - \bar{U} - S - W\delta x_0)$, the final algorithms are given in algorithms 6.1 and 6.2. Note that the expression for U_e has been reversed from the original to match the linear term $f = -(H^T Q X_e + F^T R U_e)$.

Algorithm 6.1: Dual Mode $O(N)$ Condensing Algorithm

Data: $Q, R, \Phi_k, B_k, K_k, X_e, U_e, N_p$

begin

- $\tilde{w}_{N_p} = q_{N_p} X_{e_{N_p}}$; // Initial value of the main dummy variable
- // For loop running backwards $k = N_p, N_p - 1, \cdots, 2$
- **for** $k = N_p$ **to** 2 **do**
  - $\tilde{v}_k = r_{k-1} U_{e_{k-1}}$; // Calculate secondary dummy variable
  - $f_k = -(\tilde{v}_k + B_{k-1}^T \tilde{w}_k)$; // Calculate linear term component
  - $\tilde{w}_{k-1} = q_{k-1} X_{e_{k-1}} + \Phi_{k-1}^T \tilde{w}_k - K_{k-1}^T \tilde{v}_k$; // Propagate recursively
- **end**
- $f_1 = -(r_0 U_{e_0} + B_0^T \tilde{w}_1)$; // Initial term outside the loop

end

Result: f

Algorithm 6.2: Dual Mode $O(N^2)$ Condensing Algorithm

Data: $H, Q, F, R, \Phi_k, B_k, K_k, N_p$

begin

  for $i = 1$ **to** N_p **do**

    $\tilde{w}_{N_p,i} = q_{N_p} h_{N_p,i}$; // Initial value of the main dummy variable

    // For loop running backwards $k = N_p, N_p - 1, \cdots, i + 1$

    for $k = N_p$ **to** $i + 1$ **do**

      $\tilde{v}_{k,i} = r_{k-1} f_{k,i}$; // Calculate secondary dummy variable

      $E_{k,i} = \tilde{v}_{k,i} + B_{k-1}^T \tilde{w}_{k,i}$; // Calculate Hessian term component

      $\tilde{w}_{k-1,i} = q_{k-1} h_{k-1,i} + \Phi_{k-1}^T \tilde{w}_{k,i} - K_{k-1}^T \tilde{v}_{k,i}$; // Propagate recursively

      $E_{i,k} = E_{k,i}^T$; // Copy transpose

    end

    $E_{i,i} = r_{i-1} f_{i,i} + B_{i-1}^T \tilde{w}_{i,i}$; // Diagonal term calculation

  end

end

Result: E

6.2.2 Core Algorithms

In addition to the 2 novel algorithms introduced in the previous section, there are 4 additional "core" algorithms required for the implementation of the whole approach which are (given in order of introduction and usage): 1. Time-Varying DARE; 2. Dual Mode H and F Matrices Calculation; 3. Dual Mode D and S Vectors Calculation, and; 4. Dual Mode Decompression Routine. These routines will be fundamental for the final RTI algorithms introduced in the following subsection.

The Time-Varying DARE algorithm allows us to calculate the pre-stabilised state-transition matrices Φ_k, as well as the optimal feedback gains K_k by using the recursive expression of (6.2a) and (6.2b), and is given in algorithm 6.3.

Algorithm 6.3: Dual Mode Time-Varying DARE Algorithm

Data: A_k, B_k, Q, R, N_p

begin

  $P_{N_p} = q_{N_p}$; // Initial value for weighting matrix P_{k+1}

  // For loop running backwards $k = N_p - 1, N_p - 2, \cdots, 0$

  for $k = N_p - 1$ **to** 0 **do**

    $K_k = (B_k^T P_{k+1} B_k + r_k)^{-1} B_k^T P_{k+1} A_k$; // Calculate feedback gain

    $\Phi_k = A_k - B_k K_k$; // Close System Dynamics

    $P_k = q_k + \Phi_k^T P_{k+1} \Phi_k + K_k^T r_k K_k$; // Propagate Weights

  end

end

Result: Φ_k, K_k

Moreover, the H and F matrices can be efficiently calculated by algorithm 6.4 which is based on the recursive expressions defined in (6.12c). Similarly, we can include the initial offset δx_0 into both terms D and S as performed in the original algorithm (3.6) for D, simply by having the initial conditions $\bar{d}_1 = \Phi_0 \delta x_0 + d_1$ and $\bar{s}_1 = -K_0 \delta x_0$. This can be performed efficiently with algorithm 6.5 which is based on expressions (6.12a).

Algorithm 6.4: Dual Mode H and F Condensing Calculation

Data: Φ_k, B_k, K_k, N_p

begin
  for $i = 1$ **to** N_p **do**
    $h_{i,i} = B_{i-1}$; // Diagonal Term of H
    $f_{i,i} = I^{n_u \times n_u}$; // Diagonal Term of F
    for $k = i + 1$ **to** N_p **do**
      $h_{k,i} = \Phi_{k-1} h_{k-1,i}$; // Propagate H terms recursively
      $f_{k,i} = -K_{k-1} h_{k-1,i}$; // Calculate F in terms of H
    end
  end
end

Result: H, F

Algorithm 6.5: Dual Mode D and S Calculation

Data: $\Phi_k, K_k, d_k, \delta x_0, N_p$

begin
  $\bar{d}_1 = \Phi_0 \delta x_0 + d_1$; // Calculate Initial D value
  $\bar{s}_1 = -K_0 \delta x_0$; // Calculate Initial S value
  for $k = 2$ **to** N_p **do**
    $\bar{d}_k = d_k + \Phi_{k-1} \bar{d}_{k-1}$; // Propagate D terms recursively
    $s_k = -K_{k-1} \bar{d}_{k-1}$; // Calculate S terms
  end
end

Result: D, S

Finally, to perform the "expansion step" required by the multiple shooting approach, note that we only require to calculate the additional terms $\delta\tilde{X} = H\delta\hat{C}^*$ and $\delta\tilde{U} = F\delta\hat{C}^*$ related to the optimal pre-stabilised correction. To do so, we can use algorithm 6.6 which calculates this term efficiently by propagating them through the dynamics instead of their direct evaluation, thus avoiding the calculations related to the zero terms above the diagonal, and any repeated use of the terms in vector $\delta\hat{C}^*$.

Algorithm 6.6: Dual Mode $\delta\tilde{X}$ and $\delta\tilde{U}$ Expansion Step

Data: $\Phi_k, B_k, K_k, \delta\hat{C}^*, N_p$

begin
  $\delta\tilde{u}_0 = \delta c_0^*$; // Calculate initial $\delta\tilde{U}$ value
  $\delta\tilde{x}_1 = B_0 \delta c_0^*$; // Calculate Initial $\delta\tilde{X}$ value
  for $k = 2$ **to** N_p **do**
    $\delta\tilde{u}_{k-1} = -K_{k-1}\delta\tilde{x}_{k-1} + \delta\hat{c}_{k-1}^*$; // Calculate $\delta\tilde{u}_{k-1}$
    $\delta\tilde{x}_k = \Phi_{k-1}\tilde{x}_{k-1} + B_{k-1}\delta c_{k-1}^*$; // Propagate δc_{k-1}^* through dynamics
  end
end

Result: $\delta\tilde{X}, \delta\tilde{U}$

6.2.3 RTI Algorithms

Having established the fundamental algorithms required for the implementation of the proposed Closed-Loop Dual-Mode NMPC approach under the RTI Scheme, the overall approach is finally provided in terms of the Preparation and Feedback Phases given in algorithms 6.7 and 6.8, respectively, both of which are based on the previously presented algorithms, as well as forward simulation algorithm 3.4 to facilitate the verification process of each working part of the proposed approach.

Algorithm 6.7: Dual Mode RTI NMPC Preparation Phase Algorithm

Data: $\bar{X}, \bar{U}, \bar{\lambda}, x_{-1}, u_{-1}, Q, R, N_p$

begin

$\bar{x}_0 = f(x_{-1}, u_{-1})$; // Calculate predicted state from previous state and input

Shift $\bar{X}$,$\bar{U}$, and optionally $\bar{\lambda}$ consistently ; // Initial Value Embedding

$[A_k, B_k, d_k] = Forward(\bar{X}, \bar{U}, \bar{x}_0, N_p)$; // Run algorithm 3.4

$[\Phi_k, K_k] = DARE(A_k, B_k, Q, R, N_p)$; // Run algorithm 6.3

$[H, F] = CalculateHF(\Phi_k, B_k, K_k, N_p)$; // Run algorithm 6.4

$[E] = CalculateE(H, Q, F, R, \Phi_k, B_k, K_k, N_p)$; // Run algorithm 6.2

$M = \begin{bmatrix} F^T & -F^T & H^T & -H^T \end{bmatrix}^T$; // Form M Matrix

end

Result: $E, M, \Phi_k, B_k, d_k, \bar{x}_0$

Algorithm 6.8: Dual Mode RTI NMPC Feedback Phase Algorithm

Data: $x_0, \bar{x}_0, \bar{X}, \bar{U}, \bar{\lambda}, X_r, U_r, E, M, \Phi_k, B_k, d_k, Q, R, N_p, U_{max}, U_{min}, X_{max}, X_{min}$

begin

$\delta x_0 = x_0 - \bar{x}_0$; // Calculate state deviation from measurement

$[D, S] = CalculateDS(\Phi_k, K_k, d_k, \delta x_0, N_p)$; // Run algorithm 6.5

$X_e = X_r - \bar{X} - D$; // Calculate X error

$U_e = U_r - \bar{U} - S$; // Calculate U error

$[f] = Calculatef(Q, R, \Phi_k, B_k, K_k, X_e, U_e, N_p)$; // Run algorithm 6.1

$\gamma = \begin{bmatrix} U_{max} - \bar{U} - S \\ \bar{U} + S - U_{min} \\ X_{max} - \bar{X} - D \\ \bar{X} + D - X_{min} \end{bmatrix}$; // Calculate constraint vector γ

$[\delta\hat{C}^*, \bar{\lambda}] = QPSolve(E, f, M, \gamma, \bar{\lambda})$; // Solve the Quadratic Program

$[\delta\tilde{X}, \delta\tilde{U}] = Expand(\Phi_k, B_k, K_k, \delta\hat{C}^*, N_p)$; // Run algorithm 6.6

$\bar{U} = \bar{U} + S + \delta\tilde{U}$; // Calculate new nominal input

$\bar{X} = \bar{X} + D + \delta\tilde{X}$; // Calculate new nominal state

end

Result: $\bar{X}, \bar{U}, \bar{\lambda}$

6.2.4 Generic Computations

In order to get an idea of how well the proposed approach performs compared to the standard approach, we performed a generic computations comparison of all the relevant Dual Mode algorithms (ie. algorithms 6.3, 6.4, 6.2, 6.1, 6.5, 6.1 and 6.6) against their counterpart algorithms from the standard approach (ie. algorithms 3.5, 3.2, 3.6, 3.3 and 3.7). For reference, table A.1 contains the Floating Point Operations (FLOPS) counts of the developed algorithms, along with those of the standard approach.

As it is well known that the DARE algorithm 6.3 which is one of the core fundamental differences scales linearly with both the number of inputs and the number of prediction steps, ie. it has an $O(Nn_u)$ performance [28], the most reasonable comparison to make was using different n_u and N_p for a system with fixed n_x states. Hence, to relate the results of this comparison to that of the inverted pendulum dynamics presented in chapter 5, as well as later in this chapter in case study 6.4, the comparison was made for a system with $n_x = 4$ states, $n_u = [1, 2, 3]$ inputs, and $N_p = [100, 150, 200]$ prediction steps using only doubles precision as no particular gain was observed from using floats. All the algorithms were tested in Ubuntu 20.04 running with Real-Time priority (ie. chrt -r 99 ./main) on a laptop with an Intel i7-8750 CPU overclocked @ 3.9 GHz, and 32 GB DDR4 RAM running @ 2,666 MHz, with 120000 runs per algorithm. The test C++ codes were compiled using the (-O3) optimisation C-flag, as well as with the fused-multiply-addition operations (-mfma) and auto-vectorisation (-mavx) flags enabled to use the Advanced Vector Instruction set available in the Intel CPU. The results of this comparison are gathered in table 6.1 where the minimum computation time obtained for each algorithm is reported, indicating the minimum time that could be achieved if a Real-Time OS would be used.

Type	Dual Mode Approach								
Case ($n_x = 4$)	$n_u = 1$			$n_u = 2$			$n_u = 3$		
N_p	100	150	200	100	150	200	100	150	200
DARE (alg. 6.3)	6	9	12	10	15	20	14	21	28
H/F Matrices (alg. 6.4)	12	25	51	26	63	118	53	122	249
E Matrix (alg. 6.2)	14	33	67	41	100	189	89	235	551
D/S Vectors (alg. 6.5)	1	2	2	1	2	2	2	2	3
f Vector (alg. 6.1)	1	2	2	2	2	2	2	2	3
Expansion Step (alg. 6.6)	1	2	2	1	2	2	2	2	2
Total	35	77	136	81	184	333	162	384	836
Type	Standard Approach								
Case ($n_x = 4$)	$n_u = 1$			$n_u = 2$			$n_u = 3$		
N_p	100	150	200	100	150	200	100	150	200
H Matrix (alg. 3.5)	6	14	31	14	28	64	23	46	103
E Matrix (alg. 3.2)	12	26	54	32	73	144	64	151	367
D Vector (alg. 3.6)	1	2	2	1	2	2	1	2	2
f Vector (alg. 3.3)	1	2	2	2	2	2	2	2	3
Expansion Step (alg. 3.7)	1	2	2	1	2	2	1	2	2
Total	21	46	91	50	107	214	91	203	477
Increase (+%)	67	67	50	62	70	55	78	89	75

Table 6.1: Generic Computation Times (in μs) Comparison of Standard and Closed Loop Dual Mode Approaches for a system with $n_x = 4$ states, $n_u = [1, 2, 3]$ inputs, and $N_p = [100, 150, 200]$ steps.

From this table we can see that the DARE algorithm 6.3 scales exactly linearly with the prediction horizon N_p as expected, and slightly below the expected linear dependence with the input n_u, but overall maintaining acceptable performance. Moreover, all the cases of the proposed approach have the expected inevitable increase signaled by the pink row which compares each of the "total" cyan rows, representing the summation of each of the columns of all the algorithms used by the respective approach. Note that this increase is particularly related to the computation of DARE, the H/F matrix, and the computations of the Hessian (E) as the other computations are, for all practical purposes, negligible.

Although in all cases the approach presented a relatively large increase of +50/67%, +55/62% and +75/89%, it is important to remember that the comparison is not based on the final computation times of the approach given there are still 2 main computations missing, namely: the Forward Simulation and Linearisation of the system, and the QP Solution, as seen from the Preparation and Feedback phase algorithms, 6.8 and 6.7, respectively. An example of the actual performance obtained in the Inverted Pendulum of case study 6.4 will be discussed in table 6.6, where the total increase is presented.

As it is to be expected, the performance of these algorithms will vary from platform to platform, and could result in better "relative" performance depending on the available Floating Point Unit (FPU) on the CPU that is used. However, regardless of this, we can always do an analysis of what can we expect from a given algorithm. A relatively simple example of this is the comparison between the number of computations required for the standard H matrix (algorithm 3.5), and that of the dual mode H/F matrices (algorithm 6.4). The expected performance of both of these algorithms can be easily measured in terms of the number of "vector times vector" multiplications required. Let us begin by noting that each of the computations of $h_{k,i} = A_{k-1}h_{k-1,i}$ in algorithm 3.5 (or equivalently $h_{k,i} = \Phi_{k-1}h_{k-1,i}$ in algorithm 6.4) requires exactly $n_x n_u$ "vector times vector" multiplications of vectors of size n_x. In addition to this, algorithm 6.4 requires the calculation of $f_{k,i} = -K_{k-1}h_{k-1,i}$ which adds exactly n_u^2 "vector times vector" multiplication of vectors of the same size (n_x). Because both of these operations are inside the same for-loop limits, ie. they both require the same amount of lower-triangular off-diagonal terms, they both require exactly $\frac{N_p(N_p-1)}{2}$ terms to be calculated. As a result, the number of "vector times vector" multiplications required by each algorithm are: $\frac{N_p(N_p-1)}{2}n_x n_u$ for the standard approach, and $\frac{N_p(N_p-1)}{2}(n_x + n_u)n_u$ for the dual mode approach. Consequently, the dual mode H/F calculation will always require exactly $\alpha = \frac{(n_x+n_u)}{n_x}$ times more "vector times vector" multiplications. Note that the results of table 6.1 were not even close to this ratio, which raised the question of whether the Eigen 3 library was not optimising these operations properly, including the memory related tasks.

Thus, to validate this claim and provide further insight into potential future work of the proposed approach, these 2 algorithms (alg. 3.5 and alg. 6.4) were tested in an alternative platform: the Beaglebone Blue, running a quasi-Real-Time Debian (Linux-based) OS @ 1 GHz. This platform has a NEON floating-point accelerator, specifically designed to make efficient parallel multiply and addition operations which the Eigen 3 library is prepared to handle. As the platform has a 32-bit ARM Cortex-A8 processor, the performance is expected to be better when using floats instead of doubles. Therefore, a comparison for this was made for a system with $n_x = 4$ states, $n_u = 1$ inputs, and $N_p = [100, 150, 200]$ steps, using both double and float precision. The test codes were compiled using the (-O3) optimisation C-flag, as well as the (-mfpu=neon) flag to enable the NEON fp-accelerator instructions. All the codes were run using real-time priority (ie. chrt -r 99 ./main) and the minimum computation time of 1250 iterations was captured. The comparison is presented in table 6.2.

From table 6.2 we can see that the increase from these 2 algorithms (signaled by the pink row), is closer to the expected values (+24.5/30.1% and +12.9/26%), as opposed to the results presented in table 6.1 were increases of up to +165% were obtained. Additionally, note that in this platform the algorithms presented a significant increase when using double/float point thus justifying the applicability of the approach for its use in unstable systems using reduced numeric precision. Ultimately, these algorithms could be tailored further using Real-Time Field Programmable Gate Arrays (FPGA), potentially resulting in more accurate computation times which were outside the scope of this thesis.

Case ($n_x = 4, n_u = 1$)	Doubles			Floats		
N_p	100	150	200	100	150	200
Standard H Matrix (alg. 3.5)	2040	4693	8405	416	1152	2666
Dual Mode H/F Matrices (alg. 6.4)	2654	5847	10461	524	1300	3093
Increase (+%)	30.1	24.6	24.5	26	12.9	16

Table 6.2: H and H/F Computation Times (in μs) Comparison using Beaglebone Blue platform for a system with $n_x = 4$ states, $n_u = 1$ inputs, and $N_p = [100, 150, 200]$ steps, using doubles and floats.

6.3 Case Study: The Ball Plate System

The Ball-Plate System, which is the two-dimensional extension of the Ball and Beam system [34], is a challenging nonlinear, multi-variable and open loop unstable system commonly used in higher education laboratory experiments for teaching purposes. Its inherent complexity allows the study of interesting control problems such as: 1. point stabilisation control, ie. to carry the ball to a desired position in the plate; and 2. trajectory tracking control, ie. to make the ball follow a pre-defined trajectory such as a circle, a square, a Lissajous curve, etc. [34].

The system in question was used as a case study in [18] which forms part of the core/foundational articles that motivated the work presented in the previous chapter: the Shifting Strategy (chapter 5), particularly in section 6 of the aforementioned article. Upon testing the proposed approach of chapter 5 in this system, it became clear very quickly that it was not possible to use long horizons on this system which motivated the work of this chapter as a prerequisite method towards the combined approach presented in chapter 8. To put things in perspective, the ball-plate system from [18] has 3 integrators (poles at zero), and one unstable pole. We will see in this case study that this causes a significant numeric conditioning problem in the Hessian of the optimisation, even when using prediction horizons as low as $N_p = 20$. This prevents its solution using the standard approach in what is considered a small/fast Optimal Control Problem (OCP) which forms the core interest of applications for the proposed methodology of this chapter.

6.3.1 Modeling, Optimisation and Simulation Setup

In the aforementioned work [18], a linear time-invariant model for the x-axis of the system, ie. for the ball-beam case, was given by:

$$\dot{x} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -700 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 33.18 \end{bmatrix}}_{A} x + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ 3.7921 \end{bmatrix}}_{B} u \tag{6.39}$$

where $x = [p, \dot{p}, \theta, \dot{\theta}]^T$ is the state vector related to the x-axis, p and $\dot{p}$ are the position and velocity of the ball, θ and $\dot{\theta}$ are the angle and angular velocity of the plate, and u is the voltage of the motor that makes the plate's x-axis rotate. Note that the poles of the system, ie. the eigenvalues of matrix A are: $eig(A) = [0, 0, 0, 33.18]$ which causes the aforementioned numerical conditioning problems.

As this is a linear system, one could directly discretise the system for a given sampling time, eg. using a sampling time of $T_s = 30\ (ms)$ with a Zero-Order-Hold (ZOH) method as in [18], and proceed to use the standard Linear MPC methodologies. However, to shake things up a bit, we will modify this model slightly simply to make it a nonlinear system that would justify the use of the NMPC methods discussed in this chapter.

Based on the Euler-Lagrange differential equations from [34], there are at least 2 nonlinear terms which are not represented in the motion of the ball in 6.39, namely: the gravitational term ($mg\sin(\theta)$, and the centrifugal term ($mp\dot{\theta}^2$). Considering the provided model is a linearisation of the actual model, it is evident that if the gravitational term would be included in the original nonlinear model, the system would be represented by a Ordinary Differential Equation represented by the state space ($\dot{x} = f(x, u)$):

$$\underbrace{\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\\ \dot{x}_4\end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix}x_2\\ -700\sin(x_3)\\ x_4\\ 33.18x_4 + 3.7921u\end{bmatrix}}_{f(x,u)} \tag{6.40}$$

with $x_1 = p$, $x_2 = \dot{p}$, $x_3 = \theta$ and $x_4 = \dot{\theta}$.

Clearly, the linearisation of the nonlinear model (6.40) at the origin results in the exact same linear model (6.39) of [18]. However, this modification allows the model to be valid at steeper angles which will be used as part of the initial condition of the system. Although the centrifugal term could have also been added, there was not enough information provided in [18] to verify what would constitute an equivalent or correct coefficient. Regardless, the aforementioned modification resulted in similar responses to the ones provided in [134]. To keep the responses as close as possible to the real system, the nonlinear system was simulated and linearised using algorithm 3.1 with $N_s = 20$ intermediate steps and a sampling time of $T_s = 30\ (ms)$.

The optimisation was subject to the same penalisation weights as in [18, 134], ie. a state-error penalisation weight $q_{k+i} = diag([6, 0.1, 500, 100])\ \forall i = [1, N_p]$, and an input-error penalisation weight of $r_{k+i} = diag([1])\ \forall i = [0, N_p - 1]$. Although one could optionally use the infinite horizon terminal weight ($q_{k+N_p} = P_N$) as in [18, 134] to embed the secondary mode, this was not required to observe the benefits that result from the application of the proposed approach. Moreover, the system was subject to the following input and position constraints:

$$-20 \leq p \leq 20\ (cm) \tag{6.41a}$$

$$-10 \leq u \leq 10\ (V) \tag{6.41b}$$

The optimisation was initialised with the free-response of the system, which can be obtained simply with an initial guess for the nominal input trajectory of zeros ($\bar{U} = \mathbb{O}$), and the initial guess for the nominal state trajectory being the response obtained with the nominal input guess, ie. following a single shooting philosophy. The reference of the system was set at the origin and the resulting Optimal Control Problems (OCPs) were solved using the "quadprog" function from Matlab.

To compare the performance of the proposed approach with the standard method, the system was simulated for $T = 3$ (s) starting from the initial condition $x_0 = [17, 0, 0.4, 0]^T$, and optimised using multiple-shooting with various prediction horizons $N_p = [15, 20, 30, 60]$ which overall allowed the demonstration of key benefits and problems relevant to the proposed approach of this chapter.

6.3.2 Numeric Conditioning Comparison

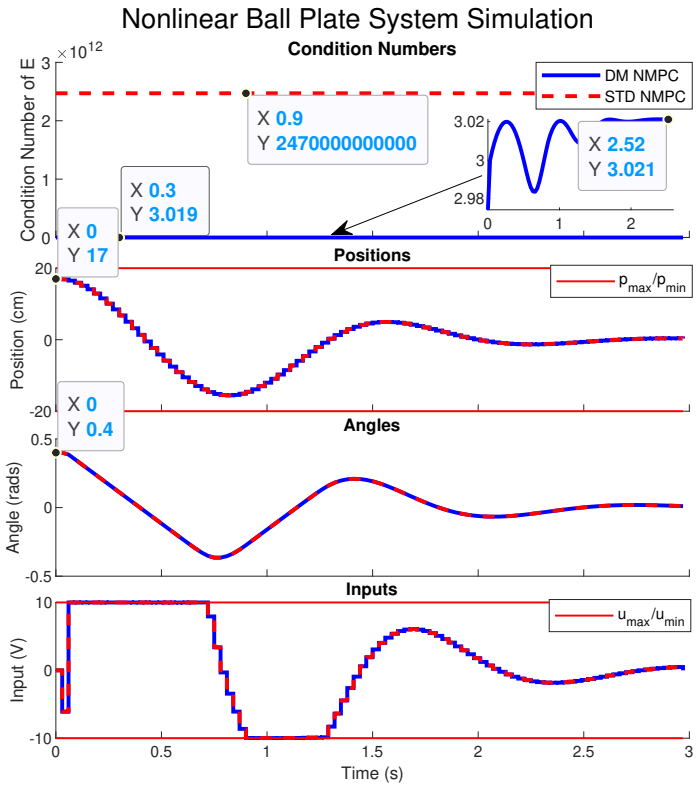
One of the most surprising parts of this system is the numeric conditioning problem that it presents. To give an idea of the severity of the ill-conditioning problem, the standard NMPC presented a numeric condition of $2.47E{+}12$ in the Hessian when using a prediction horizon as low as $N_p = 15$ and linearising the system at the origin, which would represent the steady state or "final" condition of the Hessian. Although this particular case was still solvable using the standard approach, it would present significant problems if reduced numeric precision such as floats and/or longer horizons were to be used. Indeed we will see that the optimisation resulted in numerical conditioning problems with prediction horizons as low as $N_p = 20$. For reference, table 6.3 gathers the condition number obtained for each prediction horizon at the origin of the system where the standard optimisation can be seen to reach condition numbers up to $2.38E + 50$ in what could be considered a relatively small Optimal Control Problem.

N_p	Dual Mode	Standard
15	3.021	$2.47E + 12$
20	3.025	$7.30E + 16$
30	3.277	$2.77E + 25$
60	3.295	$2.38E + 50$

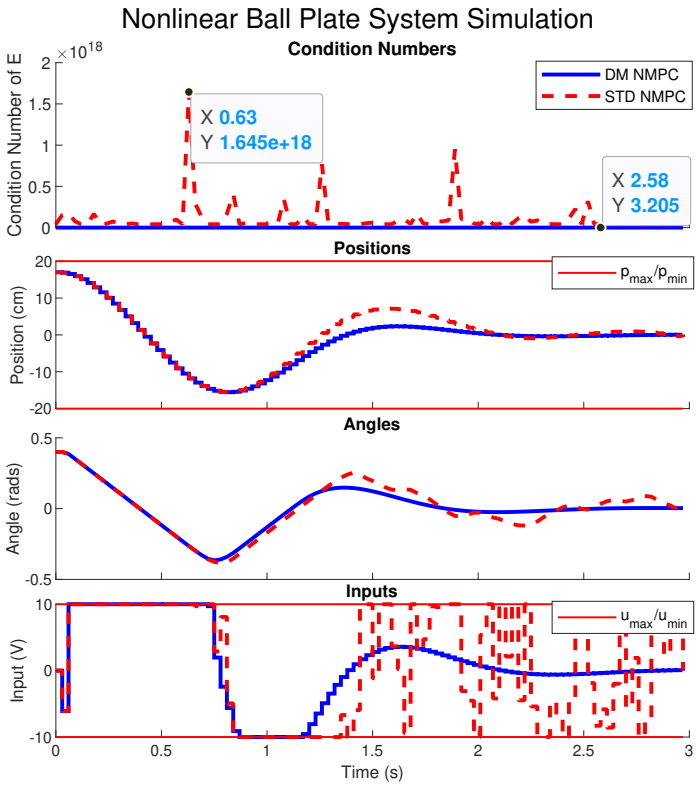
Table 6.3: Comparison of Numeric Condition Numbers of the Nonlinear Ball Plate System at the origin with prediction horizon $N_p = [15, 20, 30, 60]$

To illustrate some of the key properties of the proposed approach, figure 6.1a presents the resulting responses when using both, the Dual-Mode and Standard approaches with a prediction horizon of $N_p = 15$. From this figure we can clearly see that the responses of both approaches are exactly identical as expected from theorem 6.2. Moreover, the resulting condition numbers of each approach can be seen in the top graph of the figure where a condition number of 3.021 is achieved at the steady state condition when using the proposed approach as signaled from the inner graph, and of $2.47E + 12$ when using the standard approach, ie. resulting in a condition number $8.18E + 11$ times larger. Furthermore, the condition number of the proposed approach undergoes a transient response as seen in the inner graph which is expected as the system is linearised around the trajectory of the system as opposed to the standard approach where the system would be simply linearised at the origin.

Similarly, figure 6.1b shows the response of the system when using a prediction horizon of $N_p = 20$ were the consequences of not having an appropriate numeric condition in the optimisation can be seen. In this figure, it can be appreciated how the response obtained using the proposed Closed Loop Dual-Mode approach maintains a similar performance to that of figure 6.1a. In contrast, the performance obtained from the standard approach shows a rather erratic response, particularly in the input of the system, evidently related to the numeric conditioning problems. As expected, this problem was even worse when using longer prediction horizons, in some cases preventing the solution altogether.



(a) $N_p = 15$



(b) $N_p = 20$

Figure 6.1: Nonlinear Ball Plate System Simulation with initial condition $x_0 = [17, 0, 0.4, 0]^T$ and prediction horizons of $N_p = 15$ (6.1a) and $N_p = 20$ (6.1b). DM and STD represent the Dual-Mode and Standard approaches, respectively.

6.4 Case Study: The Inverted Pendulum

The Inverted Pendulum was a key benchmark for this chapter as it allowed not only the simulation but the experimental validation of the approach presented in the Parallel Double Inverted Pendulum experiment of section 6.6 which formed the main contribution of the IET paper [47]. Moreover, its complex nonlinear dynamic requirements formed the ideal case in which the proposed approach may be necessary (as we will show in the case study of the Triple Inverted Pendulum in section 6.5) given that the system undergoes a complete input reversal where inputs that would generate a positive moment in the lower equilibrium generate negative moments in the upper equilibrium. As a result, the user wouldn't be able to apply a "constant" feedback control law designed for the upper equilibrium as in the standard approach given it would completely destabilise the system at the lower equilibrium, potentially preventing the system from even entering the upper equilibrium region due to the ill-conditioning problem. In contrast, we will see in this case study how by applying the proposed approach, the numeric conditioning problem remains under control at all times whilst giving the exact same performance and remaining competitive in terms of computations times.

6.4.1 Modeling, Simulation and Optimisation Setup

In this case study, the same simplified model of the inverted pendulum of the case study from the previous chapter (chapter 5) with an additional position friction f_m was used which is given by:

$$\ddot{p} = f_m \dot{p} + ku \tag{6.42a}$$

$$\ddot{\theta} = a\dot{\theta} + b\sin(\theta) + c\cos(\theta)(f_m v_k + ku_k) \tag{6.42b}$$

Considering the state $x_k = [v, \omega, p, \theta]^T$ with $v = \dot{p}$ and $\omega = \dot{\theta}$ and using the Explicit Euler integration method of algorithm 3.1 with a single step ($N_s = 1$), the simplified discrete-time nonlinear model is given by:

$$x_{k+1} = x_k + T_s f(x_k, u_k) \tag{6.43a}$$

$$f(x_k, u_k) = \begin{bmatrix} f_m v_k + ku_k \\ a\omega_k + b\sin(\theta_k) + c\cos(\theta_k)(f_m v_k + ku_k) \\ v_k \\ \omega_k \end{bmatrix} \tag{6.43b}$$

where $T_s = 0.02$ (s) is the sampling time; p is the position; v is the velocity; θ is the pendulums' angle; ω is the pendulums' angular velocity; and u_k is the input of the system.

In this case, the coefficients of the system were defined as in table 6.4 in alignment with the coefficients obtained from the experiment of the double inverted pendulum case study of section 6.6.

f_m	k	a	b	c
-4.67	0.065	-0.129	38.4	3.95

Table 6.4: Inverted Pendulum Parameters

To evaluate the performance of the proposed methodology, the optimisation was done for different prediction horizons using $q_{k+i} = diag([0.1, 0.1, 10, 10])\ \forall i = [1, N_p - 1]$, and $r_{k+i} = diag([0.001])\ \forall i = [0, N_p - 1]$ to penalise the state and input errors. A terminal weight of $q_{k+N_p} = 10q_{k+1}$ was imposed in the last state of the horizon x_{k+N_p} to improve stability properties of the optimisation by emulating zero-terminal constraints. All the simulations started at the lower equilibrium in steady state $x_r = x_0 = [0, 0, 0, \pi]^T$, and a reference change to the upward equilibrium ($x_r = [0, 0, 0, 0]$) was given by introducing it at the end of the prediction horizon to achieve better performance of the RTI Scheme as discussed in [55]. Notice the required input to stabilise the inverted pendulum in the upper equilibrium is zero, thus a reference of $U_r = \mathbb{0}^{N_p n_u}$ was imposed in the inputs. Finally, constraints in the input and position were imposed as $-170 < u < 170$ and $-0.35 \leq p \leq 0.35$, respectively.

6.4.2 The Prestabilised Target

One of the most important characteristics of the proposed approach which differs from the standard prestabilisation approaches is the target that is considered for prestabilisation. In the standard prestabilisation approach, eg. the one presented in [122], the unconstrained solution is embedded on the optimisation such that when the decision variables are zero, ie. $\hat{C} = \mathbb{0}$, the response gives the optimal unconstrained solution. Such a type of prestabilisation might be more relevant if input parameterisation approaches such as the Laguerre or Chebyshev polynomials presented in chapter 4 were used as decision variables specifically to handle constraints rather than to seek optimallity as discussed in [79, 125]. However, in the case where the full degrees of freedom are kept as in the proposed approach, the solution would be the same irrespective of the prestabilisation target that is chosen as proved by theorem 6.2. In contrast, the proposed approach targets the optimal constrained solution obtained in the previous step such that if any small disturbance comes into the system it will not only cancel it (which is known to have benefits for achieving robust predictions), but also bring the solution directly back to the optimal constrained solution. It is important to note that in the ideal/nominal case where the system has converged to the global optimum, no uncertainty, noise or disturbances are present, and infinite horizon costs are used along with invariant sets, the solution to the OCP at all the future stages would be directly given by $\hat{C} = \mathbb{0}$ which can be beneficial for finding initial points in Interior Point methods, eg. as the one presented in appendix C. This is also the case when the standard approach based on the relative solution is initialised with $\delta\hat{U} = \mathbb{0}$, however the key difference is what happens when a small disturbance comes into the system which affects the overall predictions.

To illustrate the aforementioned situation, figure 6.2 presents a comparison of the "free-response" predictions of both approaches in the presence of a disturbance, ie. the predictions when the decision variables are zero. In this figure, the green dot-dashed line represents the previous nominal optimal solution, ie. the target of the proposed pre-stabilisation approach, and the blue and red-dashed lines are the "free-response" predictions of the Closed Loop Dual Mode, and Standard approaches, respectively. In this simulation a small input disturbance was introduced during the swing up phase of the optimisation as visible in the ellipse of the lower graph of the figure. As it can be appreciated, the standard approach leads to significant deviations in the "free-response" predictions of the angle and angular velocities of up to -758 and -5216, respectively as seen in the inner graphs, whereas the proposed approach quickly cancels out the disturbance and comes back to the optimal solution.

Although in this particular case the angle or angular velocities were not constrained, it would be significantly more challenging to find feasible initial points for the optimisation if the standard approach was used and these variables were constrained given the substantial violations of the free-response.

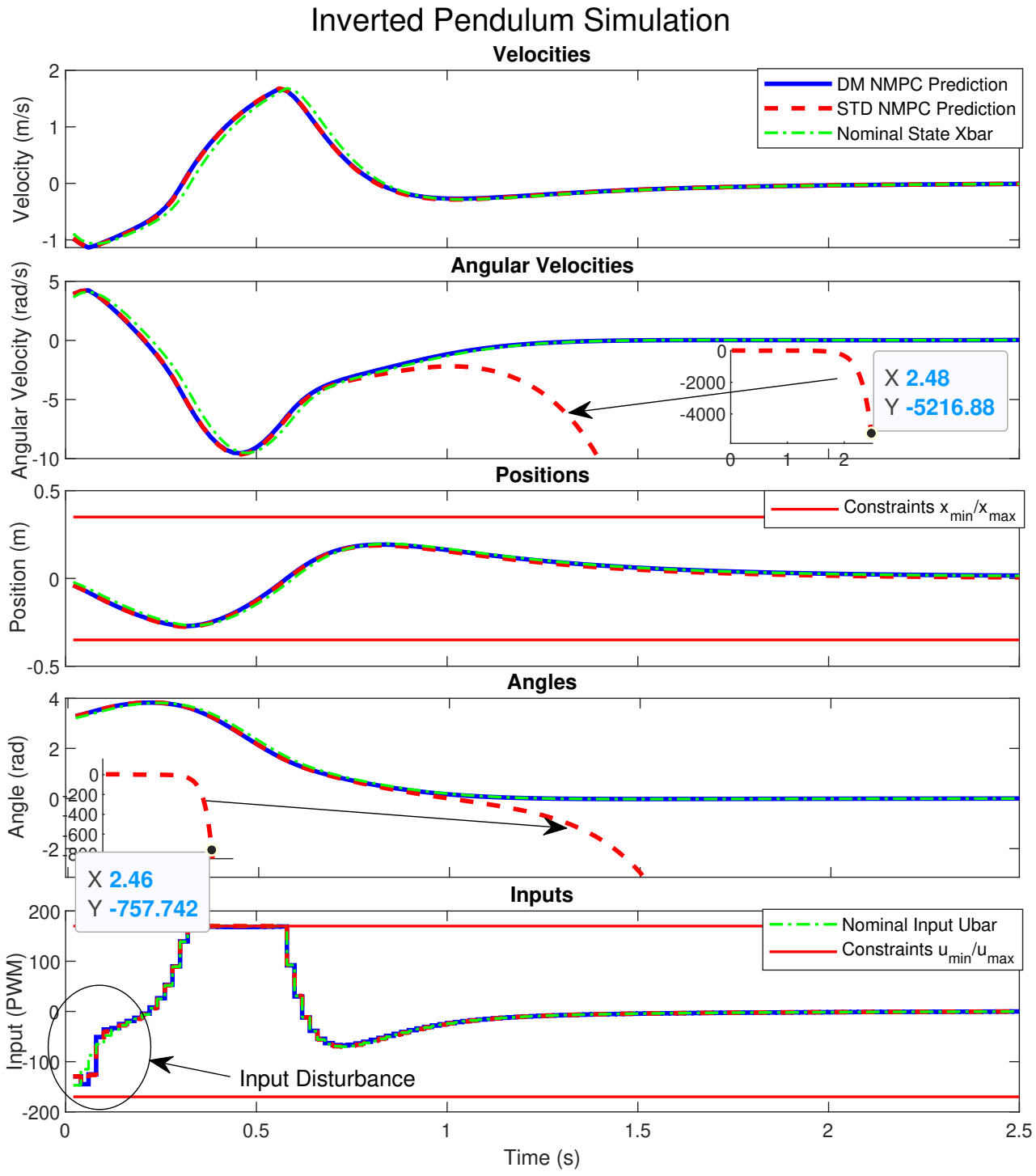


Figure 6.2: Comparison of "Free-Response" Predictions with and without Prestabilisation for the Inverted Pendulum problem during the swing up phase.

6.4.3 Numeric Conditioning Comparison

As discussed earlier, another topic of interest is the numeric condition of the optimisation. To analyse this, the condition number ($c.n$) of the Hessian E was calculated and compared between both, the standard and the proposed Closed Loop Dual Mode approaches using different numeric precision (floats and doubles), and the maximum $c.n_{max}$ of each solution was gathered in table 6.5 for all prediction horizons.

Precision	Double		Float	
N_p	DM	STD	DM	STD
75	3.58	1.39e+06	3.58	2.28e+06
100	3.58	4.52e+08	3.58	(Singular)
125	3.58	1.47e+11	3.58	(Singular)
150	3.58	5.02e+13	3.58	(Singular)

Table 6.5: Maximum Condition Numbers Comparison for Different Prediction Horizons and Numeric Precision. STD and DM refer to the standard and dual mode solution, respectively.

To visualise this differences, an example performance of the optimisation is given in figure 6.3 for the solution with prediction horizon $N_p = 75$ where the $c.n.$ is plotted for both solutions along with the resulting trajectories. As it can be seen, the solutions of both approaches are exactly the same as expected from theorem 6.2, however the standard solution gives a condition number of up to $c.n. = 1.39e{+}06$, resulting in a difference between both solutions of nearly 6 orders of magnitude larger, which is fairly significant considering the relatively short prediction horizon used. Looking further at table 6.5, the condition number of the standard solution increased as the prediction horizon increased giving differences of up to 13 orders of magnitude for $N_p = 150$, with the Hessian becoming singular for $N_p > 75$ when using float precision. This ultimately prevents the standard methodologies from using floating precision which can lead to faster computation as discussed earlier in the generic computations section 6.2. In contrast, the proposed solution maintained steady at $c.n_{max} \approx 3.58\ \forall N_p$ independent of the numeric precision. It should be mentioned that common prediction horizons for the inverted pendulum are relatively long (2 to 4 seconds [55, 104]), which is approximately the time required to swing up and stabilise the system. However, other systems can present numeric conditioning problems in as low as 1 second, as we have already shown in the previous ball-plate case study of section 6.3, for which the proposed approach offers a viable solution.

Remark 6.5. *It should be noted that the numerical conditioning problem of this or any system can be changed by selecting a different sampling time, prediction horizon, number of intermediate steps of the discretisation and linearisation process, etc. [55]. However, this doesn't tackle the source of the problem, nor does it provide a general methodology to address it using an arbitrary/desired number of elements to be selected by the user.*

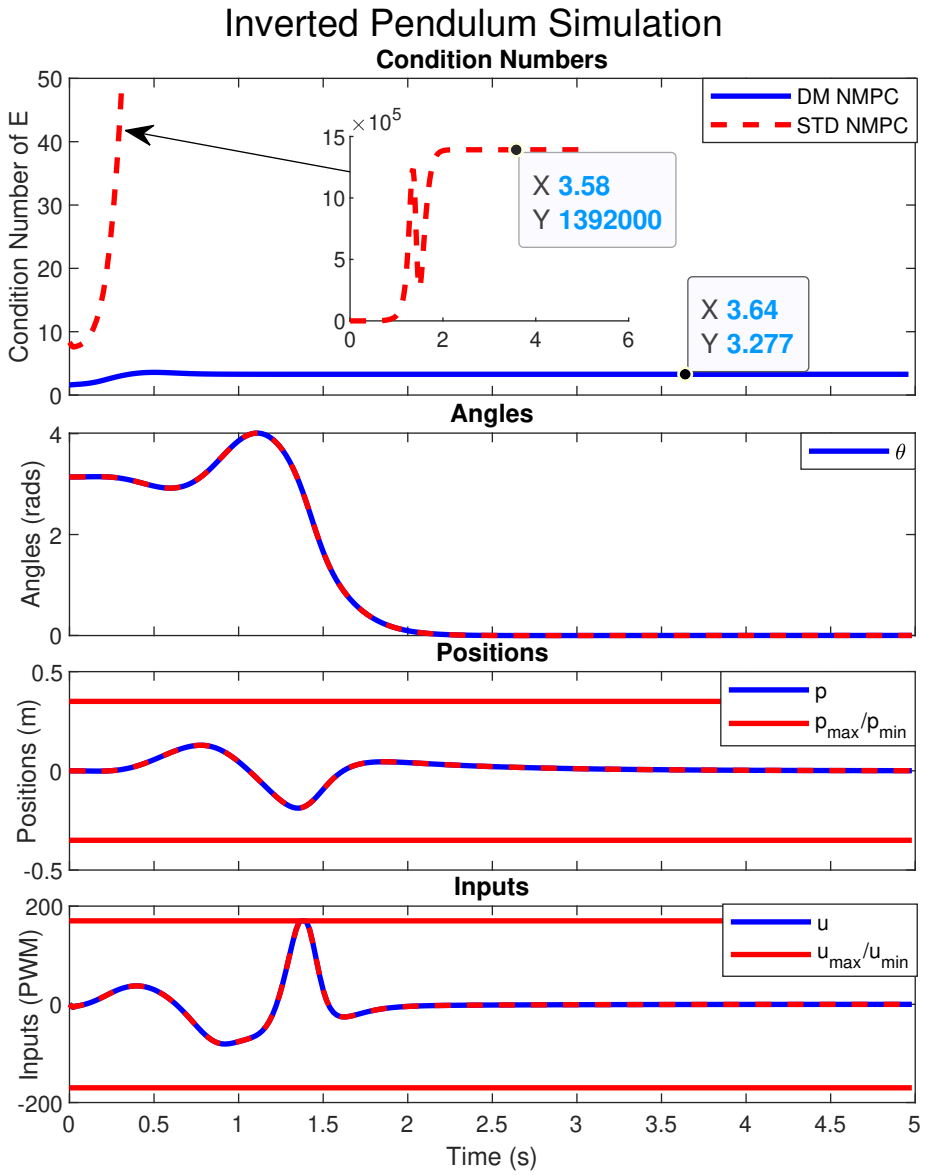


Figure 6.3: Comparison of Numeric Conditioning of Standard and Dual Mode NMPC using double precision for the Inverted Pendulum problem with a prediction horizon of $N_p = 75$, and an initial condition of $x_0 = [0, 0, 0, \pi]^T$. STD and DM refer to the standard and dual mode solution, respectively.

6.4.4 Disturbance Rejection Comparison

Another interesting result was obtained when comparing the responses against disturbance rejection which were observed to present small differences despite the equality of the solutions proven by theorem 6.2. This was particularly present when using long horizons but more importantly, when using the inverse function "inv(A)" of Matlab to obtain the unconstrained solution, which is known to be slower and less accurate than solving a linear system using $A\backslash b$. To test this, a disturbance of $x_k = x_k + [0, 0.5, 0, 0]^T$ was injected at $t = 7$ (s) (continuation from figure 6.3 - system in upper equilibrium) for which the unconstrained solution satisfies.

Figure 6.4 shows an example of the aforementioned situation where the predicted and closed-loop responses are plotted after the disturbance is injected. Only the initial predicted trajectories were plotted to avoid saturation. As it can be seen, the predicted trajectories of the angle using the standard solution (magenta dotted line - visible in the upper right corner of the upper graph) diverged significantly from the closed loop, which in essence resulted in an ill-posed optimisation [122] and caused the closed loop solution (red dashed line) to differ as it can be seen from all 3 responses (angles, position and inputs). In contrast, the predictions of the angle using proposed dual mode approach (cyan dash-dotted line - visible in the lower left corner of the upper graph) are indistinguishable from the closed-loop response (blue solid line). Interestingly, the closed-loop responses were identical before the introduction of the disturbance, which suggest that this problem is clearly related to the numeric conditioning of matrix G whose norm grows as big as $||G|| \geq 2.49e+08$, thus affecting the linear term f significantly when $||\delta x_0|| >> 0$. However, it is noted that this anomaly ONLY happened when using the aforementioned inverse function, and it was not present when using the command $A\backslash b$ to obtain the unconstrained solution, resulting in the exact same responses as expected from theorem (6.2). Nonetheless, it shows an example of another advantage that the proposed method can provide.

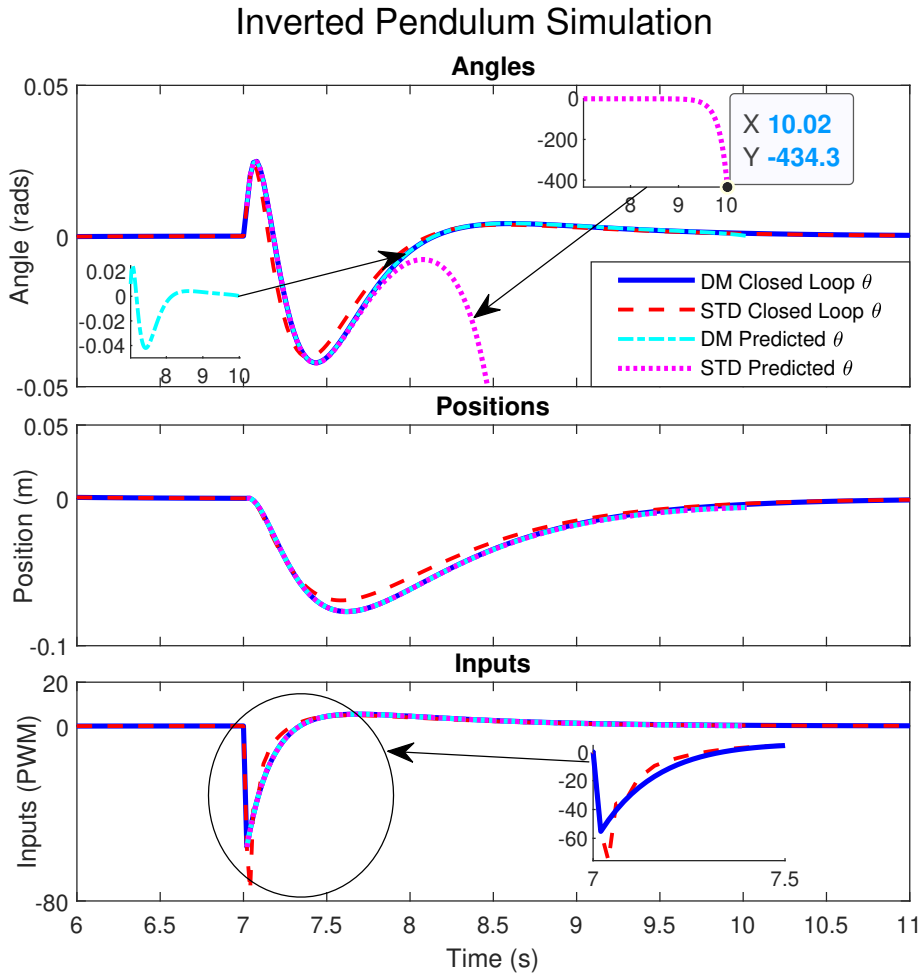


Figure 6.4: Disturbance Response - Prediction and Closed Loop Comparison for $N_p = 150$ when using Matlab "inv(A)" function. Disturbance of $x_k = x_k + [0, 0.5, 0, 0]^T$ was injected at $t = 7$ (s). STD and DM refer to the standard and dual mode solution, respectively.

6.4.5 Computation Times Comparison

In order to evaluate the computational performance of the proposed approach in this system, we developed a set of auto-generated C++ codes based on the Eigen 3 library using the RTI algorithms 6.8 and 6.7 which implemented the approach in the Inverted Pendulum system. In order to reuse the developed auto-generation routines for the tests performed in the Inverted Pendulum case study of section 5.5, three main differences were implemented in this particular test, namely: different model parameters ($a = -0.3$, $b = 9.8065$, $c = 1$, $f_m = 0$ and $k = 10$), different constraints ($-1 \leq u \leq 1$ and $-1 \leq p \leq 1$) and different weights ($q_{k+i} = diag([0.01, 0.01, 1, 1])$ $\forall i = [1, N_p - 1]$, $q_{k+N_p} = 10q_{k+1}$, $r_{k+i} = 10$ $\forall i = [0, N_p - 1]$), with the rest of the simulation specifications being identical, eg. same sampling time, same initial condition at the lower equilibrium and reference to the upper equilibrium introduced at the end of the prediction horizon. For comparison, the solution was tested against the standard RTI algorithms 3.9 and 3.8, as well as against the solutions obtained via the ACADO toolkit. Each of these algorithms was tested for different prediction horizons $N_p = [100, 150, 200]$ and the solutions of the resulting QPs were obtained using QP OASES [37, 38] which were verified to match in all cases from Matlab simulations, to developed C++ codes, to the ACADO toolkit. For further comparison purposes, the developed Dual Mode and Standard algorithms also calculated 10 iterations of the Interior Point method presented in the appendix C, particularly by repeatedly solving system (C.2) via the efficient solution given in (C.3). In this case (Interior Point), the interest was not on obtaining an optimal solution, but on quantifying how fast the algorithms would run when requiring to solve system system (C.2) 10 times, independently of whether the solution was found or not. This allowed a more "generic" measurement of the required computation times in the context of Interior Point methods. Each of these cases was run for 1000 simulations of $T = 10$ (s) giving a total of $400,000$ optimisations per case. The codes were run using the same conditions as in the generic computations comparison of section 6.2.4, ie. same laptop running with real-time priority with the same optimisation flags. The resulting average computation times of the constrained iterations of each of these approaches is presented in table 6.6 signaled by the pink cells, with the computational increase of the proposed Dual Mode compared to the Standard approach signaled in the yellow cells. For reference, the average computation times related to the "preparation" steps of each algorithm (eg. Forward/DARE/Matrices from lines 2-8 of algorithm 6.7) are also presented in table 6.6 signaled by the cyan cells, with the "total" preparation time obtained with the ACADO toolkit captured entirely in the "Matrices" row.

Type	Dual Mode			Standard			ACADO		
N_p	100	150	200	100	150	200	100	150	200
Forward	7	12	15	7	11	14	-	-	-
DARE	5	7	9	-	-	-	-	-	-
Matrices	48	111	248	35	76	177	80	173	306
QP OASES	1330	3080	7266	1111	2823	6678	1154	2863	6779
Increase (+%)	19.8	9.1	8.8	-	-	-	-	-	-
10 Int.Point steps	2973	8486	17828	2774	8226	17188	-	-	-
Increase (+%)	7.1	3.2	3.7	-	-	-	-	-	-

Table 6.6: Average constrained computation times for the Inverted Pendulum using different methods (Dual Mode, Standard, ACADO), with different prediction horizons $N_p = [100, 150, 200]$.

As it can be seen from table 6.6, the QP OASES solution obtained with the Standard approach remains remarkably close to the solution obtained with the ACADO toolkit, to the point of the solution with the developed Standard auto-generated solution being slightly faster in all the cases. We take this as an opportunity to demonstrate again that the auto-generation routines based on the Eigen 3 library developed throughout this Ph.D. resulted in even better computation times than those obtained with the ACADO toolkit itself when using the Standard NMPC approach. This was also seen in the comparison of case study 5.5, particularly in table 5.12.

On the other hand, the proposed approach can be seen to remain competitive w.r.t. to the standard approach with computational increases ranging from $+8.8/+19.8\%$ in all the QP OASES cases, with the computational increase decreasing as the horizon increases, as it was seen in some parts of generic computations comparison of table 6.1. As discussed earlier, this increase is inevitably related to the extra preparation computations visible mainly in the DARE/Matrices steps of the preparation phase, as well as the requirement for extra "output" type of constraints which may be handled inefficiently by the QP OASES algorithms. Nevertheless, we believe this increase is reasonable given the generic ability of the proposed approach to handle unstable non-linear systems. Interestingly, the "total" preparation times of all the Dual Mode approach solutions resulted in better performance that the preparation times of the ACADO toolkit (eg. $48+5+7=60<80$ for $N_p=100$), which demonstrates the increased efficiency obtained with proper implementation of the proposed extensions of the $O(N)/O(N^2)$ algorithms. Moreover, the total computational increase was much better than the preparation related operations increase, eg. the preparation steps of $N_p=200$ using both Standard/Dual Mode approaches resulting in a $(248+9+15)/(14+177)=+42\%$ increase, whereas the total solution with QP OASES resulting in an increase of only $+8.8\%$. Recalling the discussion from table 6.2, the total computational increase could improve if the algorithms would result in the "expected" performance, eg. if they were applied in actual real-time systems such as Micro-controller Units (MCU) or FPGAs which unfortunately were outside the scope of this thesis. Thus, this is considered an area of opportunity for future work to evaluate the actual performance that could be obtained with the proposed approach. Lastly, the approach presented comparatively much better computational performance when considered in the context of its implementation using the Interior Point method of appendix C. In this case, the proposed approach resulted in computational increases in the range of only $+3.2/7.1$ which again, we believe its justified considering the general advantages that come with the proposed method.

Thus, this case study provides a comparative example of the performance that can be obtained when implementing the proposed approach. Based on this evidence, we believe the proposed approach has enough benefits and enough reason for it to be considered for an actual implementation on real systems. Moreover, although the approach was not particularly required for this system, it could serve as an alternative in the case were reduced precision was required to be used given the solution using floats was observed to result in singular Hessian as seen in table 6.5. This could again result in further computational benefits when compared to the Standard solution which would require double precision for it to be able to be implemented.

6.5 Case Study: The Triple Inverted Pendulum

To further illustrate the benefits of the proposed methodology and provide a more complete example that further shows its generalisation capabilities for higher order systems, this section presents its application to a triple inverted pendulum which is a considerably more complex nonlinear system than the single inverted pendulum. Indeed, due to its highly unstable dynamics, the standard condensing based multiple shooting NMPC approach was unable to solve this problem altogether, independently of the prediction horizon used. Thus, this provides an example of a problem that previously was unable to be solved using the latter which further stresses out the importance of the contribution.

6.5.1 Modeling, Simulation and Optimisation Setup

In this thesis, the equations of motion for a point-mass triple pendulum provided in [120] were used, combined with the cart acceleration differential equation (6.42a) with the assumption that the pendulums will have no effect on the cart. This assumption is standard in many approaches present in the literature as the pendulums' effects can be cancelled using subordinate/inner acceleration/velocity controllers for the cart as described in [6, 45].

The equations are given by:

$$\ddot{p} = f_m \dot{p} + ku \tag{6.44a}$$

$$M(\theta)\ddot{\theta} = -N(\theta)\dot{\theta}^2 - R\dot{\theta} - P(\theta) - f(\theta)(f_m\dot{p} + ku) \tag{6.44b}$$

where $M(\theta)$, $N(\theta)$, R, $P(\theta)$ and $f(\theta)$ are matrices defined as in [120], and the specific parameters used for our simulation are given in table 6.7. Assuming the state $x_k = [v, \omega_1, \omega_2, \omega_3, p, \theta_1, \theta_2, \theta_3]^T$ with $v = \dot{p}$ and $\omega_i = \dot{\theta}_i$, the system was simulated and linearised using $N_s = 2$ steps of Explicit Euler algorithm 3.1 with a sampling time of $T_s = 0.02(s)$. The inner step was required to improve the accuracy and stability of the integration method as the system is known to present highly chaotic behaviour [120]. Moreover, given the complexity of the system, Matlab's Symbolic Toolbox was used to obtain the expressions of the linearisation terms.

m_1	0.3	L_1	0.3	R_1	0.1	g	9.81
m_2	0.27	L_2	0.27	R_2	0.1	f_m	-4.67
m_3	0.243	L_3	0.243	R_3	0.1	k	0.065

Table 6.7: Triple Inverted Pendulum Parameters

Regarding the optimisation setup, a prediction horizon of $T_p = 2$ $(s)(N_p = 100)$ was used, and the penalization weights were selected as $q_{k+i} = diag([0.1, 0.2, 0.3, 0.4, 10, 20, 30, 40])$ $\forall i = [1, N_p - 1]$ with the terminal weight selected as $q_{k+N_p} = 100q_{k+1}$, and the input penalisation term as $r_{k+i} = diag([0.001])$ $\forall i = [0, N_p - 1]$. As in case study 6.4, all the simulations started from the lower equilibrium in steady state ($x_r = x_0 = [0, 0, 0, 0, 0, 0, 0, 0]^T$), and a reference of $x_r = [0, 0, 0, 0, 0, -\pi, \pi, -\pi]^T$ was introduced at the end of the prediction horizon. Moreover, to relax the optimisation (as it is indeed a much more difficult problem), the position was constrained to $-0.5 \leq p \leq 0.5$ whilst keeping the same input constraints as for the single inverted pendulum of section 6.4, ie. $(-170 \leq u \leq 170)$.

To further improve the performance of the underlying SQP method, an additional exponentially decaying input penalisation term defined as $\delta(r)_{k+i} = 1000 r_{k+i}(\alpha)^i \ \forall i = [0, N_p - 1]$ with $\alpha = (0.01)^{\frac{1}{N_p}}$ was imposed on the input deviation $\delta\hat{U}$ which modified the original cost function (3.1) to:

$$J_{\delta R} = J + \left(\delta\hat{U}\right)^T \delta R \left(\delta\hat{U}\right) \tag{6.45}$$

where $\delta R = diag([\delta(r)_{k+i}]) \in \mathbb{R}^{N_p n_u \times N_p n_u}$, which modified the Hessian E and linear term f to:

$$E_{\delta R} = H^T Q H + F^T (R + \delta R) F \tag{6.46a}$$

$$\begin{aligned} f_{\delta R} = -\big[& H^T Q (X_r - \bar{X} - D - G\delta x_0) - \\ & F^T R(\bar{U} - U_r) - F^T (R + \delta R)(S + W\delta x_0)\big] \end{aligned} \tag{6.46b}$$

Although the performance can also be improved by using proper step-size of the Newton-method as discussed in the improvement methods of section 3.4, this additional term was motivated by observing that the prediction errors due to linearisation typically grow as they move forward through the horizon, as discussed in example 3.1 of section 3.1.2 where the autonomous/auto-correcting feature of multiple shooting was presented. Therefore, by preventing large deviations at the beginning of the horizon, the prediction errors in future time-steps are reduced which consequently improves the contraction rate of the underlying Gauss-Newton method. On the other hand, it can be proved that the solution with this added penalisation term only affects the rate of convergence towards the solution (from a fixed-in-time optimisation point of view), but does not change the solution itself. Finally, it is trivial to show that theorem 6.2 still holds with this modification, ie. if the additional exponential weight was imposed in the standard approach, it would result in the exact same solution as in the proposed approach.

6.5.2 Numeric Conditioning

Figure 6.5 shows a $T = 10(s)$ simulation of a swing-up and stabilisation of the triple inverted pendulum with a disturbance of $x_k = x_k + [0, 0, 0, 0.1, 0, 0, 0, 0]^T$ introduced at $t = 5$ (s) for which the unconstrained solution satisfies. Of particular interest is the figure on the lower-right corner where the steady state condition number $(c.n.)_{ss} \approx 252$ can be seen which, for this system, is naturally much higher than that of the single inverted pendulum presented in figure 6.3. Indeed, the latter undergoes critical points during the swing up reaching a maximum of $(c.n.)_{max} \approx 811$, approximately 3.2 times higher, which once again shows the complexity of the system at hand. Nonetheless, the method preserves the expected properties of low conditioning number which protect the solution from numerical instability, and the resulting controller is observed to perform well against disturbances. Moreover, it can be seen how the proposed approach follows the standard solution (as expected from theorem 6.2) up until the point in which the numeric condition of the standard approach "explodes" at around $T = 1.8$ (s) reaching $c.n. \approx 10^{27}$ where a numeric solution was no longer able to be obtained before the system even entered the linear zone. It is worth noting that linearising the system at the upward equilibrium with the standard approach resulted in $(c.n.) = 3.07 \times 10^{23}$ of the "would-be" optimisation.

For the interest of the reader, figure 6.6 shows an animation of the inverted pendulum trajectory, and a video of its evolution can be found in (https://www.youtube.com/watch?v=rMKrCKjvCtA).

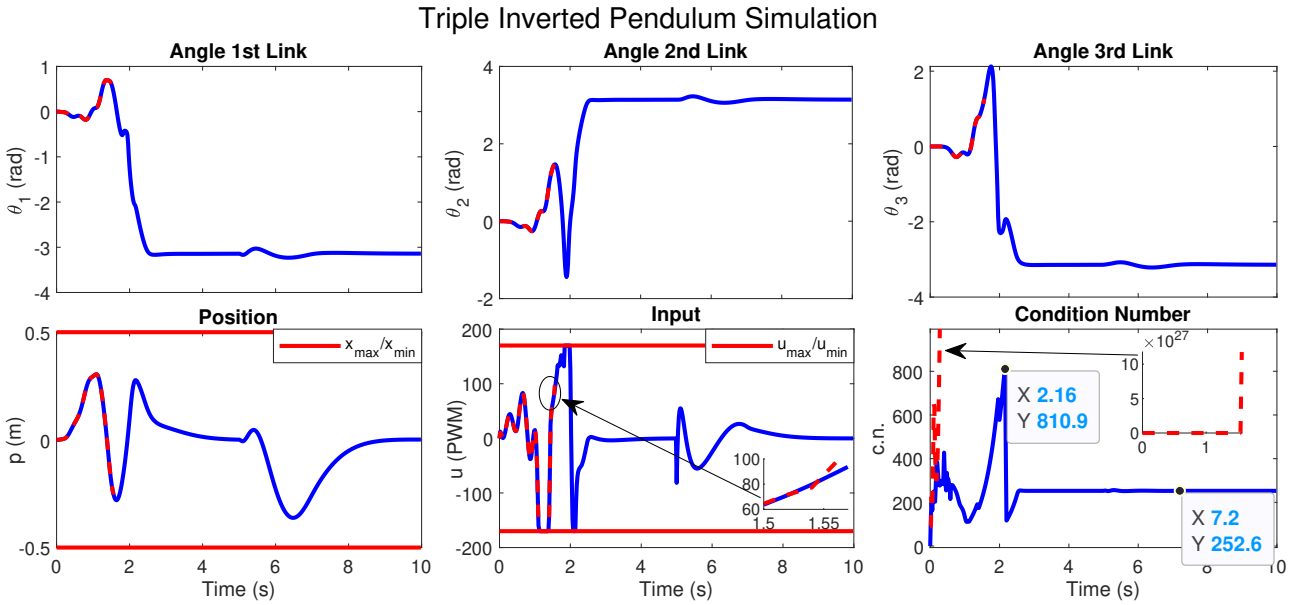


Figure 6.5: Condition Number Comparison for the Triple Inverted Pendulum swing up and stabilisation simulation with disturbance of $x_k = x_k + [0, 0, 0, 0.1, 0, 0, 0, 0]^T$ injected at $t = 5$ (s).

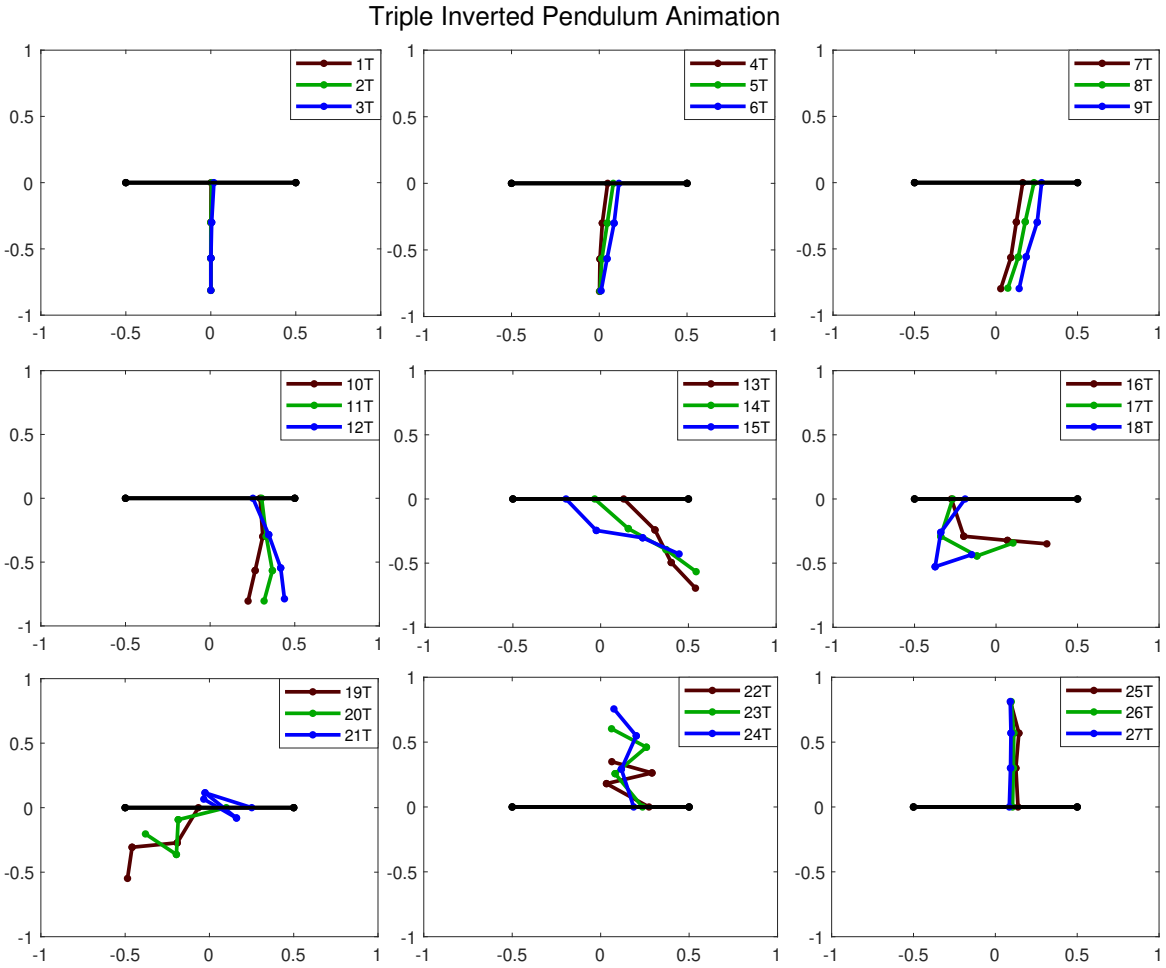


Figure 6.6: Animation of Triple Inverted Pendulum Simulation re-sampled at $T = 0.1$ (s).

6.6 Experimental Case Study: The Double Inverted Pendulum

Based on the author's experience and interest in Inverted-Pendulum-type systems, an experimental validation of the general pre-stabilisation approach was performed in a Parallel Double Inverted Pendulum which formed the main contribution of the IET paper [47] along with two main modifications which improved the performance of the RTI Scheme in the presence of disturbances, namely: additional energy based costs, and a hybrid switching scheme. Moreover, the approach used a slightly different discretisation scheme to that of the standard Explicit Euler of algorithm 3.1 and combined the implementation with an Extended Kalman Filter (EKF) for state estimation, and with the Recursive Least Squares (RLS) algorithm for OSI (OSI).

Ideally, a more complex system such as the Triple Inverted Pendulum from the previous case study (section 6.5) in which the approach proposed in this chapter is absolutely essential would have been preferred. However, such systems, eg. the variant "B" offered by REX Controls www.rexcontrols.com have a price of up to 36,000 euros which was completely outside the scope of the available funding. Instead, a relatively small modification was made to a fairly outdated Single Inverted Pendulum system available in the ACSE Department of the University, mainly with the addition/replacement of a couple of encoders, as well as the acquisition of a DC Motor driver and a Micro-Controlling Unit (MCU) which allowed the development top-to-bottom of the control system for the experiment.

As this case study was used entirely for the IET publication [47], some of the contents introduced here are inevitably repeated in the paper. In some cases, the reader will be referred to specific sections of the paper to avoid redundancy and keep the focus on the main contributions.

6.6.1 Modeling

Considering a parallel double inverted pendulum as depicted in figure 6.7 and assuming that the pendulum's will have negligible effect on the cart as in previous case studies, the equations of motion given in [6] can be used with additional friction terms to include the relevant energy dissipation properties.

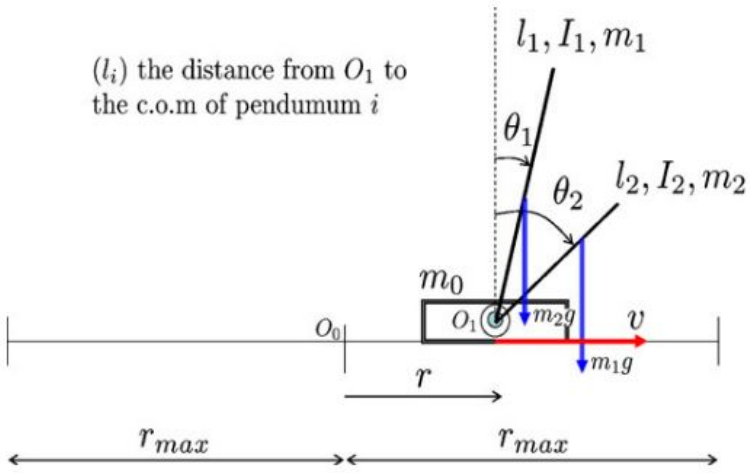


Figure 6.7: Parallel Double Inverted Pendulum Diagram from [6]

The system dynamics are then represented by:

$$\ddot{p} = f_m \dot{p} + ku \tag{6.47a}$$

$$\ddot{\theta}_i = a_i \dot{\theta}_i + b_i \sin\theta_i + c_i \cos\theta_i \left(f_m \dot{p} + ku\right) \tag{6.47b}$$

$$\forall\ i = [1, 2]$$

where p is the position of the car; u is an input signal to the system which in this case is a Pulse Width Modulated (PWM) signal for the motor driver; the i index represents the variable or parameter related to the i_{th} pendulum; k is a constant that relate the PWM with the force and mass of the system; θ_i are the angles of the i_{th} pendulum; f_m and a_i are viscous friction constants; $b_i = (m_i l_i g)/(m_i l_i^2 + I_i)$ are the pendulums' gravity related terms; $c_i = b_i/g$ are the acceleration-torque related constants; g is the gravity constant; and m_i, l_i, I_i, are the mass, length and moment of inertia of each pendulum, respectively.

In this work the relevant coefficients of the model were found by an Online System Identification (OSI) algorithm based on the Recursive Least Squares (RLS) approach presented in section 5 of the paper [47]. Note that the sign of certain coefficients might be subject to the specific experimental setup depending on orientation, e.g. positive cart motion to the left or positive angle rotation CCW, however, all the viscous friction constants (f_m and a_i) must always be negative. Moreover, it should be noted that although counter-intuitive, the lengths of the arms should be different to achieve better controllability of the system [6], particularly in the presence of noise which causes a significantly increasing amount of input shattering as the lengths become closer. This was validated through simulations to select appropriate length differences for our particular system.

Model (6.47) was valid for our particular experimental setup given two conditions were true: both pendulum's masses are much lower than the cart, and the motor driver used has a regenerative breaking feature which further cancels out any possible uncontrolled movement of the cart. In the case where the cart motion is indeed affected by the pendulum's motion, a subordinate controller can be developed to cancel this effects as in [45], or the full nonlinear model can be included in the general NMPC framework as it has been shown in [66].

As with all the methods presented in this thesis, we will use a "direct approach" which requires us to "first discretise, then optimise" the system [65]. Thus, we now look to discretise the equations of motion (6.47a) and (6.47b) which will allow us to simulate and linearize the system for both NMPC and EKF frameworks. For this work, the Explicit Forward Euler method of algorithm 3.1 was considered at first, following similar works as in [45, 104], however, based on the observation that only position and angles are measured by the system, this scheme was modified to a Forward-Backward Euler scheme. More over, given that the position model (6.47a) represent a linear system, the discretisation was modified to include an extra previous input u_{k-1} based on the fact that it can be exactly represented by a standard 2_{nd} order discrete-time model which would have 2 previous input terms if the system was discretised using Zero Order Hold methods. This was achieved by augmenting the state as seen in (6.48) to obtain a Non-Minimal State Space (NMSS) [146]. For more details regarding this modification, please refer to section 2.2 of the paper [47].

The resulting discrete-system dynamics are given by:

$$x_{k+1} = x_k + T_s f(x_k, u_k) \tag{6.48a}$$

$$f(x_k, u_k) = \left[f_{up}^T, f_{down}^T, (u_k - u_{k-1})/T_s \right]^T \tag{6.48b}$$

$$f_{up} = \begin{bmatrix} \dot{p}_k + T_s f_1 \\ \dot{\theta}_{1_k} + T_s f_2 \\ \dot{\theta}_{2_k} + T_s f_3 \end{bmatrix} \quad f_{down} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} \tag{6.48c}$$

$$f_1 = f_m \dot{p}_k + k_1 u_k + k_2 u_{k-1} \tag{6.48d}$$

$$f_2 = a_1 \dot{\theta}_{1_k} + b_1 \sin\theta_{1_k} + c_1 \cos\theta_{1_k} f_1 \tag{6.48e}$$

$$f_3 = a_2 \dot{\theta}_{2_k} + b_2 \sin\theta_{2_k} + c_2 \cos\theta_{2_k} f_1 \tag{6.48f}$$

where T_s is the sampling time. It is noted that the last term of function (6.48b) was only used to represent the propagation of the input $x_{7_{k+1}} = u_k = u_{k-1} + T_s(u_k - u_{k-1})/T_s$ and doesn't represent, in any way, a "derivative of the input" $\dot{u}_k = (u_k - u_{k-1})/T_s$ which would lead to a completely different integration scheme if more intermediate steps were computed [55].

This discretisation differs slightly from the standard Explicit Forward Euler method in the sense that the latter would compute a Forward Euler step on the higher derivative states (p, θ_1, θ_2). However, as stated previously, given only position and angle's measurement were available, the Backward Euler approximations were used instead to use the latest information of the system. Moreover, the standard Explicit Forward Euler approach would only take into account u_k for simulation purposes. Nonetheless, this modification was observed to produce much better predictions in an offline analysis of the system identification process for both, position and angle dynamics. In fact, increasing the number of previous input terms u_{k-j} was be able to improve them even further, possibly given that it accommodates some unmodelled higher order motor dynamics which are known to be at least $2nd$ order in the angular velocity; $3rd$ order in the angular position; which can be accounted for using convolution/FIR models. However, the system was observed to get good performance whilst only including 2 previous input terms, u_k and u_{k-1}.

6.6.2 A Modified Optimisation: Additional Energy Terms and Hybrid Approach

A major issue with the RTI Scheme is that the solution might give very poor performance whenever an abrupt change is made, e.g. when there is an abrupt change in the reference of the system [55], or a large fault or disturbance enters the system, which may lead to leaving the region of contraction of the Gauss-Newton method and in some cases may even lead to instability of the system [65]. In these cases, the previous solution will not be close to the optimal, and therefore, the method would need to quickly find a suitable correction from the previous solution. This issue may be addressed by adding suitable end weights and other regularity conditions [65] and convergence improvement methods as discussed in chapter 3. However, to address this issue, a different approach was taken in this work with two main modifications to the standard approach of NMPC of an inverted pendulum such as [55, 66, 73, 104], namely: an additional energy based cost; and a hybrid switching scheme.

Energy Based Costs

Motivated by the fact that a common strategy for the swing up of the pendulum are Energy Lyapunov based control laws [6], along with the fact that standard cost terms defined for inverted pendulum NMPC, e.g. angle errors [55, 104], do not actually capture the requirement of "swinging-up" but rather a more restrictive cost requiring the optimisation to drive the angles to a desired angle reference without considering other upward equilibrium points, the outputs and references typically used for the standard output cost function (3.2) were modified to include two extra terms related to the potential energy of both pendulums $E_{\theta_i} = \cos\theta_i$ as:

$$y_k = \left[\dot{p}, \dot{\theta}_1, \dot{\theta}_2, p, \theta_1, \theta_2, \cos\theta_1, \cos\theta_2\right]_k \qquad \text{Outputs} \qquad (6.49a)$$

$$y_{r_k} = \left[0, 0, 0, p_r, \theta_{1_r}, \theta_{2_r}, \cos\theta_{1_r}, \cos\theta_{2_r}\right]_k \qquad \text{References} \qquad (6.49b)$$

Remark 6.6. *With this modification, the optimisation has $n_y = 8$ outputs per step, instead of only 6 as it would be with the standard approach.*

Some of the relevant properties of this added terms are:

1. Boundedness: The error $e_k = \cos\theta_{1_r} - \cos\theta_1$ is always bounded at $e_k = [-2, 0]$ for the upright position, and $e_k = [0, 2]$ for the downside position. This in general would make the nominal output error $(Y_r - \bar{Y})$ related to the additional terms of the linear term (f) bounded.

2. Singularity: The derivative w.r.t. the added term $(C_i = -\sin\theta_i)$ required by the linearised predictions (3.21) has a singularity at any $N\pi$ multiple given the sensitivity matrix is zero. Thus, if the system is at a steady condition, e.g. all other errors zero, the optimisation would have no sensitivity on it, therefore, not reacting or causing any movement. Although the system can be confined inside an incorrect singularity, if the system is started at any other sufficiently non-singular point, the optimisation will eventually drive the solution to the desired singularity.

By penalising the energy term much higher than the angles directly, the optimisation is more relaxed, essentially aiming to drive the potential energy of the pendulum to the desired state $E_{\theta_i} = \cos\theta_i \to \cos\theta_{i_r}$ whilst accepting swinging up in either directions. This is because if at a given time the system cannot swing the pendulums up in a given direction, the optimisation would naturally select the other direction which is not the case when the standard costs are used, and the solution was observed to be severely affected when the system reached this in-feasibility condition. After a series of simulations it was concluded that imposing higher penalties on this added terms instead of the angles directly, resulted in much better convergence properties than when using the standard cost, which in turn resulted in a larger region of contraction of the Gauss-Newton method. Moreover, notice the nominal stability of the resulting scheme can still be guaranteed by using zero-terminal constraints even though the sensitivity at that condition dissipates, which in turn leads to having the original problem once the system has reached the terminal region.

To visualise the benefits of this approach, a comparison simulation is given in figure 6.8 where the predicted and closed-loop trajectories are plotted, with and without focusing on the added energy term. For clarity, the predicted responses that presented the erratic behaviour are signaled.

As it can be seen from figure 6.8b, the optimisation penalising only the angles presented mayor erratic behaviour in the closed loop input response at times $0.5 < t < 1$, and significant differences between predicted and closed-loop responses in general leading to ill-posed optimisations [122]. In contrast, the optimisation that focused effort on the added energy-cost (figure 6.8a) presented smooth and much better overall closed-loop performance.

Although this approach might not be immediately generalisable for other control systems and applications, it is very common to find trigonometric terms in robotics systems and mechatronic applications that arise from rotation matrices. This naturally brings the question of whether it is better to target a desired potential energy or an angle directly when dealing with multi-link robots. Indeed, it is well known that the understanding of the inverted pendulum dynamics helped with the development of many robotic applications nowadays, thus its generalisation to multi-link robotic problems, eg. to the triple inverted pendulum in series of case the previous case study, could lead to a significant improvement in performance in a broader spectrum of applications, particularly when using the RTI Scheme.

Hybrid Switching Scheme

As discussed previously, penalizing the energy-related terms lead to smoother responses. However, because of the aforementioned singularity problem, if only the energy terms are penalized instead of the angles directly, the optimisation would have no sensitivity to potential energy errors at the upper equilibrium, and would only be sensitive to angular velocity errors. Moreover, if a sufficiently small penalty was imposed on the angles, the optimisation could converge to upright positions were the angle errors were essentially ignored.

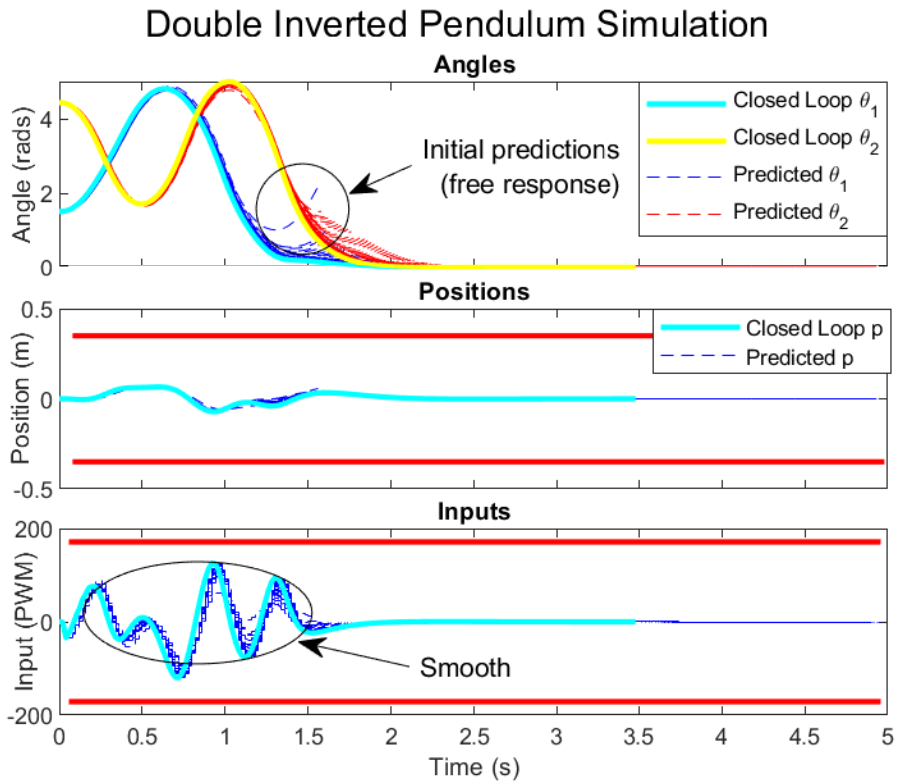
To avoid this problems whilst preserving the smoothness of the added energy terms during the swing up phase, a hybrid approach was used where the optimisation would switch between different weightings depending not only on the region in which the angles where, but also the time that they have been there.

The hybrid switching scheme is given by;

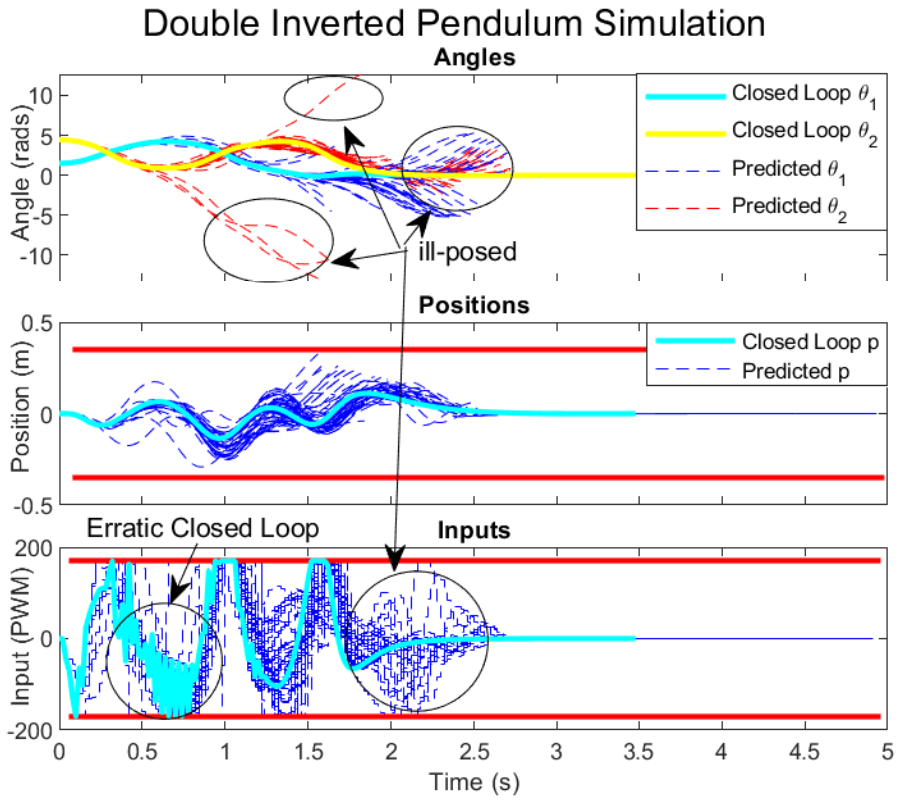
$$q_{\theta_i} = \begin{cases} 1, & t_{lin} < 2 \\ 10, & t_{lin} \geq 2 \end{cases} \tag{6.50}$$

where q_{θ_i} is the weight of the i_{th} pendulum angle error and t_{lin} is the time that has elapsed since $\cos\theta_i > 0.9$, ie. the time the system has been in the "linear" zone.

Regarding the stability of this proposed hybrid scheme, it should be noted that both penalization terms of q_{θ_i} were stable for our particular system, and the only reason for this change was to preserve the smoothness of the system during the swing up phase. As the change was implemented when the system was already in the terminal region, the cost of both selected weights had dissipated to zero within the available horizon, which in turn made the change between both weights stable. Essentially, the selection of this different weights changes the frequency response of the system to a more "rigid" or fast response for angle perturbations. Indeed, this approach could be used for fault-tolerant applications where the system momentarily has to undergo through a "softer/smoother" set of actions to bring the system back to its target before regaining a more "reactive" state.



(a) With



(b) Without

Figure 6.8: Example comparison of predicted (dashed-lines) and closed-loop (thick lines) responses with (6.8a) and without (6.8b) energy costs.

6.6.3 Experiment Setup

The test bench used for the experiments is depicted in figure 6.9. The cart is driven by a brushed 24V DC motor via a toothed-belt and a toothed-pulley of 0.05 (m) diameter. The DC motor is driven by a Cytron MD30C Motor Driver operated using sign-magnitude drive with a 8-bit resolution PWM at a frequency of 20 kHz via a Teensy 3.2 Micro Controller Unit (MCU). Three incremental encoders are used to measure both pendulum angles and the DC motor rotation. The resolution of both pendulum encoders and of the motor are 4000 and 2040 counts per revolution, respectively, which are processed by the MCU, leading to angle and position resolutions of 0.09° and 7.7×10^{-5} (m), respectively. The sampling time of the system is handled by the MCU and kept constant at $T_s = 20$ (ms). Every sampling time, encoders data is streamed via (UART) serial communication to a PC where the calculations related to proposed NMPC approach, OSI and EKF are performed in a Labview Human Machine Interface (HMI) Application which was used to visualise the information and interact with the system. After the control action is calculated via the RTI Scheme, it is sent back to the MCU which generates the motor signals. Due to network communication delays, the motor signal was always implemented exactly 5 ms after the encoders data was streamed to have a constant behaviour at least. Figure 6.10 shows a control diagram detailing the interaction between the different components.

Figure 6.9: Double Inverted Pendulum Test Bench Photograph

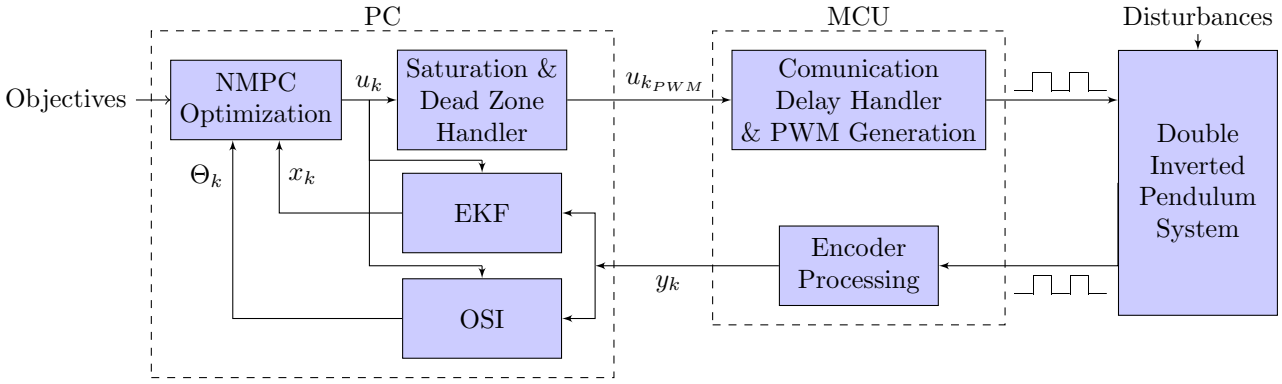


Figure 6.10: Double Inverted Pendulum Control Diagram

6.6.4 Online System Identification

To test the online system identification algorithm presented in section 5 of the paper, the system was excited using a random input $30 < ||u|| < 60$, which reversed every time the system crossed a maximum limit of the position $||x|| > 0.15$ (m) in the current direction. All the parameters were started from completely unknown values $\Theta_0 = \mathbb{O}$ with a forgetting factor of $\lambda = 0.995$ and initial covariance matrices as $P_0 = 1000 I_{3\times 3}$. The resulting performance of the overall OSI algorithm can be seen in figure 6.11. As it can be seen, the system presented very fast convergence rates, giving settling times for all the parameters of $\tau_s < 2$ (s), indicating that the models are indeed well defined. The resulting parameters after 1 minute of excitation are gathered in table 6.8, and the input-output data is available in [144]. Notice the theoretical relationship $c_i = b_i/g$ stated in section 6.6.1 is very close to the one observed in the resulting parameters. Finally, although the system was only tested for online system identification, it could work using the available adaptation mechanism provided proper rules are used to avoid the periods of poor excitation, as in the Adaptive Laguerre-based MPC publication [49] of chapter 4.

Motor	Coeffs.	Pend 1	Coeffs.	Pend 2	Coeffs.
f_m	-4.67	a_1	-0.129	a_2	-0.107
k_1	0.0174	b_1	38.4	b_2	49.6
k_2	0.0477	c_1	3.95	c_2	5.11

Table 6.8: Identified Parameters for the Double Inverted Pendulum (6.9)

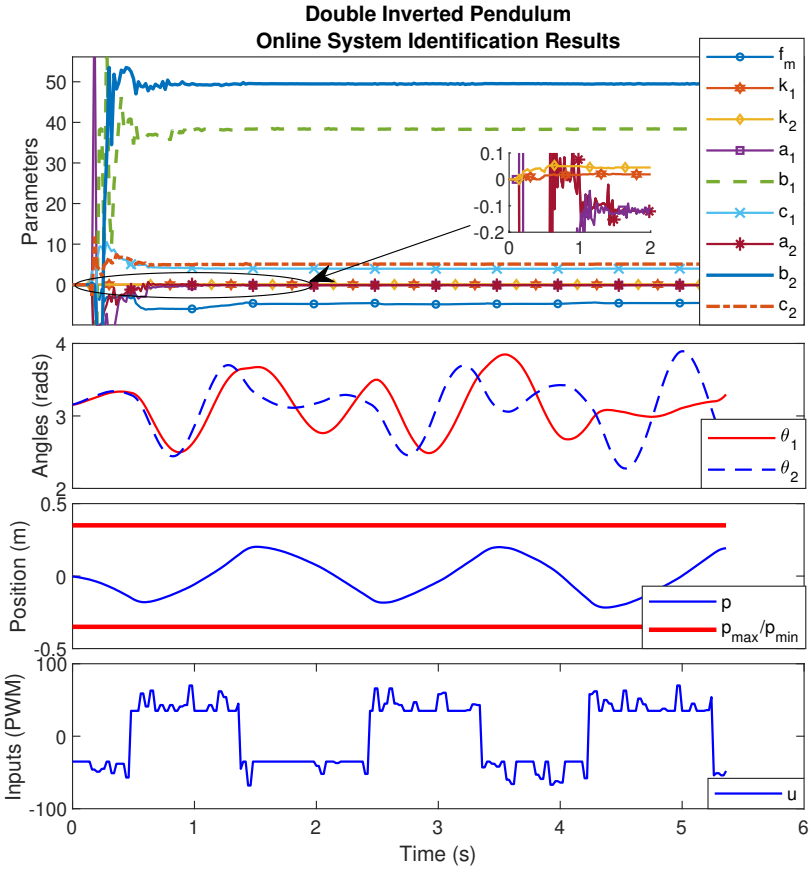


Figure 6.11: Online System Identification Example

6.6.5 Swing Up, Stabilisation and Disturbance Rejection

Regarding the optimisation setup, the cart had a maximum range for the position of $-0.35 < x < 0.35$ and the PWM input was constrained to $-200 < u_{PWM} < 200$ despite the actual maximum being 255 (8-bit) to avoid wearing of the DC motor which defined the constraints to be included in the optimisation. Furthermore, the DC motor presented a dead-zone nonlinearity of $u_{dz} \approx 30$ which was removed by implementing the conditional function (6.51), adjusting constraints to $-200 + u_{dz} < u < 200 - u_{dz}$ and using u in the relevant models to simulate and linearise the system.

$$u_{PWM} = \begin{cases} u + u_{dz}, & u > 0 \\ u - u_{dz}, & u < 0 \end{cases} \tag{6.51}$$

The prediction horizon for the NMPC was set at $N_p = 75$ ($T_p = 1.5$ (s)) leading to 600 outputs, 75 decision variables and 300 constraints to be optimised. The output and input weights were selected as $q_{k+i} = diag([1, 0.1, 0.1, 100, q_{\theta_1}, q_{\theta_2}, 10, 10])$ $\forall i = [1, N_p - 1]$ and $r_{k+i} = diag(0.003)$ $\forall i = [0, N_p - 1]$ which were observed to give a good balance between fast swing up performance and input chattering due to noise at the steady state. A terminal weight $q_{k+N_p} = 10q_{k+1}$ was selected for the last values of the prediction horizon when $t_{lin} > 2$ (s), emulating soft zero terminal constraints to improve the stability characteristics of the optimisation [65]. Moreover, a tailored C++ code available in [144] was developed using EIGEN library following suggestions of [65, 145], and was tested in a laptop running Ubuntu 18.04 with an Intel i7-5700 HQ @ 2.7 GHz giving computation times of $t_{unc} < 800$ μs for the unconstrained solution and $t_{con} < 2500$ μs for the constrained one when doing 10 iterations of an efficient version of Hildreth's QP found in [146]. Finally, the EKF weights were set at $Q_{EKF} = diag([0.1, 0.1, 0.1, 0.0001, 0.0001, 0.0001, 1])$ and $R_{EKF} = diag([0.0001, 0.0001, 0.0001, 1])$ based on the variance of the errors observed in an offline analysis of the system identification process.

Remark 6.7. *It should be noted that the $O(N)$ and $O(N^2)$ algorithm extensions (6.1 and 6.2) presented in this chapter were not implemented in this experiment as they hadn't been developed yet, and due to COVID-19, access to the equipment has not been possible since March 2020. This ultimately means faster solutions could be obtained.*

The resulting performance of the overall scheme can be seen in figure 6.12 starting from the rest position at the lower equilibrium and introducing large disturbances at $t \approx 9$ (s) and $t \approx 17$ (s). As it can be seen, the system clearly exhibits much faster performance than [6] giving settling times of $\tau_s \approx 1.4$ (s) for the swing up maneuver and of $\tau \approx 2.5$ (s) after large disturbances. Moreover, the system presented smooth input shapes during the swing up phases as a result of the added energy costs and the hybrid switching scheme. Furthermore, notice the position constraint is clearly satisfied at $t \approx 17$ (s) after the disturbance was given, demonstrating good handling of the rapid active-set changes by the QP. Finally, in some cases the position presented small steady state error and the well known limit cycle, however, this can be removed using standard methods such as integral control or disturbance estimation methods which were outside the scope of this work, and therefore were omitted. For the interest of the reader, an overall video is provided in (https://youtu.be/7E-SXi3YKQo) where the results can be seen, and the input-output data is available in [144].

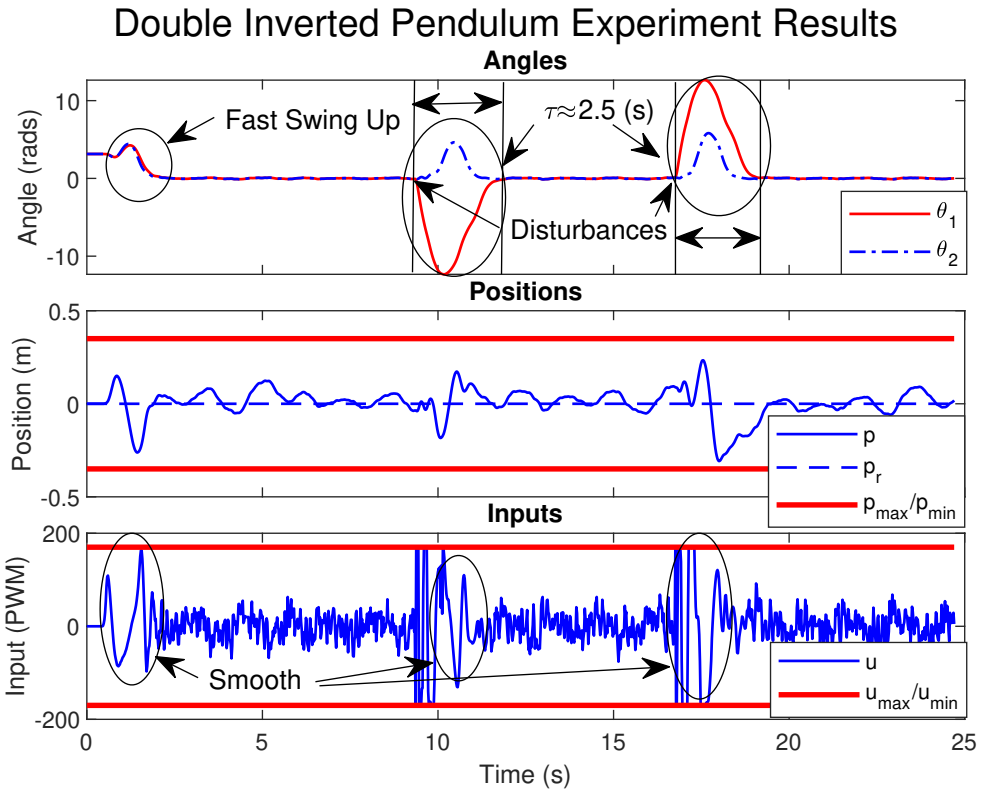


Figure 6.12: Example performance with initial condition steady at lower equilibrium and large disturbances at $t \approx 9$ (s) and $t \approx 17$ (s).

6.7 Summary

In this chapter we presented the development of a novel pre-stabilisation approach for Nonlinear Model Predictive Control, sometimes referred to as Closed Loop Dual Mode NMPC. The proposed approach uses the Time-Varying Discrete Algebraic Riccati Equation (DARE) to obtain a set of optimal feedback gains that pre-stabilise the predicted trajectory along the current optimal guess. A key difference of the proposed approach is that the pre-stabilisation target is the previous constrained optimal trajectory itself rather than the unconstrained solution as done by many standard approaches. Moreover, because the approach uses a set of time-varying feedback gains, it allows the user to obtain pre-stabilised predictions along virtually any possible (or desired) trajectory that emerges from the optimisation, rather than using a single feedback gain as in standard approaches, eg. to stablise the system to the origin, which may not even be adequate in specific scenarios such as in Inverted Pendulum systems.

One of the main contributions of the approach is theorem 6.2 which proves the equivalency with the standard NMPC approach, hence guaranteeing the exact same to nominal stability, recursive feasibility and convergence properties. In addition to this, two important contributions presented in this chapter are the extensions of the $O(N)$ and $O(N^2)$ algorithms, presented in section 6.2 which overall allow efficient computation of the Hessian E and linear term f. Moreover, the chapter includes the "core" as well as feedback and preparation algorithms required to implement the approach using the RTI Scheme.

Given that the approach requires additional computations, it inevitably increases the computation time of the solutions. To assess this, a generic computations comparison with the standard method was performed based on the Eigen 3 library where the approach was observed to give up to $50 - 89\%$ increased computation times when computing some of the main preparation and feedback operations, depending predominantly on the prediction horizon and number of inputs of the optimisation as seen in table 6.1. However, upon reviewing the "expected" computational increase with the one reported in table 6.1, it was clear that some parts of the underlying matrix-matrix or matrix-vector operations were not being done optimally which motivated the verification process in a different computing platform; the Beaglebone Blue, where increases as low as 13% were obtained in one of the operations. This comes to say that the implementation of the developed algorithms in other platforms could lead to better computational performance, particularly if tailored designs eg. based on Field Programmable Gate Arrays (FPGAs) were to be used. Irrespective of this, it is important to understand that these were not the "total" computation times required as the solution of the resulting quadratic program itself was not included, and the approach resulted in much better performance when the approach was considered in this scenario as seen from table 6.6 where increases as low as 3.2% were obtained. Nonetheless, the reader should be reminded that this chapter was developed as a pre-requisite methodology towards the combined approach presented in chapter 8 where the computational benefits of the shifting strategy of chapter 5 allow significantly better computational performance.

The chapter included 4 case studies, demonstrating the various properties of the approach, namely: the nonlinear ball-plate system which presented severe unstable dynamics that in certain scenarios were unable to be handled by the standard approach; the standard Inverted Pendulum system where important properties were discussed related to potential issues of the standard approach that would be solved using the proposed approach along with a computational performance comparison based on an auto-generation toolkit developed as part of the contributions; the Triple Inverted Pendulum which formed the ideal case in which the proposed approach would be required as it was unable to be solved with the standard approach altogether; and the Experimental Validation of a Parallel Double Inverted Pendulum, which formed the main contribution of the IET paper [47].

Chapter 7

Infinite Horizon Costing for the Ideal Moving Window Blocking Approach

In chapter 5 we presented the Shifting Strategy approach which focuses on reducing the amount of decision variables and constraints of the system in a systematic way to preserve nominal stability and recursive feasibility proofs. However, the nominal stability proofs that were provided in the aforementioned chapter were based on both: infinite horizons, which for all practical purposes are intractable for real-time implementation; and zero-terminal constraints, which are known to be too stringent without necessarily giving good performance as it was shown in figure 3.3a of chapter 3. Therefore, it was decided to investigate the possibility of obtaining an infinite horizon costing solution for the Ideal Moving Window Blocking (MWB) approach that would guarantee nominal stability using a more relaxed condition. Indeed, infinite horizon costing with invariant sets is known to be one of the strongest approaches for guaranteeing nominal stability of the optimisation, as discussed in chapter 3, section 3.4.

On the other hand, the previous chapter 6 introduced the proposed Dual Mode Closed Loop NMPC framework for efficiently handling unstable systems. The interest in this type of solution emerged after the Ball Plate System from [18] was observed to present numerical challenges when applying the Ideal MWB approach presented in chapter 5, even when using relatively small horizons of $N_p = 20$. In practice one may opt to embed the unconstrained solution into the predictions, and embed the block structure into the "additional" decision variables $u_k = -Kx_k + \boxed{c_k}$ as discussed in the previous chapter, which in general, would allow the user to use standard infinite horizon costing techniques. Indeed, this is how the authors from [18] applied the original MWB approach (see equation 10 of the paper). However, in this scenario, the optimisation would require one to maintain constraints on all the inner inputs of all the blocks in order to guarantee recursive feasibility due to the embedded unconstrained path on the inputs, thus removing one of the main computational benefits. Instead, obtaining the infinite horizon solution to the problem whilst embedding the blocked solution would result in a set "blocked feedback gains" that would allow the required input-related constraints to be reduced as in the original blocked problem. Thus, this represents another motivation of this chapter which will be exploited further in the following chapter 8: The Combined Approach, where closed loop dual mode blocked feedback gains will be used to obtain blocked pre-stabilised prediction models for the final approach proposed in this thesis.

Given the relative simplicity of the solution that will be presented, this chapter will be kept brief. The chapter is organised as follows: Section 7.1 describes the conceptual task related to both obtaining and appropriately embedding the infinite horizon costing in the Ideal MWB approach. Section 7.2 presents the derivation of one possible general solution to the infinite horizon problem of the Ideal MWB based on the Dynamic Programming approach along with algorithm 7.1 for its implementation. Finally, section 7.3 presents the case study of applying the proposed solution to a linear double integrator system where relevant properties are discussed. The chapter ends with a brief summary.

Remark 7.1. *Range of N_{b_0} and N_{b_i}*

This chapter will use the range $N_{b_0} = N_{b_i} = [1 \rightarrow N_b]$ discussed in remark 5.6.

7.1 The Time-Varying Infinite Horizon Costing and Solution

One of the very first concepts to understand when looking for the infinite horizon costing is that given the time-varying nature of the Ideal MWB parameterisation (5.11), the resulting infinite horizon solution, and therefore, costing will also be time-varying. In particular, it will vary according to the "virtual block position indicator" (N_{b_0}), ie. the index that keeps track of the absolute time frame. In order to explain this, let us consider the infinite horizon solution that would result when using a block-size of $N_b = 10$ with an Ideal Prediction Horizon of $N_p = 41$ at the absolute time-step $N_{b_0} = 7$. This scenario is pictured in figure 7.1 for a random system where various important parts are labelled.

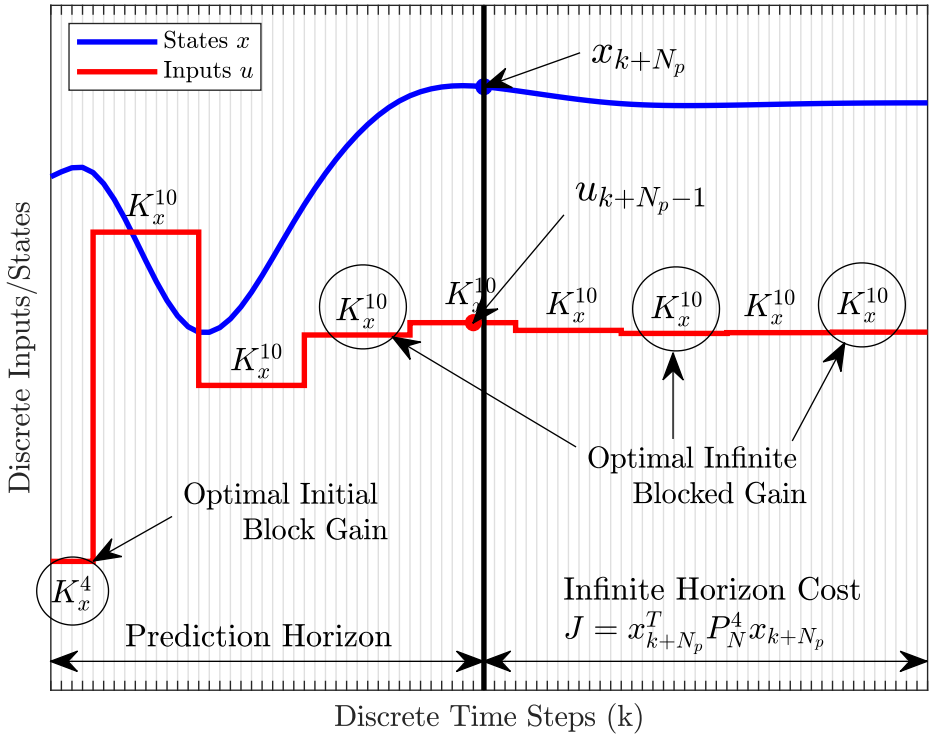


Figure 7.1: Infinite Horizon Optimal Solution of Ideal MWB for $N_b = 10$ and $N_{b_0} = 7$

There are 3 mains parts of interest in this figure, namely: the initial blocked gain, the infinite horizon blocked gain, and the infinite horizon cost, on which we will elaborate to provide insight of their general properties and utility.

Consider at first that there is no distinction between the Prediction Horizon and the Infinite Horizon parts, assuming we are interested in finding the entire solution using the Dynamic Programming approach introduced in chapter 3, section 3.4. Note that at the specified absolute time-step ($N_{b_0} = 7$), the first block would have a size of $N_b - N_{b_0} + 1 = 4$ steps, ie. embedding equal values on the first 4 input which can be seen in the lower-left corner. At this segment, the infinite horizon solution would apply an "optimal initial block gain" K_x^4, representing a gain specifically designed for a block of size 4. Given the size of the first block would vary as the time-step moves forward and resets $N_{b_0} = [1 \rightarrow N_b, 1, \rightarrow N_b, \infty]$, the initial gain would vary in terms of it being optimal for a different block size, eg. K_x^3 begin the optimal initial blocked gain for the following time-step ($N_{b_0} = 8$). After the initial block segment, the rest of the solution would embed static feedback gains for blocks of size 10, referred to as the "optimal infinite blocked gain" in the figure, represented by the multiple K_x^{10} gains embedded across the rest of the horizon. Both of this feedback gains could be used to obtain blocked pre-stabilised prediction models, eg. in the context of linear MPC. Indeed, the following chapter 8 will be focused on finding blocked feedback gains such as this ones which will allow the "blocked" pre-stabilisation of nonlinear models used by NMPC.

On the other hand, consider the given prediction horizon of $N_p = 41$, depicted by the double-arrow in the lower-left side of the figure. The interest is to find an infinite horizon cost or "terminal weight", potentially having the conceptual standard form of $x_{k+N_p}^T P_N^i x_{k+N_p}$ as depicted in the lower-right side of the figure. What is key here to notice, is that the resulting cost must represent the rest of the solution whilst keeping the last input u_{k+N_p-1} constant for the remaining steps before applying the infinite horizon gain again. In this particular example, the infinite horizon cost must consider the last input will remain constant for 3 additional steps BEYOND the horizon, visible from discrete spaces between the black vertical line and the immediate following block edge, giving a total of 4 steps when measured from the final decision variable (u_{k+N_p-1}). Thus, the conceptual notation $x_{k+N_p}^T P_N^4 x_{k+N_p}$ provided in the figure is used to indicate the "terminal weight for a remaining block of size 4". Note that, like in the case of the initial block gain, the size of the remaining steps beyond the horizon on which the last input must considered constant will also vary. As a result, it can be shown that both the optimal initial blocked gain (K_{x_0}) and the terminal weight (P_N) to be used at time-step N_{b_0} would vary according to equation (7.1). An example of this is equation applied to the problem of figure 7.1 at all time-steps $N_{b_0} = [1, N_b]$ is given in table 7.1.

$$K_{x_0} = K_x^{N_b - N_{b_0} + 1} \tag{7.1a}$$

$$P_N = P_N^{N_b - N_{b_0} + 1} \tag{7.1b}$$

Initial Block Size N_{b_0}	1	2	3	4	5	6	7	8	9	10
Terminal Weight P_N	P_N^{10}	P_N^9	P_N^8	P_N^7	P_N^6	P_N^5	P_N^4	P_N^3	P_N^2	P_N^1
Initial Block Feedback Gain K_{x0}	K_x^{10}	K_x^9	K_x^8	K_x^7	K_x^6	K_x^5	K_x^4	K_x^3	K_x^2	K_x^1

Table 7.1: Table of admissible terminal weights and initial block feedback gains of figure 7.1.

7.2 Methodology and Algorithm

So far we haven't really discussed how to obtain any of the aforementioned parts: the initial blocked gain, the infinite blocked gain, and the terminal weight, and we have focused mainly on the conceptual time-varying requirements of the resulting solutions, irrespective of how they are obtained.

In this section, we will present a methodology for obtaining the required blocked feedback gains and terminal weights via the Dynamic Programming approach, based on the technique of state-augmentation. Although other methods such as lifted systems [124], as well as modified versions of the standard Dynamic Programming approach including crossed input-state penalisation terms can be used, we found the proposed method to be the simplest to implement in practice. Moreover, given there is no particular requirement for real-time performance, as this is typically considered an "offline" procedure, the additional computations related to the state-augmentation are not relevant, and the method is simply provided as proof that obtaining infinite horizon solutions for guaranteeing nominal stability of the proposed Ideal MWB approach can be achieved.

Recalling the Dynamic Programming procedure presented in chapter 3, section 3.4, the method is based on Bellman's "principle of optimality" where the optimal costs of two subsequent stages (J_k and J_{k+1}) can be reduced to a single stage, by solving the optimisation of the second stage (J_{k+1}) and propagating its cost backwards to be represented at the first stage J_k via the so-called "Riccati" recursion. In this case, we require to do something similar, with the main difference of having the additional equalities of the Ideal MWB approach, which could potentially cause some difficulties or require some additional terms when considered in the standard Riccati recursion ($P_k = Q + A^T P_{k+1} A - A^T P_{k+1} B K_k$).

A simple alternative is to consider the system using a state-space augmentation of the form:

$$z_k = \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix} \tag{7.2}$$

where the predictions of the system are given by two models, namely:

1. A "Closed Loop" model that will be used to make the blocked decisions of the optimisation, given by:

$$z_{k+1} = A_{CL} z_k + B_{CL} u_k \qquad \text{Closed Loop} \tag{7.3a}$$

$$A_{CL} = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix} \quad B_{CL} = \begin{bmatrix} B \\ I \end{bmatrix} \tag{7.3b}$$

   which captures the basic dynamics of standard state-space models $x_{k+1} = Ax_k + Bu_k$.

2. An "Open Loop" model, which will be used to "maintain" the blocked decisions of the optimisation through an "integrator-type" of dynamics, given by:

$$z_{k+1} = A_{OL} z_k \qquad \text{Open Loop} \tag{7.4a}$$

$$A_{OL} = \begin{bmatrix} A & B \\ 0 & I \end{bmatrix} \tag{7.4b}$$

Note that this last model does not have decision variables as it will only be used for propagation the blocked inputs in the Riccati recursion.

Consider now the general method for handling the Shifting Strategy described in section 5.2.1, where the inner block position index (N_{b_i}) allows the user to keep track of the block position at any predicted time-step.

Let us assume that the infinite horizon solution is started at a block position of $N_{b_i} = 1$ on which a "new" blocked decision must be made. The optimisation of this stage, say J_{k+1}, could be easily solved by using model (7.3) and optimising:

$$J_{k+1} = \hat{z}_{k+1}^T P_N^{N_b} \hat{z}_{k+1} \tag{7.5a}$$

$$s.t. \quad \hat{z}_{k+1} = A_{CL}\hat{z}_k + B_{CL}\hat{u}_k \tag{7.5b}$$

Remark 7.2. *Embedded Input Penalisation*

It is considered in (7.5) that the weight related to the input penalisation of u_k is inside $P_N^{N_b}$. This can be done, eg. by considering the augmented state penalisation matrix as $Q_{aug} = blkdiag([q_{k+1}, r_k])$.

The well known unconstrained solution to (7.5) is given by:

$$u_k = -K_x^{N_b}\hat{z}_k \tag{7.6a}$$

$$K_x^{N_b} = (B_{CL}^T P_N^{N_b} B_{CL})^{-1} B_{CL}^T P_N^{N_b} A_{CL} \tag{7.6b}$$

We can then follow the same procedure of chapter 3, section 3.4 where by substituting the solution in the original cost (7.5) and grouping the terms w.r.t. $\hat{z}_k$ allows us to express the cost of J_{k+1} in terms of the previous cost $J_k = \hat{z}_k^T P_N^1 \hat{z}_k$ by:

$$K_x^{N_b} = (B_{CL}^T P_N^{N_b} B_{CL})^{-1} B_{CL}^T P_N^{N_b} A_{CL} \tag{7.7a}$$

$$P_N^1 = Q_{aug} + A_{CL}^T P_N^{N_b} A_{CL} - A_{CL}^T P_N^{N_b} B_{CL} K_x^{N_b} \tag{7.7b}$$

Note that the calculation is stored in the infinite horizon weight P_N^1 given it would be the terminal weight related to having an input blocked for 1 step before applying the infinite blocked gain of $K_x^{N_b}$.

Similarly, when the inner block index is not at the start of a "new" block, the original task would be to maintain the previous input, or more specifically, propagate the cost related to blocking the input backwards. This can be done easily by propagating the related weights backwards through the "integrator-type" dynamics from (7.4), resulting in a recursion for when the algorithm is NOT at a new block ($N_{b_i} > 1$), given by:

$$P_N^{N_b - N_{b_i} + 2} = Q_{aug} + A_{OL}^T P_N^{N_b - N_{b_i} + 1} A_{OL} \tag{7.8a}$$

The iterative use of both equations (7.7 and 7.8) allow the calculation of the terminal weights for all N_{b_0}, and the "infinite blocked gain" ($K_x^{N_b}$). However, the initial blocked gain has still not yet been addressed. Nevertheless, it can be obtained simply by calculating it as if it would be applied for feedback but without applying it. This can be seen in line 8 of the final algorithm 7.1 where the feedback gain is always calculated, but it is only "applied" if at a new block ($N_{b_i} = 1$).

The final algorithm is given in algorithm 7.1 which calculates the infinite horizon solution for the Ideal MWB approach. This algorithm includes terminal condition checks (δ^i) which compare each of the terminal weights P_N^i with the ones obtained in the previous iteration. The user must then provide a maximum number of iterations (N_{max}), along with the desired tolerance (ϵ_{tol}). Moreover, note that the inner block index is required to be iterated backwards as the blocks position is indeed moving backwards.

We will provide an example of its application along with several properties of interest in the case study of section 7.3.

Algorithm 7.1: General Infinite Horizon Solution for the Ideal MWB Approach

Data: $Q_{aug}, A_{OL}, A_{CL}, B_{CL}, N_b, N_{max}, \epsilon_{tol}$

begin

- $P_{k+1} = Q_{aug}$; // Initialise DARE Weight Matrix
- $P_N^i = Q_{aug}\ \forall\ i = [1, N_b]$; // Initialise all Terminal Weights
- $\delta^i = 100000\ \forall\ i = [1, N_b]$; // Initialise Differences for Terminal Condition
- $N_{b_i} = N_b$; // Initialise Inner Block Position Index
- $n = 1$; // Initialise Iteration counter
- /* Find Optimal Blocked Terminal Weights and Feedback Gains */
- **while** $\max(\delta^i) > \epsilon_{tol}\ AND\ n < N_{max}$ **do**
  - $K_x^{N_b - N_{b_i}+1} = (B_{CL}^T P_{k+1} B_{CL})^{-1} B_{CL}^T P_{k+1} A_{CL}$; // Calculate optimal feedback gain
  - **if** $N_{b_i} = 1$ **then**
    - /* Propagate weight and apply feedback only at new block ($N_{b_i} = 1$) */
    - $P_k = Q_{aug} + A_{CL}^T P_{k+1} A_{CL} - A_{CL}^T P_{k+1} B_{CL} K_x^{N_b - N_{b_i}+1}$
    - $i = 1$; // Terminal weight index
  - **else**
    - /* Propagate Weight only if not at new block */
    - $P_k = Q_{aug} + A_{OL}^T P_{k+1} A_{OL}$
    - $i = N_b - N_{b_i} + 2$; // Terminal weight index
  - $\delta^i = ||P_k - P_N^i||_2$; // Calculate norm difference of i_{th} weight
  - $P_N^i = P_k$; // Store new terminal weight for i_{th} index
  - $N_{b_i} = N_{b_i} - 1$; // Move inner block position index backwards
  - **if** $N_{b_i} < 1$ **then**
    - $N_{b_i} = N_b$; // Reset weight
    - $n = n + 1$; // Increase iteration counter

Result: $P_N^{N_{b_i}}, K_x^{N_{b_i}}\ \forall\ N_{b_i} = [1, N_b]$

7.3 Case Study: Double Integrator

In order to validate the developed infinite horizon solution for the Ideal MWB, we selected the linear double integrator system from the zero-terminal example 3.2, given by:

$$x_{k+1} = \underbrace{\begin{bmatrix} 2 & -1 \\ 1 & 0 \end{bmatrix}}_{A} x_k + \underbrace{\begin{bmatrix} 0.1 \\ 0 \end{bmatrix}}_{B} u_k \tag{7.9}$$

subject to the penalisation weights $q = diag([1, 1]), r = 1$ with a sampling time of $T_s = 0.02$ (s).

7.3.1 Exact Solution

One of the simplest ways to verify that the correct solution has been found is to compare the cost predicted by the terminal weight, and the cost obtained when simulating the system. However, given that the predicted cost is calculated considering that an actual blocked plan will be executed, the simulation must apply this plan explicitly by only applying the "infinite blocked gain" $K_x^{N_b}$ every time there is a new block (ie. $N_{b_0} = 1$). Otherwise, the predicted cost will differ.

To give an example of this, consider a block-size of $N_b = 5$. The resulting terminal costs and feedback gains that result from using algorithm 7.1 would be:

$$P_N^1 = \begin{bmatrix} 90.9 & -66.8 & 0 \\ -66.8 & 53 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad P_N^2 = \begin{bmatrix} 150.3 & -115 & 11.5 \\ -115 & 91.9 & -9.1 \\ 11.5 & -9.1 & 2.9 \end{bmatrix} \quad P_N^3 = \begin{bmatrix} 234.2 & -185.6 & 32.5 \\ -185.6 & 151.3 & -55.9 \\ 32.5 & -26.5 & 7.7 \end{bmatrix}$$

$$P_N^4 = \begin{bmatrix} 346.6 & -282.8 & 66.7 \\ -282.8 & 235.2 & -55.9 \\ 66.7 & -55.9 & 17.5 \end{bmatrix} \quad P_N^5 = \begin{bmatrix} 491.6 & -410.5 & 118.5 \\ -410.5 & 347.6 & -101.3 \\ 118.5 & -101.3 & 35.3 \end{bmatrix} \tag{7.10a}$$

$$K_x^1 = \begin{bmatrix} 6.02 & -4.76 & 0 \end{bmatrix} \quad K_x^2 = \begin{bmatrix} 4.84 & -3.95 & 0 \end{bmatrix} \quad K_x^3 = \begin{bmatrix} 4.03 & -3.38 & 0 \end{bmatrix}$$

$$K_x^4 = \begin{bmatrix} 3.45 & -2.95 & 0 \end{bmatrix} \quad K_x^5 = \begin{bmatrix} 3.02 & -2.62 & 0 \end{bmatrix} \tag{7.10b}$$

There are a few thing to note from these gains and costs. First, note that the only cost which does not involve crossed-terms multiplications between x_k and u_{k-1} is P_N^1. This is because this cost represents the scenario where input u_{k-1} would remain constant for only a single step, and feedback would be immediately applied after that. As a result, the solution will not depend on where the value of u_{k-1} could be taking the state x_k before applying feedback. In contrast, all the other costs $P_N^i \ \forall i = [2, N_b]$ represent the alternative scenario where u_{k-1} will be maintained constant for 2 to N_b steps before feedback would be applied. Thus, the terminal weights must capture the costs related to the evolution of x_k with u_{k-1} before feedback is applied. The second thing to note here is that the last value of all initial and infinite blocked feedback gains, is always zero. This can be verified by observing that the expression for K_x involves a multiplication with A_{CL} which causes these zeros. Essentially, this means that non of the optimal blocked control actions depend on the last value of the input u_{k-1} which is to be expected given a correction to the current path is available, as opposed to the terminal costs $P_N^i = [2, N_b]$ which require to consider the last input value u_{k-1} will be kept constant.

Now, consider an initial augmented state of $z_0 = [10, 5, 4]^T$. Note that this initial condition will be used at several points in this case study. Figure 7.2 shows a $T = 1$ second simulation where the obtained responses for 2 cases can be seen: 1. when the "infinite blocked gain" K_x^5 is immediately applied ($N_{b_0} = 1$), ie. applied in time steps $k, k+5, k+10, \cdots, \infty$, depicted by the blue line; and 2. when the gain is applied after 5 steps ($N_{b_0} = 2$), ie. at time steps $k+4, k+9, \cdots, \infty$, depicted by the red dot-dashed line where the initial value of the input can be seen to be maintained at the initial value of ($u_{k-1} = 4$) before applying feedback at $T = 0.08$, ie. $k+4$ when considering the sampling time ($T_s = 0.02$ (s)).

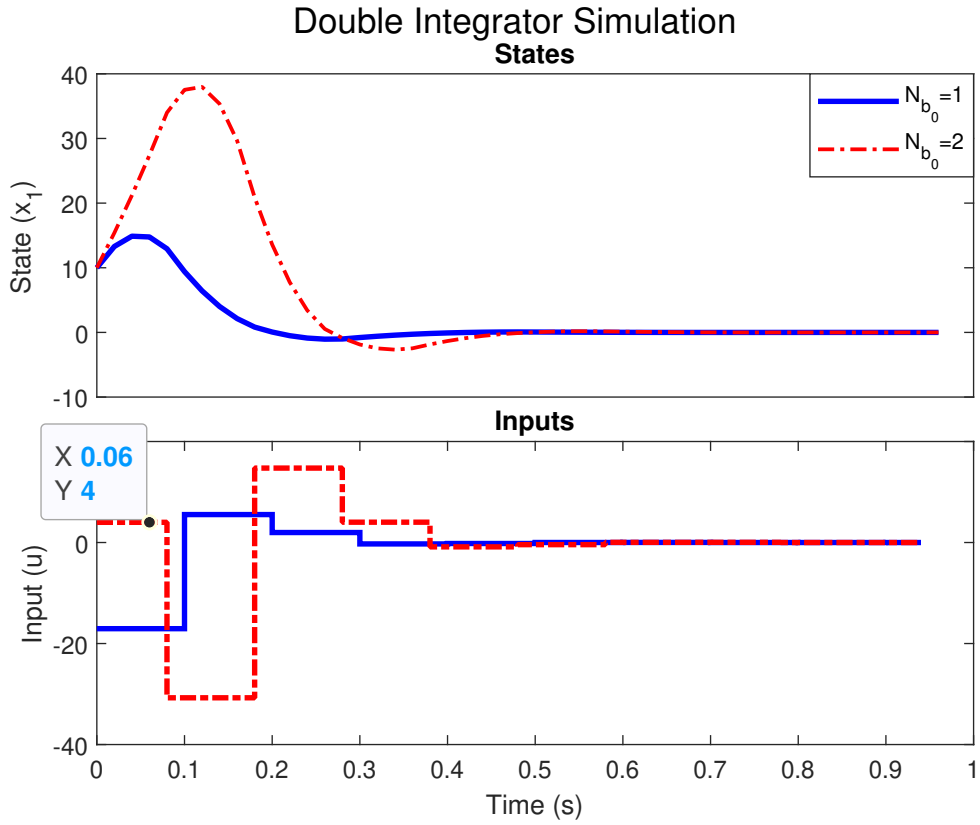


Figure 7.2: Example Simulation of Double Integrator System with Infinite Horizon Solutions at two different initial time-steps $N_{b_0} = [1, 2]$

The reader can verify that the resulting costs of both simulations using the aforementioned conditions are $J_1 = 3,748$ and $J_2 = 22,795$, both of which can be calculated exactly by $J_1 = z_0^T P_N^1 z_0$ and $J_2 = z_0^T P_N^5 z_0$ which satisfies the basic requirement of representing the infinite horizon cost EXACTLY. This was also true for starting the simulation at any other time-step $N_{b_0} = [1 \rightarrow N_b]$. Moreover, this example is used to illustrate the significant difference of the predicted/resulting cost obtained when the input must be kept constant for certain amount of steps before feedback is applied. Thus, it is very important to properly account for this in the terminal weights for them to be used as infinite horizon costs of the proposed Ideal MWB approach.

7.3.2 Infinite Horizon Costing Comparison

Having demonstrated the validity of the proposed solution, we can now look to implement the terminal weights in the context of the Ideal MWB MPC implementation. To do this, let us consider the smallest Ideal Prediction Horizon for the aforementioned block-size of $N_p = 6$. Clearly, the solution to the problem used in figure 7.2 does not reach a near "zero" state (where costs would be zero) within $N_p = 6$ steps, ie. $T_p = 0.12\ (s)$ of the initial time step. Thus, we can expect that applying the terminal weight in this case could lead to relevant improvements.

To evaluate this, consider again the application of feedback only at the start of new blocks, ie. when $N_{b_0} = 1$. This means that the Ideal MWB approach in this scenario would only use a single parameterisation matrix given by:

$$\mathbb{N} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{7.11}$$

In the case where the predictions were formulated using the augmented models as in chapter 8 we can directly use the augmented terminal weights P_N^i, however, in order to implement them in the original non-augmented framework, we must separate the terminal weights in terms of states and inputs by:

$$\begin{aligned} z_{k+N_p}^T P_N^i z_{k+N_p} &= \begin{bmatrix} x_{k+N_p}^T & u_{k+N_p-1}^T \end{bmatrix} \begin{bmatrix} P_{N_1}^i & P_{N_2}^i \\ {P_{N_2}^i}^T & P_{N_3}^i \end{bmatrix} \begin{bmatrix} x_{k+N_p} \\ u_{k+N_p-1} \end{bmatrix} \\ &= \begin{bmatrix} x_{k+N_p}^T & u_{k+N_p-1}^T \end{bmatrix} \begin{bmatrix} P_{N_1}^i x_{k+N_p} + P_{N_2}^i u_{k+N_p-1} \\ {P_{N_2}^i}^T x_{k+N_p} + P_{N_3}^i u_{k+N_p-1} \end{bmatrix} \\ &= x_{k+N_p}^T P_{N_1}^i x_{k+N_p} + x_{k+N_p}^T P_{N_2}^i u_{k+N_p-1} + u_{k+N_p-1}^T {P_{N_2}^i}^T x_{k+N_p} + u_{k+N_p-1}^T P_{N_3}^i u_{k+N_p-1} \end{aligned} \tag{7.12}$$

Clearly $P_{N_1}^i$ and $P_{N_3}^i$ can be considered in the standard Q and R matrices as the terminal weights for $q_{k+N_p} = P_{N_1}$ and $r_{k+N_p-1} = P_{N_3}$. However, a slight more involved modification is required to include the cross-terms terminal weights P_{N_2}. The reader can verify that including these extra weights can be obtained simply by modifying the Hessian and linear term of the input-parameterised MPC to:

$$J = \frac{1}{2}\hat{\mathbb{U}}^T E_{\mathbb{N}} \hat{\mathbb{U}} + \hat{\mathbb{U}}^T f_{\mathbb{N}} \tag{7.13}$$

$$E_{\mathbb{N}} = \mathbb{N}^T (H^T Q H + R)\mathbb{N} + \mathbb{N}^T H_{N_p}^T P_{N_2}^i \mathbb{N}_{N_p} + \mathbb{N}_{N_p}^T {P_{N_2}^i}^T H_{N_p} \mathbb{N} \tag{7.14}$$

$$f_{\mathbb{N}} = N^T (H^T Q G x_k) + \mathbb{N}_{N_p}^T {P_{N_2}^i}^T G_{N_p} \tag{7.15}$$

where H_{N_p}, G_{N_p} and $\mathbb{N}_{N_p}$ are the $N_{p_{th}}$ rows of H, G and $\mathbb{N}$.

Because we will only apply feedback at the start of new blocks, ie. when $N_{b_0} = 1$, the terminal weight must consider the last input u_{k+N_p-1} will be maintained constant for 4 additional steps, giving a total of 5 steps. Thus, the terminal weight to be used is P_N^5. To match the responses with the previous example, the Ideal MWB MPC was started at time-step $N_{b_0} = 1$.

Figure 7.3 shows an example of the solution with and without the terminal weight when solved using the Ideal MWB MPC framework where it can be seen that the solution without the terminal weight presents rather poor performance with oscillatory behaviour. In contrast, the solution with the terminal weight maintains the exact same performance of figure 7.2, inevitably resulting in the same cost $J_{with} = 3,748$. Thus, this gives an example of the advantage of including the terminal cost in the Ideal MWB framework which ultimately provides the desired nominal stability guarantee.

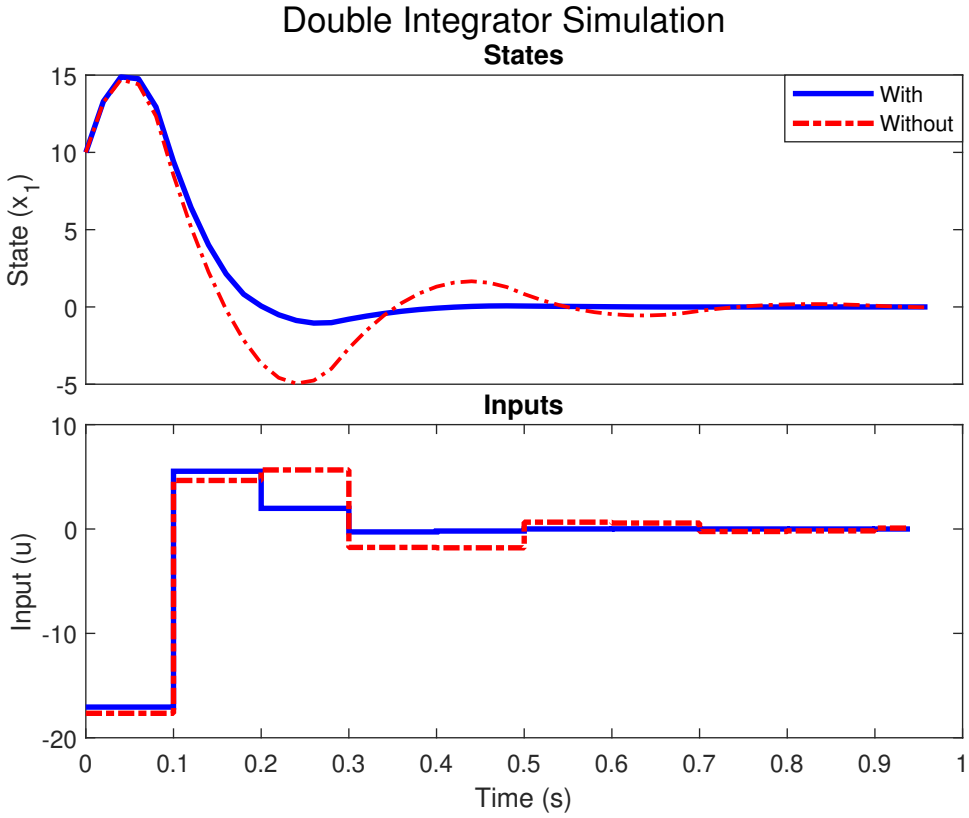


Figure 7.3: Example Simulation of Double Integrator System with and without Infinite Horizon Costing

7.3.3 Cost Limits

So far we have focused on following the "original" plan by applying feedback only at the start of a block $N_{b_0} = 1$. However, an interesting feature of the Ideal MWB approach is that unless the optimisation is going through a critical feasibility stage where it is enforced to continue with the current plan to satisfy constraints, the solution will typically be able to improve as the horizon moves forward which essentially means that the terminal weights of the Ideal MWB are only "Cost Limits" which in reality can be smaller when solution applies feedback at every step instead. We have already shown an example of this in figure 5.5 where the input can be seen to be making small corrections to the blocks.

As an example, figure 7.4 shows the response obtained for two cases: 1. when applying the feedback gains of (7.10b) sequentially, resetting when the new block emerges and repeating infinitely $K_x^5, K_x^4, \cdots, K_x^1, K_x^5, K_x^4, \cdots, K_x^1, \infty$, depicted by the blue line; and when feedback is only applied at new blocks as before, depicted by the red dot-dashed line. The resulting costs of both of these trajectories are: $J_{with} = 3,685$ and $J_{without} = 3,748$, ie. around 1.68% better when compared in terms of suboptimality $\Delta J = (J_{with}/J_{without} - 1) \times 100$. This is to be expected given the optimisation would only modify the input if it can improve the current blocked plan. The key message from this is that the costs calculated with the terminal weights are only limits and the Ideal MWB approach will typically present better costs that the ones it initially predicts. Lastly, note that this could equally be obtained if the Ideal MWB approach were to be used with the time-varying parameterisation (5.11) along with the terminal weights $P_N^5, P_N^4, \cdots, P_N^1, P_N^5, P_N^4, \cdots, P_N^1, \infty$ applied as in equation (7.13).

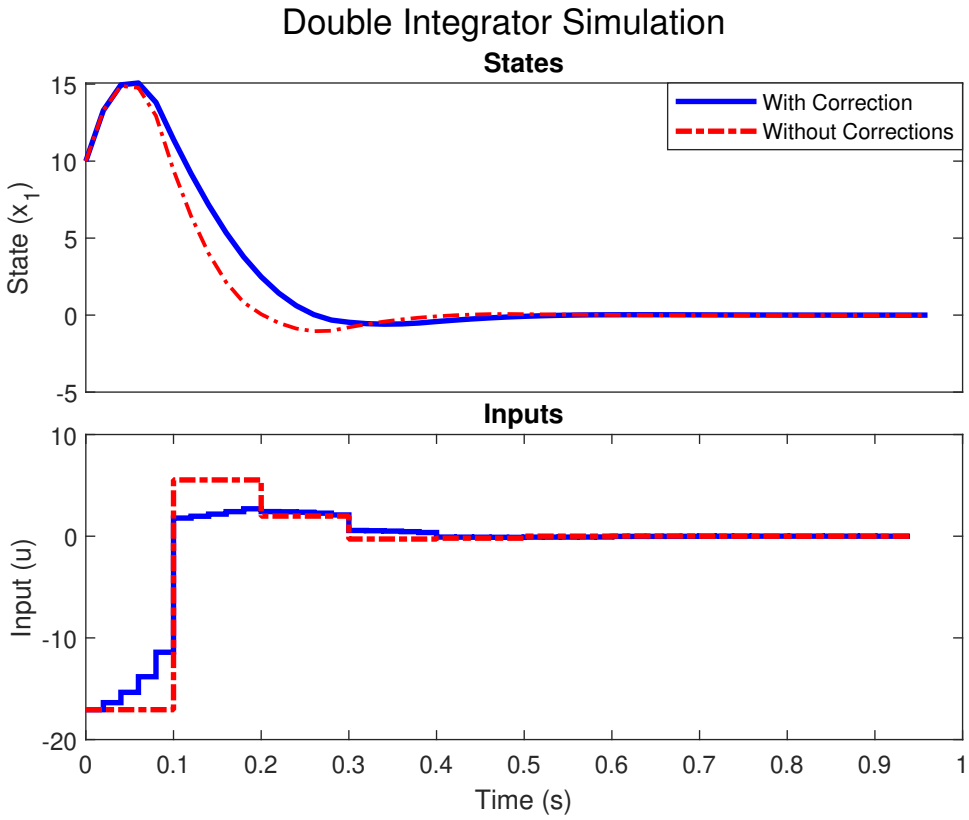


Figure 7.4: Example Simulation of Double Integrator System with and without corrections

7.4 Summary

In summary, this chapter presented one possible solution to the infinite horizon Ideal MWB problem which can be used to obtain a set of infinite horizon costs that when applied appropriately, represent the infinite horizon solution of the Ideal MWB approach embedded into the predictions of the optimisation which consequently result in stronger nominal stability guarantees for the nominal solution. Moreover, the developed solution includes the acquisition of optimal feedback gains for the entire approach which could potentially be used to generate blocked pre-stabilised prediction models. The resulting models could then be used to obtain invariant sets which contain the Maximum Admissible Set (MAS), (or more likely, a set of Maximum Admissible Sets for each time-step N_{b_0}) specific for the Ideal MWB. This however was outside the scope of this thesis and left for future work.

The chapter includes the development of the proposed solution, along with algorithm 7.1 which implements it. Moreover, it includes a case study of a linear double integrator system which was used to demonstrate the validity of the proposed solution along with relevant details related for its implementation in the framework presented in chapter 5 where the advantages of being able to obtain an infinite horizon solution were seen.

Chapter 8

Blocked Closed Loop Dual Mode Nonlinear Model Predictive Control with Shifting Strategies: The Combined Approach

Having observed the benefits of the Shifting Strategy for Blocked NMPC solutions presented in chapter 5, along with the benefits of the generalisable Closed Loop Dual Mode NMPC Prestabilisation Scheme of chapter 6 for handling unstable systems with condensing-based NMPC, and the ability to obtain the Infinite Horizon Solution of blocked linear MPC by using open loop and closed loop models, presented in chapter 7; this chapter aims at merging these methods into a combined approach: the "Blocked Closed Loop Dual Mode Nonlinear Model Predictive Control with Shifting Strategy", potentially merging the advantages and disadvantages from all the other strategies.

Some of the key properties of the proposed method to be discussed in this chapter are:

1. The equality between closed loop and open loop solutions, as derived in chapter 6, theorem 6.2.
2. Low condition number of the Hessian as inherited from methodology of chapter 6.
3. Slightly higher computation times due to use of an augmented state, when comparing to solutions without using the augmented state.
4. Slightly higher computation times due to the pre-stabilisation steps, when comparing to non-prestabilised solutions, as discussed in chapter 6.
5. Significantly lower computation times due to the input-blocking structure and the shifting inequality constraints of chapter 5.
6. Nominal stability and recursive feasibility guarantees for the shooting points inherited from the shifting strategy presented in chapter 5, and the dual mode terminal weight of chapter 7.
7. Ability to implement blocked closed loop solutions in highly unstable systems, something which otherwise could not be possible.

As this is a "gathering" chapter, it is kept brief. The chapter focuses mainly on the required methodological and algorithmic steps to implement the proposed approach, and discusses some of the expected properties along with some examples. Nonetheless, its contents represent a key contribution of this thesis which include the philosophy itself, along with the modelling and optimisation methodology, described in section 8.1; the equality of the solution theorem 8.1 (given without proof as inherits this property from theorem 6.2 of chapter 6), required to preserve the nominal stability and recursive feasibility properties of the original Shifting Strategy presented in chapter 5; and finally, the overall algorithms required for its implementation, given in section 8.2.

The chapter is organised as follows: Section 8.1 introduces the general optimisation of interest, and develops the prediction models containing the embedded blocked closed loop, along with several intermediate steps required for proper operation of the final algorithms. Moreover, it derives the resulting optimisation to be used, and presents theorem 8.1 briefly. Section 8.2 introduces the algorithms required for the efficient implementation of the proposed approach using the RTI Scheme. This includes a further extension to the already extended $O(N)$ and $O(N^2)$ algorithms of the Shifting Strategy presented in chapter 5, adjusting them to use the proposed method; a set of core algorithms; and finally, the algorithms for the preparation and feedback phases of the RTI Scheme. The section concludes with a generic computation analysis. Section 8.3 presents the case study of an inverted pendulum where the properties of chapters 5 and 6 are clearly shown to be inherited from the respective methodologies, as per design. Section 8.4 presents the case study of the nonlinear ball-plate system from chapter 6, a system in which the standard blocking approach of chapter 5 could not be applied due to a significant underlying numeric conditioning problem. Finally, section 8.5 presents a summary of the contributions.

8.1 Closed Loop Blocked Prediction Models and Optimisation

Following a similar approach as in chapter 7, the proposed methodology relies on an augmented state of the form:

$$z_k = \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix} \tag{8.1}$$

and selects between two possible discrete system dynamics (f_z^1 or f_z^2) of the form:

$$z_{k+1} = f_z(z_k, u_k) = \begin{cases} f_z^1(z_k, u_k) = \begin{bmatrix} f(x_k, u_k) \\ u_k \end{bmatrix} & \text{(Closed Loop)} \\ f_z^2(z_k, u_k = u_{k-1}) = \begin{bmatrix} f(x_k, u_k = u_{k-1}) \\ u_k = u_{k-1} \end{bmatrix} & \text{(Open Loop)} \end{cases} \tag{8.2}$$

depending on whether it is desired to have a decision (u_k) available (Closed Loop - f_z^1), e.g. when the prediction step is at the beginning of a block; or whether it is required to maintain the previous control action ($u_k = u_{k-1}$) constant, ie. embedding the blocking equality (Open Loop - f_z^2). This type of model will simplify the representation of the prediction models defined in (8.5), as well as simplify the DARE procedure (8.7) for embedding the blocked feedback control law.

As we have redefined the state variable, we are now interested in optimisations of the form:

$$J = (Z_r - \hat{Z})^T W (Z_r - \hat{Z}) \tag{8.3a}$$

$$s.t. \quad z_k = z_0 \tag{8.3b}$$

$$\hat{z}_{k+i} = f_z(\hat{z}_{k+i-1}, \hat{u}_{k+i-1}) \quad \forall i = [1, N_p] \tag{8.3c}$$

$$Z_{min} \leq \hat{Z} \leq Z_{max} \tag{8.3d}$$

where $\hat{Z} = [\hat{z}_{k+1}^T, \cdots, \hat{z}_{k+N_p}^T]^T \in \mathbb{R}^{N_p(n_x+n_u)}$ is a vector containing the predictions of the augmented state variable; $Z_r = [z_{r_{k+1}}^T, \cdots, z_{r_{k+N_p}}^T]^T \in \mathbb{R}^{N_p(n_x+n_u)}$ is a vector of references for the augmented state variables; $W = blkdiag([w_{k+1}, \cdots, w_{k+N_p}]) \in \mathbb{R}^{N_p(n_x+n_u)\times N_p(n_x+n_u)}$ is the weighting matrix penalising the augmented state deviations, ie. penalising states and inputs deviations from their references, for all practical purposes considered block diagonal formed with the Q and R matrices of the standard NMPC in the form of: $W = blkdiag([q_{k+1}, r_k, q_{k+2}, r_{k+1}, \cdots, q_{k+N_p}, r_{k+N_p-1}])$; (8.3b) is the initial condition; (8.3c) are the system dynamics; and (8.3d) are the inequality constraints of the optimisation.

Remark 8.1. *Blocked Dual Mode Terminal Weight*
As with chapter 6, we propose the dual mode framework for NMPC by imposing the infinite horizon terminal weight (w_{k+N_p}) obtained in chapter 7. Note that this terminal weight should only be imposed when the predicted state at the final step ($\hat{z}_{k+N_p}$) has reached the terminal region, ie. when the state enters a quasi-linear zone where the terminal weight is indeed valid. Thus, the proposed approach only adds this nominal stability guarantee method when the system can reach this condition. Otherwise, long horizons and/or other terminal conditions such as zero-terminal constraints still need to be used.

For this optimisation we also use definitions 5.1 and 5.2 of chapter 5 to embed the Shifting Strategy on the blocks and inequality constraints in an *absolute time frame*, and assume the Ideal Prediction Horizon of theorem 5.1 is used. Moreover, we assume that the provided nominal input $\bar{U}$ is aligned with the blocks for the relative and non-relative solution to be identical as discussed in remark 5.1. Finally, to simplify the mathematical representation of the prediction models, the proposed method uses the concept of the "inner block position index" ($N_{b_i}^k$) introduced in the general method of chapter 5, subsection 5.2.1, described by:

$$N_{b_i}^k = \begin{cases} N_{b_0} & k = 0 \quad \text{(starting condition)} \\ 1 & N_{b_i}^{k-1} = N_b \; AND \; k > 0 \\ N_{b_i}^{k-1} + 1 & N_{b_i}^{k-1} < N_b \; AND \; k > 0 \\ & \forall k = [0 \rightarrow N_p - 1] \end{cases} \tag{8.4}$$

This index ($N_{b_i}^k$) represents the "virtual" position of the inputs at a given block for each time-step ($k = 0 \rightarrow N_p - 1$) when moving forward along the prediction horizon, starting from an initial block position index (N_{b_0}) until reaching the block limit and resetting ($N_{b_i} = N_{b_0} \rightarrow N_b, 1 \rightarrow N_b, 1, \cdots$). Note that the range of $N_{b_0}/N_{b_i} \leftarrow [1 \rightarrow N_b]$ is used as discussed in remark 5.6.

To illustrate the use of equation (8.4), an example of the resulting sequences for all initial block position indexes $N_{b_0} = [1 \rightarrow N_b]$, using a block size of $N_b = 4$, and Ideal Prediction Horizon of $N_p = 13$ is given in table 8.1. Note that this is similar to that of table 5.4, but is re-included in this chapter for it to be self-contained. This set of values of $N_{b_i}^k$ basically help identify which points in time represent the beginning of a new block through the use of if-then conditional statements: eg. if ($k = 0$ OR $N_{b_i} = 1$), then "new block", something which can be appreciated in table 8.1, and can also be derived from equation (8.4) itself. Depending on the values of $N_{b_i}^k$ at a given step k, a different model will be selected or a different computation will be performed in the final algorithms. Note that for this particular example, the number of blocks (or decision variables in the case of $n_u = 1$) with the selected block size and ideal prediction horizon would be exactly $N_{E_{\mathbb{N}}} = \left\lceil \frac{N_p}{N_b} \right\rceil = 4 \; \forall N_{b_0} = [1 \rightarrow 4]$, something which can be clearly appreciated by the 4 coloured cells of the table.

N_{b_0}	k	0	1	2	3	4	5	6	7	8	9	10	11	12
$(N_{b_0} = 1)$	$N_{b_i}^k$	1	2	3	4	1	2	3	4	1	2	3	4	1
$(N_{b_0} = 2)$	$N_{b_i}^k$	2	3	4	1	2	3	4	1	2	3	4	1	2
$(N_{b_0} = 3)$	$N_{b_i}^k$	3	4	1	2	3	4	1	2	3	4	1	2	3
$(N_{b_0} = 4)$	$N_{b_i}^k$	4	1	2	3	4	1	2	3	4	1	2	3	4

Table 8.1: Example of all possible sequences for the MWB index $N_{b_i}^k$ with a block size of $N_b = 4$, and an ideal prediction horizon of $N_p = 13$. Each block is represented by the coloured cells

The proposed methodology must then provide the initial block position index $N_{b_0} = [1 \rightarrow N_b]$ in sequence, resetting when reaching the block limit and repeating infinitely as the horizon moves forward to embed the blocked input parameterisation whilst retaining the nominal stability and recursive feasibility properties discussed in chapter 5. Moreover, the method embeds a feedback correction $\delta u_k = -K_k \delta z_k + \delta c_k$, aimed at pre-stabilising the multiple shooting trajectory itself as discussed in chapter 6, with the main difference of only applying feedback at the beginning of each block, allowing the feedback-corrected control action to be maintained constant through the open-loop dynamics of (8.2). As discussed previously, this happens when $N_{b_i}^k = 1$ or $k = 0$, ie. at the beginning of a block, or in the very first step of the horizon. In order to achieve this efficiently, we utilise the concept of open loop and closed loop models as in chapter 7 where the prediction models change depending on the position of the block $N_{b_i}^k$, and are able to maintain constant the input through the "integrator-with-no-input" type of dynamics of function (8.2). The linearised prediction models of function (8.2) at all future time steps $k = 0 \rightarrow N_p - 1$ are then given by:

$$\hat{z}_{k+1} = \begin{cases} \bar{z}_{k+1} + A_{CL_k} \delta \hat{z}_k + B_{CL_k} \delta \hat{u}_k + d_{k+1} & N_{b_i}^k = 1 \; OR \; k = 0 \\ \bar{z}_{k+1} + A_{OL_k} \delta \hat{z}_k + d_{k+1} & else \end{cases} \tag{8.5a}$$

$$d_{k+1} = \begin{cases} f_z^1(\bar{x}_k, \bar{u}_k) - \bar{z}_{k+1} & N_{b_i} = 1 \; OR \; k = 0 \\ f_z^2(\bar{x}_k, \bar{u}_{k-1}) - \bar{z}_{k+1} & else \end{cases} \tag{8.5b}$$

$$\delta \hat{u}_k = \begin{cases} -K_k \delta \hat{z}_k + \delta \hat{c}_k & N_{b_i}^k = 1 \; OR \; k = 0 \\ 0 & else \end{cases} \tag{8.5c}$$

$$\forall k = [0 \rightarrow N_p - 1]$$

where $\bar{z}_{k+1}$ is the nominal augmented state at time step $k+1$; δz_k is the augmented state deviation at time k; $\delta\hat{u}_k$ is the input deviation at time step k; d_{k+1} is the multiple-shooting offset at time $k+1$; and A_{OL_k}, A_{CL_k} and B_{CL_k} are the partial derivatives of the open loop (OL) and closed loop (CL) models of (8.2), respectively, and are defined as:

$$A_{OL_k} = \left.\frac{\partial f_z^2(\hat{z}_k, \hat{u}_k)}{\partial \hat{z}_k}\right|_{\substack{\hat{x}_k=\bar{x}_k \\ \hat{u}_k=\bar{u}_{k-1}}} = \begin{bmatrix} A_k & B_k \\ 0 & I \end{bmatrix} \tag{8.6a}$$

$$A_{CL_k} = \left.\frac{\partial f_z^1(\hat{z}_k, \hat{u}_k)}{\partial \hat{z}_k}\right|_{\substack{\hat{x}_k=\bar{x}_k \\ \hat{u}_k=\bar{u}_k}} = \begin{bmatrix} A_k & 0 \\ 0 & 0 \end{bmatrix} \tag{8.6b}$$

$$B_{CL_k} = \left.\frac{\partial f_z^1(\hat{z}_k, \hat{u}_k)}{\partial \hat{u}_k}\right|_{\substack{\hat{x}_k=\bar{x}_k \\ \hat{u}_k=\bar{u}_k}} = \begin{bmatrix} B_k \\ I \end{bmatrix} \tag{8.6c}$$

with A_k and B_k being the state and input partial derivatives used by the standard NMPC.

On the other hand, K_k is a feedback control law obtained from using the Time-Varying Discrete Algebraic Ricatti Equation (DARE) backwards starting from the terminal weight $P_{N_p} = w_{N_p}$, and applying feedback only at the beginning of a block, but propagating the cost P_k through the open loop dynamics at all the other points as performed in chapter 7, defined as:

$$K_k = \begin{cases} (B_{CL_k}^T P_{k+1} B_{CL_k})^{-1} B_{CL_k}^T P_{k+1} A_{CL_k} & N_{b_i}^k = 1 \; OR \; k = 0 \\ \mathbb{O} & else \end{cases} \tag{8.7a}$$

$$P_k = \begin{cases} w_k + A_{CL_k}^T P_{k+1} A_{CL_k} + A_{CL_k}^T P_{k+1} B_{CL_k} K_k & N_{b_i}^k = 1 \; OR \; k = 0 \\ w_k + A_{OL_k}^T P_{k+1} A_{OL_k} & else \end{cases} \tag{8.7b}$$

$$\forall k = [0 \rightarrow N_p - 1]$$

Indeed, by using this approach, the solution naturally embeds the cost of the blocked inputs on the DARE recursion whilst imposing feedback only at the start of each block, which in turns leads to obtaining optimal blocked feedback control gains, as discussed in chapter 7.

Substituting the value of K_k from (8.7a) in model (8.5a), results in the final linearised blocked closed loop prediction model:

$$\hat{z}_{k+1} = \begin{cases} \bar{z}_{k+1} + \Phi_k \delta\hat{z}_k + B_{CL_k}\delta\hat{c}_k + d_{k+1} & N_{b_i}^k = 1 \; OR \; k = 0 \\ \bar{z}_{k+1} + \Phi_k \delta\hat{z}_k + d_{k+1} & else \end{cases} \tag{8.8a}$$

$$d_{k+1} = \begin{cases} f_z^1(\bar{x}_k, \bar{u}_k) - \bar{z}_{k+1} & N_{b_i}^k = 1 \; OR \; k = 0 \\ f_z^2(\bar{x}_k, \bar{u}_{k-1}) - \bar{z}_{k+1} & else \end{cases} \tag{8.8b}$$

$$\Phi_k = \begin{cases} A_{CL_k} - B_{CL_k} K_k & N_{b_i}^k = 1 \; OR \; k = 0 \\ A_{OL_k} & else \end{cases} \tag{8.8c}$$

$$\forall k = [0 \rightarrow N_p - 1]$$

Notice that this model only has control actions ($\delta\hat{c}_k$) available at specific points ($N_{b_i}^k = 1$ OR $k = 0$), ie. $\delta\hat{c}_k = 0$ elsewhere, which causes the input-to-state prediction matrix H to contain zero-columns for values of $\delta\hat{c}_k = 0$, all of which can be disregarded as they were considered zero by design. This allows the "compressed" prediction matrix $H_{\mathbb{N}}$ containing only the values of non-zero columns to be given directly. Moreover, notice that the blocking approach with no feedback discussed in chapter 5 can also be applied with this model simply by enforcing $K_k = \mathbb{O}\ \forall k = [0 \to N_p - 1]$ in 8.5. Finally, as with all the other methods discussed in this thesis, the single-shooting approach can be applied by using the simulated value at each time step prediction, instead of using the nominal value obtained in the previous iteration, thus causing all offsets (d_k) to be zero, as discussed in chapter 3.

By propagating the models resulting from all possible absolute time-steps ($N_{b_0} = [1 \to N_b]$) N_p steps ahead, results in N_b different prediction models which depend on N_{b_0}, and are defined in condensed format (8.9) given by:

$$\hat{Z}^{N_{b_o}} = \bar{Z} + \delta\hat{Z} = \bar{Z} + \underbrace{D + G\delta z_0 + H_{\mathbb{N}}\delta\hat{\mathbb{C}}}_{\delta\hat{Z}} \qquad \forall N_{b_0} = [1 \to N_b] \tag{8.9}$$

where $\bar{Z} = [\bar{z}_{k+1}^T, \cdots, \bar{z}_{k+N_p}^T]^T \in \mathbb{R}^{N_p(n_x+n_u)}$ is a vector containing the nominal augmented state (state and inputs) predictions, $\delta\hat{\mathbb{C}} \in \mathbb{R}^{N_E n_u}$ is a vector of the feedback decision variables to be used for the optimisation, ie. only those which were not imposed to be zero by design; $\delta z_0 = z_0 - \bar{z}_0$ is the initial condition mismatch which forms part of the RTI Scheme; and $D \in \mathbb{R}^{N_p(n_x+n_u)}$, $G \in \mathbb{R}^{N_p(n_x+n_u)\times(n_x+n_u)}$ are a vector and a matrix, respectively, defined as:

$$D = \begin{bmatrix} \bar{d}_1^T & \bar{d}_2^T & \cdots & \bar{d}_{N_p}^T \end{bmatrix}^T \quad G = \begin{bmatrix} g_1^T & g_2^T & \cdots & g_{N_p}^T \end{bmatrix}^T \tag{8.10a}$$

where the inner matrices/vectors are defined through the recursions:

$$\bar{d}_k = \begin{cases} d_k & k = 1 \\ d_k + \Phi_{k-1}\bar{d}_{k-1} & k > 1 \end{cases} \tag{8.11a}$$

$$g_k = \begin{cases} \Phi_{k-1} & k = 1 \\ \Phi_{k-1} g_{k-1} & k > 1 \end{cases} \tag{8.11b}$$

$$\forall k = [1, N_p]$$

using the proposed blocked closed loop matrix Φ_k of equation (8.8c).

Matrix $H_{\mathbb{N}} \in \mathbb{R}^{N_p(n_x+n_u)\times N_{E_{\mathbb{N}}} n_u}$ deserves special attention as it presents the special structure of the Ideal MWB Scheme (5.42) introduced in chapter 5 with the inner matrices defined through the following recursions as:

$$h_{k,j} = \begin{cases} \mathbb{O} & j^k > n^k \\ B_{CL_{k-1}} & (N_{b_i}^{k-1} = 1\ AND\ j^k = n^k)\ OR\ (k = 1) \\ \Phi_{k-1} h_{k-1,j} & N_{b_i}^{k-1} > 1\ OR\ j^k < n^k \end{cases} \tag{8.12}$$

$$\forall k = [1 \to N_p] \quad \forall j = [1 \to N_{E_{\mathbb{N}}}]$$

where j^k represent the j_{th} column in the k_{th} row, ie. at k_{th} time-step, and n^k represents the column on which the block at the k_{th} time-step is embedded, similar to the block counter (n) introduced in the general method 5.2.1, defined as:

$$n^k = \begin{cases} 1 & k = 1 \quad \text{(starting condition)} \\ n^{k-1} + 1 & N_{b_i}^{k-1} = 1 \; AND \; k > 1 \\ n^{k-1} & N_{b_i}^{k-1} > 1 \; AND \; k > 1 \end{cases} \tag{8.13}$$

$$\forall k = [1 \rightarrow N_p] \tag{8.14}$$

In simple terms, n^k represents a column index which increments every time there is a new block. To illustrate this, consider the values of $N_{b_i}^k$ from table 8.1. The required values of n^k for that example are given in table (8.2). Notice there is a difference in the k range used for that example (ie. $0 \rightarrow 12$ and $1 \rightarrow 13$). This is because of how the matrix $H_{\mathbb{N}}$ range was defined with future "states" in mind (ie. $\hat{x}_{k+1}, \hat{x}_{k+2}, \cdots, \hat{x}_{k+N_p}$), and $N_{b_i}^k$ was defined with inputs/decision variables in mind (ie. $\hat{u}_k, \hat{u}_{k+1}, \cdots, \hat{u}_{k+N_p-1}$). Nonetheless, they represent the same prediction steps.

N_{b_0}	k	1	2	3	4	5	6	7	8	9	10	11	12	13
$(N_{b_0} = 1)$	n^k	1	1	1	1	2	2	2	2	3	3	3	3	4
$(N_{b_0} = 2)$	n^k	1	1	1	2	2	2	2	3	3	3	3	4	4
$(N_{b_0} = 3)$	n^k	1	1	2	2	2	2	3	3	3	3	4	4	4
$(N_{b_0} = 4)$	n^k	1	2	2	2	2	3	3	3	3	4	4	4	4

Table 8.2: Continuation example from table (8.1) of all possible sequences for the MWB column index n^k starting from all possible initial block index $N_{b_0} = [1 \rightarrow 4]$ with a block size of $N_b = 4$, and an ideal prediction horizon of $N_p = 13$. Each block is represented by coloured cells.

Having defined the condensed prediction models (8.9), the standard steps can be taken where the models are substituted in cost function (8.3), and constant terms are disregarded to obtain the standard QP form. This leads to N_b different costs functions of the form (8.15) as discussed in the Shifting Strategy optimisation framework of section 5.1.4, each of which depends on the initial block index N_{b_0}, and are defined as:

$$J^{N_{b_0}} = \frac{1}{2}\delta\hat{\mathbb{C}}^T E_{\mathbb{N}} \delta\hat{\mathbb{C}} + \delta\hat{\mathbb{C}}^T f_{\mathbb{N}} \qquad \forall N_{b_0} = [1 \rightarrow N_b] \tag{8.15a}$$

$$s.t. \quad M_{\mathbb{N}}\delta\hat{C} \leq \gamma \tag{8.15b}$$

$$E_{\mathbb{N}} = H_{\mathbb{N}}^T W H_{\mathbb{N}} \tag{8.15c}$$

$$f_{\mathbb{N}} = -H_{\mathbb{N}}^T W (Z_r - \bar{Z} - D - G\delta z_0) \tag{8.15d}$$

$$M_{\mathbb{N}} = \begin{bmatrix} (H_{\mathbb{N}})_u \\ -(H_{\mathbb{N}})_u \\ (H_{\mathbb{N}})_x \\ -(H_{\mathbb{N}})_x \end{bmatrix} \quad \gamma = \begin{bmatrix} U_{max} - \bar{Z}_u - D_u - G_u\delta z_0 \\ \bar{Z}_u + D_u + G_u\delta z_0 - U_{min} \\ X_{max} - \bar{Z}_x - D_x - G_x\delta z_0 \\ \bar{Z}_x + D_x + G_x\delta z_0 - X_{min} \end{bmatrix} \tag{8.15e}$$

where $E_{\mathbb{N}} \in \mathbb{N}^{N_{E_{\mathbb{N}}} n_u \times N_{E_{\mathbb{N}}} n_u}$ is the "compressed" Hessian; $f_{\mathbb{N}}$ is the "compressed" linear term; $M_{\mathbb{N}}$ is the "compressed" constraints matrix; and γ is the constraints vector.

We assume here that the values of $(H_{\mathbb{N}})_u, , \bar{Z}_u, D_u$ and G_u represent the rows of $H_{\mathbb{N}}, \bar{Z}, D$ and G, respectively, related to the input constraints (U_{max} and U_{min}). Likewise, the values of $(H_{\mathbb{N}})_x, \bar{Z}_x, D_x$ and G_x represent the rows of $H_{\mathbb{N}}, \bar{Z}, D$ and G related to the state constraints (X_{max} and X_{min}).

Moreover, to be able to apply the shifting constraints strategy presented in chapter 5, the constraints must be selected according to definition (5.2), which essentially selects the constraints at the end of each block, as well as the constraints at the end of the prediction horizon (which may or may not be coincident with the end of a block). The resulting sequence of end-points of each block depends on $N_{b_i}^k$, and happens whenever $N_{b_i}^k = N_b$ when using the range $k = [1 \rightarrow N_p]$ or $k = N_p$, as defined by (8.4 shifted in k). This task is fundamentally performed by algorithm 5.4 introduced in chapter 5, where at each time step the algorithm checks a simple conditional flag of the form:

$$flag = \begin{cases} \text{Include Constraint} & N_{b_i}^k = N_b \; OR \; k = N_p \\ \text{Ignore Constraint} & else \end{cases} \qquad \forall k = [1 \rightarrow N_p] \tag{8.16}$$

The proposed shifting constraints approach would then select only the predictions at the the end of each block, resulting in the selected matrices and vectors $(H_{\mathbb{N}})_{u-sel}, \bar{Z}_{u-sel}, D_{u-sel}, G_{u-sel}$ related to the inputs, and $(H_{\mathbb{N}})_{x-sel}, \bar{Z}_{x-sel}, D_{x-sel}$ and G_{x-sel}, related to the states, to be included in the optimisation.

Once the optimal $\delta\hat{\mathbb{C}}^*$ has been found, the expansion step $\hat{Z}^* = \bar{Z} + D + G\delta z_0 + H_{\mathbb{N}}\delta\hat{\mathbb{C}}^*$ is applied, as in the standard multiple shooting method for $\hat{Z}^*$ to be used in the next iteration of the solution. Having calculated this expansion step, the nominal input $\bar{U}$, and the nominal state $\bar{X}$ can be extracted from $\hat{Z}^*$ for the next iteration. Only the first input is applied to the system, as applied by the standard "receding horizon" strategy, and the whole process is repeated in the next time step. To retain the nominal stability and recursive feasibility properties, the initial block index N_{b_0} must then be moved forward with (8.17) at every time-step, as required by the Shifting Strategy presented in chapter 5.

$$N_{b_0}^+ = \begin{cases} 1 & N_{b_0} = N_b \\ N_{b_0} + 1 & N_{b_0} < N_b \end{cases} \tag{8.17}$$

Theorem 8.1. *Equality of the Blocked Pre-stabilised Solution*
The solution with and without the blocked pre-stabilisation is exactly the same. (Given without proof)

As stated in theorem 6.2 of chapter 6, "if the new decision variables can replicate the original variables, they will !!!". Thus, the solutions are expected to be the same given the same amount of decision variables as in the Ideal MWB approach of chapter 5 is available, and the original blocked decision can be exactly replicated with the available decision variables. This allows all the nominal stability and recursive feasibility properties for the original blocked approach with shifting constraints presented in chapter 5 to be retained.

This property was consistently observed during the tests of this proposed method, and can be appreciated in figure 8.2. Nonetheless, a formal proof is not given.

8.2 Algorithm Details and Autogeneration

Having defined the theory and formulae that supports the proposed approach, we can now proceed to define the required algorithms for its efficient implementation using the RTI Scheme that will allow the development of an auto-generation toolkit which will be used for benchmarking the proposed approach. Thus, this section will provide a set of algorithms to be used which include: an extension to the already extended $O(N)$ and $O(N^2)$ algorithms presented in chapter 5 for the proposed Ideal MWB approach; a set of 4 "core" algorithms that perform various important operations; and the preparation and feedback phases algorithms of the RTI Scheme. The section concludes with a generic computation analysis on the relevant algorithms.

It is worth mentioning that all the algorithms are supported by the general method introduced in section 5.2.1 of chapter 5, based on the "inner block position index" (N_{b_i}) which was used for the proposed modelling of this chapter, as seen eg. in equations (8.4), (8.5) and (8.7). Moreover, all the algorithms use the standard notation A_k and B_k to avoid saturation for representing the augmented matrices, eg. $A_{CL_k}, B_{CL_k}, A_{OL_k}$, for which the inner definitions (8.6) must be used.

8.2.1 Further Extension of $O(N)$ and $O(N^2)$ Algorithms

Because, this method follows the same structure of $H_{\mathbb{N}}$, the entire procedure used for the derivation of the extension of the $O(N)$ and $O(N^2)$ algorithms for the Ideal MWB presented in chapter 5, section 5.2.3 is valid, with the understanding that $Q = W$, $R = \mathbb{O}$, $A_k = \Phi_k$ and $B_k = \mathbb{O}$ at some points.

Algorithm 8.1: Closed Loop Ideal MWB $O(N)$ Condensing Algorithm

Data: $W, \Phi_k, B_k, Z_e, N_p, N_b, N_{b_0}$

begin

- $N_{b_i} = N_{b_0}$; // Initialise inner block position index
- $n = N_{E_{\mathbb{N}}}$; // Initialise row of $f_{\mathbb{N}}$
- $\tilde{w}_{N_p} = w_{N_p} Z_{e_{N_p}}$; // Initial value of the dummy variable

// For loop running backwards $k = N_p, N_p - 1, \cdots, 2$

- **for** $k = N_p$ **to** 2 **do**
  - **if** $N_{b_i} = 1$ **then**
    - $(f_{\mathbb{N}})_n = -B_{k-1}^T \tilde{w}_k$; // Calculate linear term component
  - **end**
  - $\tilde{w}_{k-1} = w_{k-1} Z_{e_{k-1}} + \Phi_{k-1}^T \tilde{w}_k$; // Propagate recursively
  - $N_{b_i}^+ = N_{b_i}^- - 1$; // Iterate inner block position index backwards
  - **if** $N_{b_i} < 1$ **then**
    - $n^+ = n^- - 1$; // Decrease row of $f_{\mathbb{N}}$ elements
    - $N_{b_i} = N_b$; // Reset inner block position index to N_b
  - **end**
- **end**
- **if** $N_{b_i} = 1$ **then**
  - $(f_{\mathbb{N}})_0 = -B_0^T \tilde{w}_0$; // Calculate first component of linear term $(f_{\mathbb{N}})_0$
- **end**

end

Result: $f_{\mathbb{N}}$

However, given that matrix B_k is non-zero only at the start of a block, eg. $N_{b_i} = 1$, the algorithms were no longer required to calculate the Hessian terms recursively, eg. as in equation (5.50). Instead, the algorithms now only have to check for the start of a block with conditions $k = 1$ or $N_{b_i} = 1$ to perform a single calculation of the Hessian term. This allows a slight reduction in the number of required operations. Thus, the algorithms were adjusted to look for these conditions which can be seen throughout the codes, eg. in lines 9-10 of algorithm 8.2. This extended version of the already extended $O(N)$ and $O(N^2)$ algorithms is given in algorithms 8.1 and 8.2.

Algorithm 8.2: Closed Loop Ideal MWB $O(N^2)$ Condensing Algorithm

Data: $H_{\mathbb{N}}, W, \Phi_k, B_k, N_p, N_b, N_{b_0}$
1 **begin**
2 $N_{b_i} = N_{b_0}$; // Initialise inner block position index
3 $n = N_{E_{\mathbb{N}}}$; // Initialise row of $E_{\mathbb{N}}$
4 **for** $i = 1$ **to** n **do**
5 $\tilde{w}_{N_p,i} = w_{N_p}(h_{\mathbb{N}})_{N_p,i}$; // Initialise last row of $\tilde{W}$ ($\tilde{w}_{N_p,1\to n}$ elements)
6 **end**
// Main for loop running backwards $k = N_p, N_p - 1, \cdots, 2$
7 **for** $k = N_p$ **to** 2 **do**
8 **for** $i = 1$ **to** $n - 1$ **do**
9 **if** $N_{b_i} = 1$ **then**
10 $(E_{\mathbb{N}})_{n,i} = B_{k-1}^T \tilde{w}_{k,i}$; // Calculate Hessian component
11 **end**
12 $\tilde{w}_{k-1,i} = w_{k-1}(h_{\mathbb{N}})_{k-1,i} + \Phi_{k-1}^T \tilde{w}_{k,i}$; // Propagate recursively
13 **end**
14 **if** $N_{b_i} = 1$ **then**
15 $(E_{\mathbb{N}})_{n,n} = B_{k-1}^T \tilde{w}_{k,n}$; // Calculate Hessian diagonal component
16 **else**
17 $\tilde{w}_{k-1,n} = w_{k-1}(h_{\mathbb{N}})_{k-1,n} + \Phi_{k-1}^T \tilde{w}_{k,n}$; // Propagate recursively
18 **end**
19 $N_{b_i}^+ = N_{b_i}^- - 1$; // Iterate inner block position index backwards
20 **if** $N_{b_i} < 1$ **then**
21 $n^+ = n^- - 1$; // Decrease row of $E_{\mathbb{N}}$
22 $N_{b_i} = N_b$; // Reset inner block position index to N_b
23 **end**
24 **end**
25 $(E_{\mathbb{N}})_{0,0} = B_0^T \tilde{w}_{0,0}$; // Calculate first diagonal element outside main loop
26 **end**
Result: $E_{\mathbb{N}}$

8.2.2 Core Algorithms

In addition to the 2 novel algorithms (8.1 and 8.2), 4 additional "core" algorithms were required for implementing the proposed approach in the RTI algorithms of subsection 8.2.3 in a structured manner. The developed algorithms are (given in order introduction and usage in the RTI): 1. Forward Propagation, 2. Ideal MWB Time-Varying DARE, 3. Closed Loop Ideal MWB $H_{\mathbb{N}}$ Computation, and 4. $H_{\mathbb{N}}\delta\hat{\mathbb{C}}^*$ Decompression and Expansion. All the algorithms take an entire page to improve the spacing and flow of the section.

Given it is required to select between 2 different "Simulation and Linearisation" models, as established from equation (8.2), the standard Forward Propagation algorithm 3.4 proved incomplete. Thus, algorithm 8.3 performs this fundamental task by using the "inner block position index" along with the required conditionals to select between the 2 models of (8.3c). This can be seen explicitly in lines 4-10 of the algorithm. As in the standard algorithm, it considers the "single shooting" case visible in lines 11-13.

Algorithm 8.3: Closed Loop Ideal MWB Forward Simulation Algorithm

Data: $\bar{Z}, \bar{U}, \bar{z}_0, N_p, N_b, N_{b_0}$

1 **begin**
2 $N_{b_i} = N_{b_0}$; // Initialise inner block position index
3 **for** $k = 1$ **to** N_p **do**
/* Simulate And Linearise the system, eg. using algorithm 3.1 with the selected dynamics */
4 **if** $k = 1$ *OR* $N_{b_i} = 1$ **then**
5 $\tilde{z}_k = f_z^1(\bar{z}_{k-1}, \bar{u}_{k-1})$; // Use Closed Loop Dynamics
6 $A_{k-1} \leftarrow A_{CL_{k-1}}$; // Closed Loop State-Transition Matrix
7 $B_{k-1} \leftarrow B_{CL_{k-1}}$; // Closed Loop Input-to-State Matrix
8 **else**
9 $\tilde{z}_k = f_z^2(\bar{z}_{k-1}, \bar{u}_{k-2})$; // Use Open Loop Dynamics
10 $A_{k-1} \leftarrow A_{OL_{k-1}}$; // Open Loop State-Transition Matrix
11 $B_{k-1} \leftarrow \mathbb{O}$; // Open Loop Input-To-State Matrices
12 **end**
13 **if** *Single Shooting=1* **then**
14 $\bar{z}_k = \tilde{z}_k$; // Special case for the single shooting case
15 **end**
16 $d_k = \tilde{z}_k - \bar{z}_k$; // Calculate individual offset at each step
17 $N_{b_i}^+ = N_{b_i}^- + 1$; // Iterate inner block position index forward
18 **if** $N_{b_i} > N_b$ **then**
19 $N_{b_i} = 1$; // Reset inner block position index to 1
20 **end**
21 **end**
22 **end**

Result: A_k, B_k, d_k

On the other hand, because the proposed approach only applies feedback at the start of a block, the Time-Varying DARE algorithm 6.3 was required to be modified, although arguably can be applied as it is with the understanding that the feedback gain would be zero ($K_k = 0$) if the input matrix is zero ($B_k = 0$). However, given the indefiniteness of the required inverse $(B_k^T P_{k+1} B_k)^{-1}$, a proper modification was due. The modified algorithm 8.4 performs the feedback step only when the condition ($N_{b_i} = 1$ OR $k = 0$) is met, as seen in lines 5-11. Thus, the algorithm avoids the problem of the indefiniteness, whilst also reducing the computations that would be related to K_k.

Algorithm 8.4: Ideal MWB Dual Mode Time-Varying DARE Algorithm

Data: $A_k, B_k, W, N_p, N_b, N_{b_0}$

1 **begin**
2 $N_{b_i} = N_{b_0}$; // Initialise inner block position index
3 $P_{N_p} = w_{N_p}$; // Initial value for weighting matrix P_{k+1}
// For loop running backwards $k = N_p - 1, N_p - 1, \cdots, 0$
4 **for** $k = N_p - 1$ **to** 0 **do**
5 **if** $N_{b_i} = 1$ *OR* $k = 0$ **then**
6 $K_k = (B_k^T P_{k+1} B_k)^{-1} B_k^T P_{k+1} A_k$; // Calculate feedback gain
7 $\Phi_k = A_k - B_k K_k$; // Apply Feedback
8 **else**
9 $\Phi_k = A_k$; // No Feedback
10 **end**
11 $P_k = w_k + \Phi_k^T P_{k+1} \Phi_k$; // Propagate Weights
12 $N_{b_i}^+ = N_{b_i}^- - 1$; // Iterate inner block position index backwards
13 **if** $N_{b_i} < 1$ **then**
14 $N_{b_i} = N_b$; // Reset inner block position index to N_b
15 **end**
16 **end**
17 **end**

Result: Φ_k

Afterwards, algorithm 5.3 presents the modification of algorithm 5.3 which basically reduces to not requiring the recursive expression $(h_{\mathbb{N}})^{+}_{k,n} = (h_{\mathbb{N}})^{-}_{k,n} + B_{k-1}$ visible in line 13 of algorithm 5.3, which in essence means the "pink" part of equation (5.45) is never present in our approach given it is embedded in the augmented "integrator-type" matrix A_{OL} of equation (8.5). This again can save some of the required operations, thus decreasing computation times slightly. The algorithm, retains the "cleaning" operation seen in lines 15-17 required for the overlapping spaces of memory as explained in equation (5.46).

Algorithm 8.5: Closed Loop Ideal MWB Condensing $H_{\mathbb{N}}$ Matrix Calculation

Data: $\Phi_k, B_k, N_p, N_b, N_{b_0}$

1 **begin**
2 $N_{b_i} = N_{b_0}$; // Initialise inner block position index
3 $n = 1$; // Initialise columns of $H_{\mathbb{N}}$
4 $j = n$; // Initialise limit for recursive loop
5 **for** $k = 1$ **to** N_p **do**
6 **for** $i = 1$ **to** $j - 1$ **do**
7 $(h_{\mathbb{N}})_{k,i} = \Phi_{k-1}(h_{\mathbb{N}})_{k-1,i}$; // Propagate recursively
8 **end**
9 **if** $k = 1$ *OR* $N_{b_i} = 1$ **then**
10 $(h_{\mathbb{N}})_{k,n} = B_{k-1}$; // Initialise n_{th} column element
11 $j^{+} = j^{-} + 1$; // Increase limit for recursive loop
12 **end**
13 $N^{+}_{b_i} = N^{-}_{b_i} + 1$; // Iterate inner block position index forward
14 **if** $N_{b_i} > N_b$ **then**
15 **if** $n < N_{E_{\mathbb{N}}}$ **then**
16 $(h_{\mathbb{N}})_{2+(n-1)N_b \to nN_b+1, n+1} = 0$; // Clean elements of following column
17 **end**
18 $n^{+} = n^{-} + 1$; // Increase columns of $H_{\mathbb{N}}$
19 $N_{b_i} = 1$; // Reset inner block position index to 1
20 **end**
21 **end**
22 **end**

Result: $H_{\mathbb{N}}$

Finally, algorithm 8.6 performs the simultaneous Decompression and Expansion of the term $\delta\hat{\mathbb{C}}^*$ which is required to complete the predicted optimal correction to the nominal state trajectory, ie. $\hat{Z}^* = \bar{Z} + D + G\delta z_0 + H_{\mathbb{N}}\delta\hat{\mathbb{C}}^*$ of equation (8.9), which is then used to provide the guess for $\bar{Z}$ at the next iteration, as performed by the standard multiple-shooting approach. This algorithm can be seen as a combination of the tasks performed by algorithms 3.7, 5.5 and 6.6.

Algorithm 8.6: Closed Loop Ideal MWB $\delta\tilde{Z} = H_{\mathbb{N}}\delta\hat{\mathbb{C}}^*$ Decompression/Expansion Step

Data: $\Phi_k, B_k, \delta\hat{\mathbb{C}}^*, N_p, N_b, N_{b_0}$
begin
  $N_{b_i} = N_{b_0}$; // Initialise inner block position index
  $n = 1$; // Initialise working row of $\delta\hat{\mathbb{C}}^*$
  for $k = 1$ **to** N_p **do**
    if $k = 1$ **then**
      $\delta\tilde{z}_k^* = B_{k-1}\delta\hat{\mathbb{C}}_n^*$; // Initial value
    else
      if $N_{b_i} = 1$ **then**
        $\delta\tilde{z}_k = \Phi_{k-1}\delta\tilde{z}_{k-1} + B_{k-1}\delta\hat{\mathbb{C}}_n^*$; // Propagate recursively (with input)
      else
        $\delta\tilde{z}_k = \Phi_{k-1}\delta\tilde{z}_{k-1}$; // Propagate recursively (without input)
      end
    end
    $N_{b_i}^+ = N_{b_i}^- + 1$; // Iterate inner block position index forward
    if $N_{b_i} > N_b$ **then**
      $n^+ = n^- + 1$; // Increase working row of $\delta\hat{\mathbb{C}}^*$
      $N_{b_i} = 1$; // Reset inner block position index to 1
    end
  end
end
Result: $\delta\tilde{Z}$

8.2.3 RTI Algorithms

Having established the fundamental algorithms required for the implementation of the proposed Blocked Closed-Loop Dual-Mode NMPC approach with the Shifting Strategy of chapter 5, and using the RTI Scheme, the overall approach is finally provided in terms of the Preparation and Feedback Phases given in algorithms 8.7 and 8.8, respectively, both of which are based on previously presented algorithms to facilitate the verification process of each working part of the proposed approach.

Algorithm 8.7: Closed Loop Ideal MWB RTI NMPC Preparation Phase with Shifting Strategy

Data: $\bar{Z}, \bar{U}, \bar{\lambda}_{sel}, z_{-1}, u_{-1}, W, N_p, N_b, N_{b_0}$

begin

- $\bar{z}_0 = f_z^1(z_{-1}, u_{-1})$; // Calculate predicted state from previous state and input
- Shift $\bar{Z}$,$\bar{U}$ consistently ; // Initial Value Embedding
- **if** $N_{b_0} = 1$ **then**
  - Shift $\bar{\lambda}_{sel}$ consistently ; // Shifting Lagrange Multipliers theorem 5.2
- **end**
- $[A_k, B_k, d_k] = Forward(\bar{Z}, \bar{U}, \bar{z}_0, N_p, N_b, N_{b_0})$; // Run algorithm 8.3
- $[\Phi_k] = DARE(A_k, B_k, W, N_p, N_b, N_{b_0})$; // Run algorithm 8.4
- $[H_{\mathbb{N}}] = CalculateH_{\mathbb{N}}(\Phi_k, B_k, N_p, N_b, N_{b_0})$; // Run algorithm 8.5
- $[E_{\mathbb{N}}] = CalculateE_{\mathbb{N}}(H_{\mathbb{N}}, W, \Phi_k, B_k, N_p, N_b, N_{b_0})$; // Run algorithm 8.2
- // Form $(M_{\mathbb{N}})_{sel}$ using algorithm 5.4
- $(M_{\mathbb{N}})_{sel} = \begin{bmatrix}(H_{\mathbb{N}})^T_{u-sel} & -(H_{\mathbb{N}})^T_{u-sel} & (H_{\mathbb{N}})^T_{x-sel} & -(H_{\mathbb{N}})^T_{x-sel}\end{bmatrix}^T$

end

Result: $E_{\mathbb{N}}, (M_{\mathbb{N}})_{sel}, \Phi_k, B_k, d_k, \bar{z}_0, \bar{\lambda}_{sel}$

Algorithm 8.8: Closed Loop Ideal MWB RTI NMPC Feedback Phase with Shifting Strategy

Data: $z_0, \bar{z}_0, \bar{Z}, \bar{U}, \bar{\lambda}_{sel}, Z_r, E_{\mathbb{N}}, (M_{\mathbb{N}})_{sel}, \Phi_k, B_k, d_k, W, U_{max}, U_{min}, X_{max}, X_{min}, N_p, N_b, N_{b_0}$

begin

- $\delta z_0 = z_0 - \bar{z}_0$; // Calculate state deviation from measurement
- $[D] = FormD(\Phi_k, d_k, \delta z_0, N_p)$; // Run algorithm 3.6
- $Z_e = Z_r - \bar{Z} - D$; // Calculate Z error
- $[f_{\mathbb{N}}] = Calculatef_{\mathbb{N}}(W, \Phi_k, B_k, Z_e, N_p, N_b, N_{b_0})$; // Run algorithm 8.1
- $\gamma_{sel} = \begin{bmatrix}(U_{max} - \bar{Z}_u - D_u)_{sel} \\ (\bar{Z}_u + D_u - U_{min})_{sel} \\ (X_{max} - \bar{Z}_x - D_x)_{sel} \\ (\bar{Z}_x + D_x - X_{min})_{sel}\end{bmatrix}$; // Form γ_{sel} vector using algorithm 5.4
- $[\delta\hat{\mathbb{C}}^*, \bar{\lambda}_{sel}] = QPSolve(E_{\mathbb{N}}, f_{\mathbb{N}}, (M_{\mathbb{N}})_{sel}, \gamma_{sel}, \bar{\lambda}_{sel})$; // Solve the Quadratic Program
- $[\delta\tilde{Z}] = Decompress\_And\_Expand(\Phi_k, B_k, \delta\hat{\mathbb{C}}^*, N_p, N_b, N_{b_0})$; // Run algorithm 8.6
- $\bar{Z} = \bar{Z} + D + \delta\tilde{Z}$; // Calculate new nominal state
- $\bar{U} = \bar{Z}_u$; // Extract $\bar{U}$ from $\bar{Z}$

end

Result: $\bar{Z}, \bar{U}, \bar{\lambda}_{sel}$

8.2.4 Generic Computations

In order to evaluate how well the proposed algorithms perform, we performed a generic computation comparison of the main algorithms, namely: algorithm 8.4 for performing the proposed Time-Varying DARE; algorithm 8.5 for calculating $H_{\mathbb{N}}$; algorithm 8.2 for calculating $E_{\mathbb{N}}$; algorithm 8.1 for calculating $f_{\mathbb{N}}$; and algorithm 8.6 for performing the simultaneous decompression/expansions step. One of the performance measurements of interest was the "gain factor" (α) discussed in the generic computations section of the Shifting Strategy (section 5.2.6) which represent the comparison of the algorithms for different block sizes to the "standard" ($N_b = 1$). On the other hand, we are also interested in comparing how well the proposed approach performs when compared to that of the respective counterpart algorithms of the Shifting Strategy of chapter 5, namely those of table 5.5, as well as their respective counterparts of the Dual Mode approach of chapter 6, namely those of table 6.1.

To perform this comparison, we selected the Inverted Pendulum, introduced previously in case study 5.5, which has $n_x = 4$ states and $n_u = 1$ inputs. The algorithm's performance was evaluated on this system using block sizes of $N_b = [1 \rightarrow 6]$ with Ideal Prediction Horizons $N_p = [121, 241]$. The algorithms were then programmed using automatically generated C++ codes for each of the cases based on the Eigen 3 library, and were tested in Ubuntu 20.04 running with Real-Time priority (ie. chrt -r 99 ./main) on a laptop with an Intel i7-8750 CPU overclocked @ 3.9 GHz, and 32 GB DDR4 RAM @ 2,666 MHz, with 120,000 runs per algorithm. The test C++ codes were compiled using the (-O3) optimisation C-flag, as well as with the fused-multiply-addition operations (-mfma) and auto-vectorisation (-mavx) flags enabled to use the Advanced Vector Instruction set available in the Intel CPU. The results of this comparison are gathered in table 8.3 where the minimum computation time obtained for each algorithm is reported, indicating the minimum time that could be achieved if a Real-Time OS would be used.

Case ($n_z = 5$)	$N_p = 121$						$N_p = 241$					
N_b	1	2	3	4	5	6	1	2	3	4	5	6
DARE (alg. 8.4)	10	$9_{1.1}$	$7_{1.4}$	$7_{1.4}$	$6_{1.7}$	$6_{1.7}$	18	$15_{1.2}$	$13_{1.4}$	$13_{1.4}$	$13_{1.4}$	$12_{1.5}$
$H_{\mathbb{N}}$ (alg. 8.5)	24	12_{2}	8_{3}	$7_{3.4}$	$5_{4.8}$	$5_{4.8}$	98	$46_{2.1}$	$30_{3.3}$	$22_{4.5}$	$18_{5.4}$	$15_{6.5}$
$E_{\mathbb{N}}$ (alg. 8.2)	42	$18_{2.3}$	$11_{3.8}$	$9_{4.7}$	$8_{5.3}$	7_{6}	162	$71_{2.3}$	$44_{3.7}$	$31_{5.2}$	$25_{6.5}$	$22_{7.4}$
$f_{\mathbb{N}}$ (alg. 8.1)	2	2	2	2	2	2	3	3	3	3	3	3
$H_{\mathbb{N}}\delta\hat{\mathbb{C}}^*$ (alg. 8.6)	2	2	2	2	2	2	3	3	3	3	3	3
Total	80	$43_{1.9}$	$30_{2.7}$	27_{3}	$23_{3.5}$	$22_{3.6}$	284	$138_{2.1}$	$93_{3.1}$	$72_{3.9}$	$62_{4.6}$	$55_{5.2}$

Table 8.3: Generic Computation Times (in μs) of Ideal MWB Closed Loop Dual Mode Approach for a system with $n_x = 4$ states, $n_u = 1$ inputs (augmented state-size of $n_z = 5$) using different block sizes $N_b = [1, 2, 3, 4, 5, 6]$ with Ideal Prediction Horizons $N_p = [121, 241]$. The gain factor is indicated in red.

In this table (table 8.3) each of the cases are signaled in the gray coloured cases, with the computing times being reported beneath for the $N_b = [1 \rightarrow 6]$ block sizes for each of the algorithms (alg. 8.4, 8.5, 8.2, 8.1 and 8.6), and the "Total" cyan coloured rows representing the summation of these algorithms. Moreover, most of the table cells contain a red-coloured under-script indicating the "gain factor" (α) as explained in section 5.2.6. The rows of the linear term ($f_{\mathbb{N}}$) and decompression/expansion step ($H_{\mathbb{N}}\delta\hat{\mathbb{C}}^*$) are not signaled as they are not the main source of computation times.

As in the generic computation comparison of the Shifting Strategy (sec. 5.2.6), we can see from table 8.3 that the gain factor of $H_{\mathbb{N}}$ (alg. 8.5) and $E_{\mathbb{N}}$ (8.2) follow the block-size itself closely, in some cases being up to 7.4 times faster as in $E_{\mathbb{N}}$ of $N_p = 241$, and giving up to 5.2 times faster calculation in the "Total" times for $N_p = 241$,$N_b = 6$. Note that the "Total" times may be affected by the DARE calculation (alg. 8.4), which presents a rather minimum gain factor of up to $1.7 - 1.5$, respectively, although decreasing as expected from the method of calculating feedback gains (with the required inverse - $K_k = (B_k^T P_{k+1} B_k)^{-1} B_k^T P_{k+1} A_k$) only at N_{b_i} OR $k = 0$ as visible in lines $5-8$ of the algorithm. What this indicates is that the backward propagation of the weights $P_k = w_k + \Phi_k^T P_{k+1} \Phi_k$, visible in line 11 of the algorithm, remains the main source of computations. However, this may change if systems with more inputs were used given the required $n_u \times n_u$ inverse for the calculation of K_k.

On the other hand, we are also interested in knowing how does the proposed method compare to that of those introduced earlier in chapters 5 and 6, namely those of tables 5.5 and 6.1. Comparatively speaking, the proposed approach performs the calculation of $H_{\mathbb{N}}$ and $E_{\mathbb{N}}$ much slower than that of the Ideal MWB of table 5.5, with the original version being up to 2.8 times faster for the calculation of $H_{\mathbb{N}}$ with $N_p = 241$, $N_b = 1$ (98 to $35\mu s$), but becomes closer to the performance of the original version as the block size increases, eg. being only 1.7 times faster for the calculation of $E_{\mathbb{N}}$ with $N_p = 241$,$N_b = 6$ (22 to $13\mu s$). Similarly, the algorithms also presented an increase when compared to those of the Dual Mode approach presented in table 6.1. As an example, the calculation of H/F and E matrix (which represent matrix $H_{\mathbb{N}}$ and $E_{\mathbb{N}}$ with $N_b = 1$), was $25\mu s$ and $33\mu s$, respectively, for a Prediction Horizon of $N_p = 150$, whereas the proposed approach resulted in $24\mu s$ and $42\mu s$ for the Ideal Horizon $N_p = 121$, higher already in the case of $E_{\mathbb{N}}$. This comparative disadvantage is quite surprising considering the state was only augmented by one ($n_z = 5$ against $n_x = 4$) but brings an important question of whether it would be possible to develop algorithms that exploit the zeros structures of A_{CL} and A_{OL} more efficiently which could potentially reduce the computations that come from the state augmentation. We will discuss this at the end of the chapter for future work.

To corroborate this results, a secondary platform was used: the Beaglebone Blue, a robotics-oriented embedded platform running a quasi-Real-Time Debian (Linux-based) OS @ 1 GHz with a NEON floating-point accelerator (fpa), specifically designed to perform efficient parallel floats computations. Only the case with Ideal Prediction Horizon $N_p = 121$ was evaluated in this platform. Moreover, given the platform has a 32-bits ARM processor, and the NEON fpa can only use floats for its operation, the algorithms were tested using floats and doubles to evaluate the advantage that could be obtained in this platform from being able to use reduced numeric precision via the proposed blocked prestabilised approach without having numeric conditioning problems. The developed C++ codes were compiled using the (-O3) optimisation C-flag, and the (-mfpu=neon) flag to enable the NEON instructions set and were run using real-time priority (ie. chrt -r 99 ./main) for 1250 iterations per algorithm. The comparison is presented in table 8.4 where the minimum time is reported, and the gain factor of using floats instead of doubles indicated in the pink row.

From this table it can be appreciated that the approach maintains a similar performance to that of table 8.3, reaching up to 8.8 times faster for the calculation of $E_{\mathbb{N}}$ and being up to 5.8 times faster for the "Total" calculation when using doubles, with DARE preserving the relatively low gain factor of up to 1.7. Lastly, note that just by being able to use floats instead of doubles, the approach would be able to compute up to 3.4 times faster in this platform.

Case ($n_z = 5, N_p = 121$)	Floats					
N_b	1	2	3	4	5	6
DARE (alg. 8.4)	390	$318_{1.2}$	$298_{1.3}$	$281_{1.4}$	$274_{1.4}$	$271_{1.4}$
$H_{\mathbb{N}}$ (alg. 8.5)	1592	$757_{2.1}$	$505_{3.2}$	$386_{4.1}$	$315_{5.1}$	$271_{5.9}$
$E_{\mathbb{N}}$ (alg. 8.2)	2138	$995_{2.1}$	$614_{3.5}$	$464_{4.6}$	$375_{5.7}$	$319_{6.7}$
$f_{\mathbb{N}}$ (alg. 8.1)	33	34_{1}	33_{1}	32_{1}	32_{1}	32_{1}
$H_{\mathbb{N}}\delta\hat{\mathbb{C}}^*$ (alg. 8.6)	35	$32_{1.1}$	$31_{1.1}$	$30_{1.2}$	$30_{1.2}$	$29_{1.2}$
Total	4188	2136_{2}	$1481_{2.8}$	$1193_{3.5}$	$1026_{4.1}$	$922_{4.5}$
Case ($n_z = 5, N_p = 121$)	Doubles					
N_b	1	2	3	4	5	6
DARE (alg. 8.4)	1335	$987_{1.4}$	$875_{1.5}$	$848_{1.6}$	$786_{1.7}$	$769_{1.7}$
$H_{\mathbb{N}}$ (alg. 8.5)	5045	$2124_{2.4}$	$1349_{3.7}$	1001_{5}	$805_{6.3}$	$677_{7.5}$
$E_{\mathbb{N}}$ (alg. 8.2)	7477	$2747_{2.7}$	$1718_{4.4}$	$1267_{5.9}$	$1011_{7.4}$	$850_{8.8}$
$f_{\mathbb{N}}$ (alg. 8.1)	93	$80_{1.2}$	$75_{1.2}$	$73_{1.3}$	$74_{1.3}$	$72_{1.3}$
$H_{\mathbb{N}}\delta\hat{\mathbb{C}}^*$ (alg. 8.6)	92	$69_{5.7}$	$65_{5.7}$	$63_{5.7}$	$62_{5.7}$	$62_{5.7}$
Total	14042	$6007_{2.3}$	$4082_{3.4}$	3525_{4}	$2738_{5.1}$	$2430_{5.8}$
Double-Float Gain Factor	3.4	2.8	2.8	3	2.7	2.6

Table 8.4: Generic Computation Times (in μs) of Ideal MWB Closed Loop Dual Mode Approach using Beaglebone Blue for a system with $n_x = 4$ states, $n_u = 1$ inputs (augmented state-size of $n_z = 5$) using different block sizes $N_b = [1, 2, 3, 4, 5, 6]$ with Ideal Prediction Horizon $N_p = [121]$ and different numeric precision (doubles and floats). The individual gain factor is indicated in red.

8.3 Case Study: The Inverted Pendulum

To evaluate the performance of the proposed approach, the Inverted Pendulum problem presented in previous chapters was selected given its unstable dynamics in the upper equilibrium can easily lead to ill-conditioned problems as discussed in chapter 6. As this system has been repeatedly tested, we will keep this case study brief and we will focus only on the relevant/expected properties which ultimately can relate the results of this chapter with that of the case studies presented in chapters 5 and 6.

For this case study, we used the dynamics of the inverted pendulum from case study 6.4 with the same simulation parameters. Moreover, all the simulations were done using an Ideal Prediction Horizon of $N_p = 121$ for different block sizes $N_b = [1 \rightarrow 6]$, which allowed the verification of the relevant properties. Furthermore, the optimisation was selected as in chapter 6 with the augmented state weights $w_{k+i} = diag([1, 1, 10, 10, 0.1])\ \forall i = [1, N_p - 1]$, and the terminal cost selected as $w_{k+N_p} = 10w_{k+1}$ (ie. no infinite horizon terminal cost used). Finally, the same parameters from table 6.4 were used with the same constraints, selected as $-170 \leq u \leq 170$ for the input, and $-0.35 \leq x \leq 0.35$.

8.3.1 Blocked Pre-stabilisation Example

One of the main concepts or "ideas" motivating this chapter's contribution is that of "blocked prestabilisation". To illustrate this, figure 8.1 shows a comparison between the predicted trajectories during the swing up phase with a relatively small input disturbance (visible in the inner axis of the lowest graph of the figure) on the decision variable when using two approaches: the Ideal MWB approach presented in chapter 5, where the numeric conditioning problem was ignored; and the blocked pre-stabilised approach proposed in this chapter. Note that this is the blocked pre-stabilised version of figure 6.2.

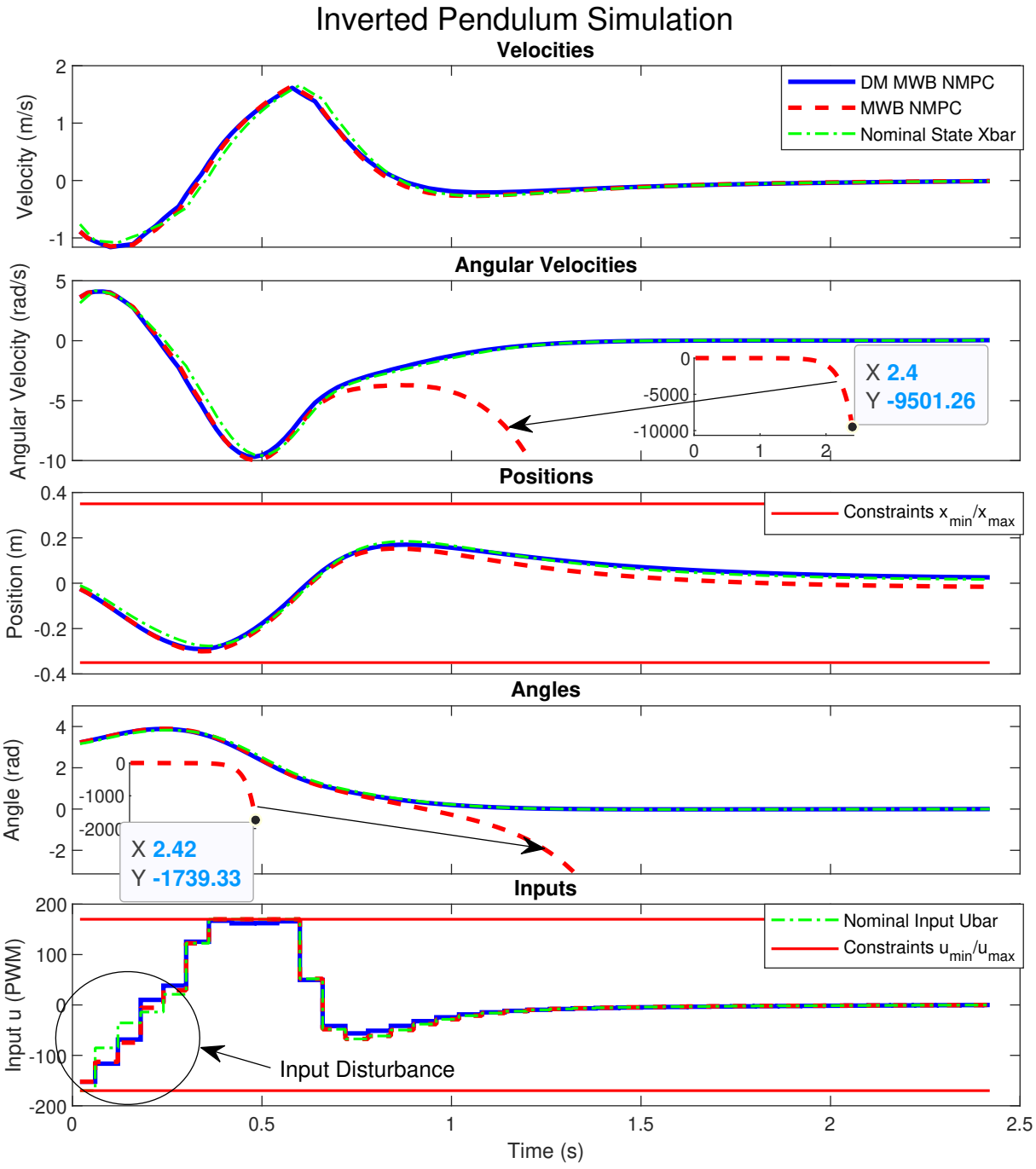


Figure 8.1: Comparison of Blocked Predictions with and without Prestabilisation for the Inverted Pendulum problem during the swing up phase.

As it can be appreciated, the predictions of the proposed approach (blue line) undergo a short transition phase, particularly visible in the upper three graphs before stabilising in the nominal trajectory of the state and input (green dot-dashed-line). In contrast, the predictions of angle and angular velocities (2nd and 4th graphs) when using the original shifting blocks approach (red dashed-line) of chapter 5, can be observed to diverge quite significantly, reaching deviations of up to -9501 and -1739 for the angular velocity and angle, respectively, visible in the inner axis of the respective graphs.

What this demonstrates is that making a relatively small mistake in the blocked decision variable has a significant impact on the predicted trajectories which for a nonlinear optimisation, can affect the linearisation process, as well as the numeric conditioning of the problem in the general case.

8.3.2 Equality of Solutions, Numeric Conditioning and Shifting Constraints

Another set of properties that are expected when applying the proposed method of this chapter are:

1. Equality of the Solutions
2. Lower condition number of the Hessian ($E_{\mathbb{N}}$).
3. Satisfaction of constraints at the selected shooting points.

To illustrate these properties, figure 8.2 shows a 5 second simulation of the system with initial condition $x_0 = \begin{bmatrix} 0 & 0 & 0 & \pi \end{bmatrix}^T$ (ie. the lower equilibrium), using a block size of $N_b = 6$, and starting the optimisation with the free response ($\bar{U} = \mathbb{O}$). The simulation was run using two methods, namely: the proposed blocked closed loop approach of this chapter, depicted by the blue line; and the originally proposed blocking approach from chapter 5, depicted by the red dashed-line.

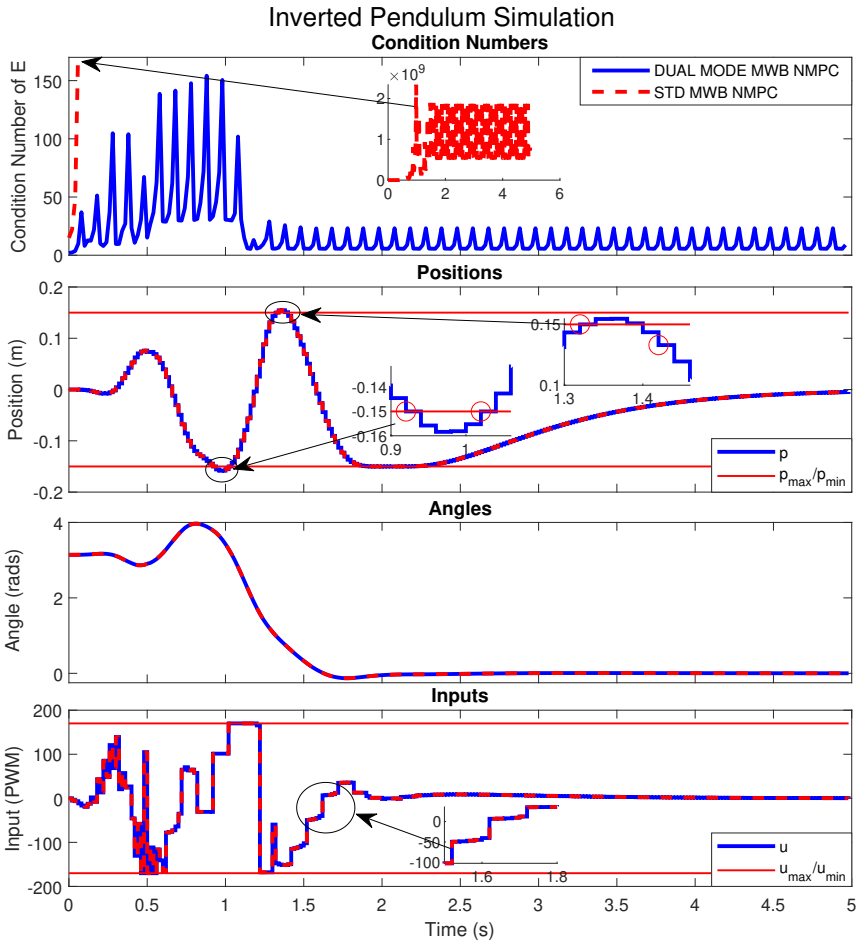


Figure 8.2: Comparison of Blocked Closed Loop DM NMPC with MWB NMPC for the Inverted Pendulum problem with an Ideal Prediction Horizon of $N_p = 121$, a block size of $N_b = 6$, and using the shifting strategy for the constraints. The condition number is given in the top figure for comparison.

The first of these properties can be clearly identified by observing that both solutions are exactly identical, with the embedded blocked solution clearly visible in the lower graph of the figure. Thus, this represents a key illustration of theorem 8.1, which was required to link the nominal stability and recursive feasibility properties of the proposed approach with those presented in chapter 5.

The second of these properties can be clearly seen in the upper graph of the figure where the condition number of $E_{\mathbb{N}}$ is plotted for both solutions: open loop and closed loop. From this, it can be clearly appreciated how the originally proposed blocking approach (red dashed-line) leads to a condition number on the order of 10^9, something which was initially ignored; eventually reaching a steady oscillatory state around the values of $(1.5 - 2) \times 10^9$, even after the system had stabilised. Note that this oscillation on the condition number happens because the embedded blocking structure is shifting forward until resetting its value, which in turn leads to different Hessian's at each time step, each of which has a different condition number. In contrast, the resulting condition number of the proposed approach reaches a maximum value of around 150, and settles its oscillations around $10 - 25$, clearly demonstrating the improved numeric conditioning of the proposed approach.

Finally, regarding the constraint satisfaction of the shooting points, the solution using the original optimisation setup was observed to be more relaxed in the blocked case than the standard NMPC (eg. see figure 8.1), and as such was not even reaching the constraints for some of the evaluated block sizes and initial condition, which in turn didn't allow a proper illustration of the property to be discussed. To address this, the constraints on the positions were restricted further to the range of $-0.15 \leq x \leq 0.15$ as seen in the figure. With this modification in place, the satisfaction of the shooting points can clearly be seen in the two inner graphs of the position graph at times approximately $t_1 = 0.92$ (s) and $t_2 = 1.32$ (s) where the constraint satisfaction is signaled in red circles. Indeed, the separation between the shooting points of the inner graph at t_1 can clearly be seen to be exactly 6, which is the selected block size for the simulation, thus giving a clear example of the constraint satisfaction property of the proposed shifting strategy of chapter 5. Such property remains to be one of the key contributions of this thesis, thus forms an important part to be distinguished for the proposed combined approach.

8.3.3 Computation Times Comparison

In order to evaluate the computational performance of the proposed approach in this system, we developed a set of auto-generated C++ codes based on the Eigen 3 library using the RTI algorithms 8.7 and 8.8 which implemented the approach in the Inverted Pendulum system. In order to reuse the developed auto-generation routines for the tests performed in the Inverted Pendulum case study of sections 5.5 and 6.4, we implemented the same three main modifications indicated in section 6.4.5, namely: different model parameters ($a = -0.3$, $b = 9.8065$, $c = 1$, $f_m = 0$ and $k = 10$), different constraints ($-1 \leq u \leq 1$ and $-1 \leq p \leq 1$) and different weights ($q_{k+i} = diag([0.01, 0.01, 1, 1])$ $\forall i = [1, N_p - 1]$, $q_{k+N_p} = 10q_{k+1}$, $r_{k+i} = 10$ $\forall i = [0, N_p - 1]$), with the rest of the simulation specifications being identical, eg. same sampling time, same initial condition at the lower equilibrium and reference to the upper equilibrium introduced at the end of the prediction horizon. For comparison, the algorithms were developed for different block sizes $N_b = [1 \rightarrow 6]$ with an Ideal Prediction Horizon $N_p = 121$. The solutions of the resulting QPs were obtained using QP OASES [37, 38], all of which were verified to match in all cases from Matlab simulations, to developed C++ codes.

Each of the aforementioned cases was run for 1000 simulations of $T = 10\ (s)$ giving a total of $400,000$ optimisations per case. The codes were run using the same conditions as in the generic computations comparison of section 8.2.4, ie. same laptop running the codes with real-time priority (ie. chrt -r 99 ./main) with the same optimisation flags (-O3,-mavx,-mfma) for the compilation. The resulting average computation times of the constrained iterations of each of these cases is presented in table 8.5. For reference, the resulting gain factors are indicated in the red under-scripts.

N_b	1	2	3	4	5	6
QP OASES	2616	$428_{6.1}$	$180_{14.5}$	$112_{23.4}$	$83_{31.5}$	$69_{37.9}$

Table 8.5: Average constrained computation times for the Inverted Pendulum using different block sizes $N_b = [1 \rightarrow 6]$ with Ideal Prediction Horizon $N_p = 121$. The gain factor is indicated in red.

From this table (table 8.5), it can clearly be appreciated how the proposed approach results in substantial computational gains, being up to 38 times faster than the standard solution ($N_b = 1$) when using block size $N_b = 6$, increasing rapidly as the block-size increases as in the results of table 5.12. Moreover, the proposed approach presents a certain degree of "inefficiency" when comparing the results to the approaches of tables 5.12 or 6.6. However, the approach still presented significant gains, even when compared with the Standard NMPC of table 5.12, and starts to come close to the performance of table 5.12 as the block-size increases, thus demonstrating that the approach results in adequate computational performance and could be used as a viable alternative, especially in the case where reduced numeric conditioning is required to be used with significantly unstable systems.

8.4 Case Study: The Nonlinear Ball-Plate System

To further illustrate the possible applications of this chapter's contribution, this section presents the final case study of this thesis; the non-linear ball-plate system from chapter 6, section 6.3. The system was simulated using the same conditions and parameters, ie. with that of equation (6.40) simulated using Forward Euler with $N_{euler} = 20$ intermediate steps, and a sampling time of $T_s = 30\ (ms)$. The prediction horizon was selected as $N_p = 61$ for it to be an ideal horizon of $N_b = 6$. The optimisation weights were selected as $w_{k+i} = [6, 0.1, 500, 100, 1]\ \forall i = [1, N_p - 1]$, with the terminal weight selected as $w_{k+N_p} = w_{k+1}$, ie. no infinite horizon costing. Finally, the constraints were set as in section 6.3, ie. with $-20 \leq p \leq 20\ (cm)$, and $-10 \leq u \leq 10\ (V)$.

Figure 8.3 shows an example $T = 2.5\ (s)$ simulation of the system where the embedded blocking structure and the condition numbers of the Hessian can clearly be seen in the lower and upper graphs, respectively. As it can be appreciated, the system presents such a highly unstable dynamics that even with the proposed blocked pre-stabilisation, the resulting condition number reaches orders up 2.13×10^6 and presents fluctuations due to the time-varying nature of the nodes at which feedback is applied which ultimately affects the condition number of the Hessian. As mentioned earlier, this system was unable to be solved using the standard blocking method of chapter 5 given it resulted in a Hessian with a condition number in the order of 10^{51}. Thus, this case study provides a relatively simple example of a relatively small system in which the typical input-blocking condensing based methods wouldn't be able to be applied.

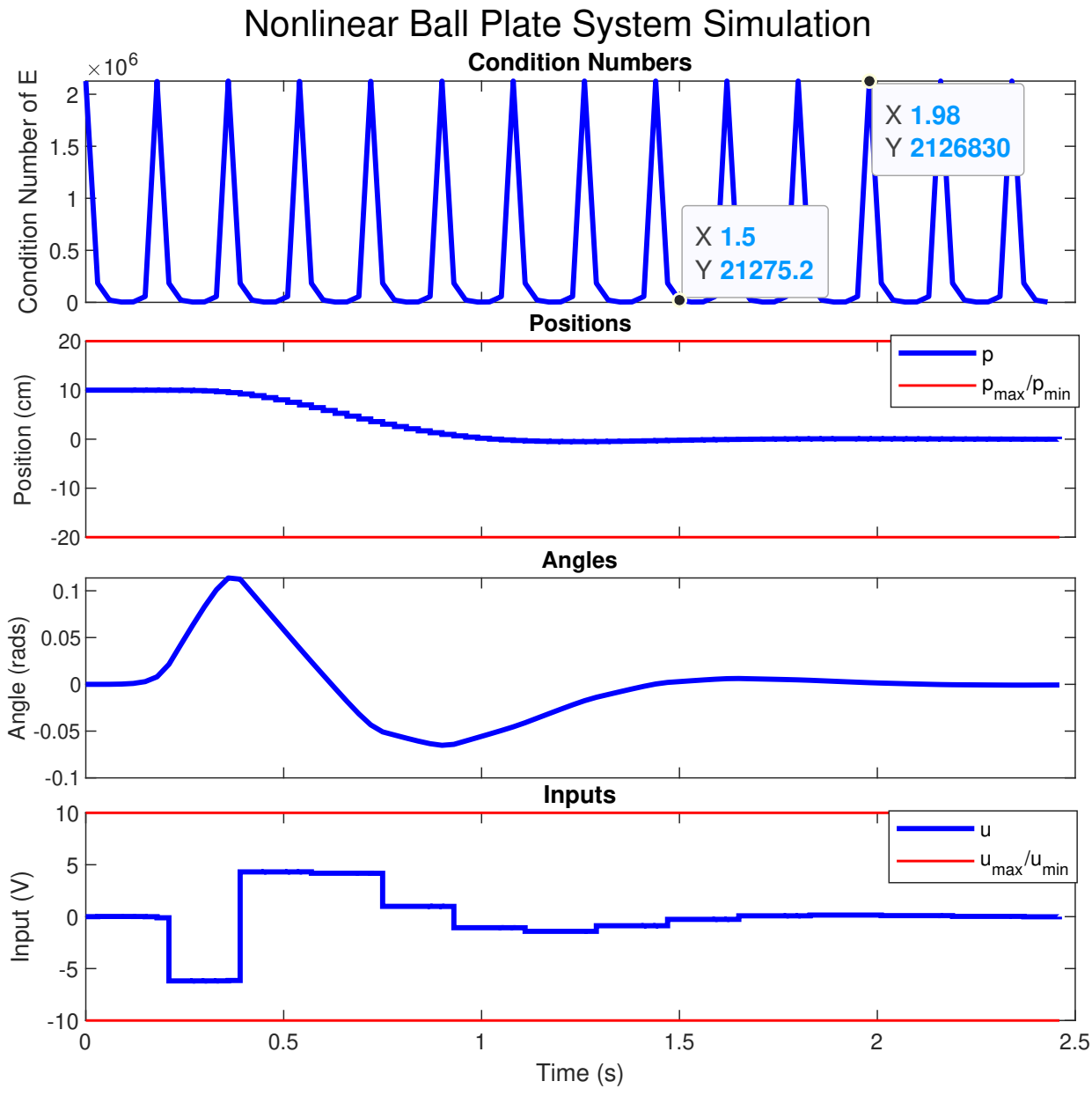


Figure 8.3: Example Response Blocked Closed Loop DM NMPC for the Nonlinear Ball Plate System with a block size of $N_b = 6$, and an Ideal Prediction Horizon of $N_p = 61$. The condition number is given in the top figure for reference.

8.5 Summary

This chapter presented the development of a newly proposed Blocked Closed Loop Dual Mode Nonlinear Model Predictive Control with Shifting Strategies. The proposed approach combined the contributions of chapters 5, 6 and 7 into a single approach which benefits from all their respective advantages and/or disadvantages. The main advantages of the proposed method are:

1. Nominal stability and recursive feasibility guarantees due to the shifting blocks of chapter 5, along with theorem 8.1 and the Dual Mode terminal weight for the blocked infinite horizon solution obtained in chapter 7.

2. Proper Numeric Conditioning for the Hessian due to the closed loop pre-stabilisation, as in chapter 6, allowing reduced precision (ie. floating point) to be used, as well as being able to deal with heavily unstable nonlinear systems.

3. Reduced amount of input and state constraints with recursive feasibility guarantees for the shooting points (selected constraints) in the infinite horizon case, as discussed in chapter 5.

4. Significant overall computational benefits due the embedded blocking structure of chapter 5.

On the other hand, some of the underlying disadvantages are:

1. Slightly higher computation times due to the pre-stabilisation steps (eg. when compared to non-prestabilised solutions from chapter 5)

2. Slightly higher computation times due to the use of an augmented state (eg. when compared to solutions from previous chapters such as chapters 5 and 6).

3. Possibility of violating intermediate constraints as inhereted from the method of chapter 5, which can be compensated using small slacks.

Nonetheless, the overall approach may be critical for the implementation of condensing-based real time NMPC with blocked input parameterisation for heavily unstable nonlinear systems, such as nonlinear ball-plate system presented in section 8.4, or the triple pendulum of chapter 6.

The chapter presented the required steps to be able to combine the other approaches, as well as on the underlying properties to be expected. The proposed approach was successfully applied to the inverted pendulum problem of section 6.4 where all the benefits/properties above were seen and discussed, including significant computational gains of up to 38 times faster than the standard NMPC solution. Furthermore, to give a stronger example of the potential applications of the proposed methodology, the chapter presents a case study for a nonlinear ball-plate system which was unable to be solved using the standard blocking approach of chapter 5. The method was supported by theorem 8.1 (given without formal proof) which allowed the solution to retain the relevant nominal stability and recursive feasibility guarantees of the method introduced in chapter 5. Moreover, the chapter presented a set of algorithms for its efficient implementation in section 8.2, including a further extension to the $O(N)/O(N^2)$ algorithms of chapter 5. In the case where further computational benefits are required, future work will look to take advantage of the special structures presented in matrices A_{OL} and A_{CL} to develop tailored algorithms which could reduce the computational burden further.

Thus, based on the provided evidence, this chapter is considered a key contribution of this thesis, offering a general methodology for implementing blocked closed loop dual mode nonlinear model predictive control with shifting constraints which was observed to result in enough benefits for it to be relevant for its implementation in real systems.

Chapter 9

Summary, Conclusions and Future Work

This thesis presented a set of novel methods for Nonlinear Model Predictive Control (NMPC) tackling a fundamental problem; the reduction of the computation burden. These methods were developed in the hopes of providing the user with a wide range of suitable options for obtaining efficient real-time solutions of NMPC that could serve as alternatives for implementation of NMPC in low-cost/low-performance embedded real-time control systems.

The thesis starts by introducing the general notation, algorithms and methodologies to be used in chapter 3, developed as a "detailed" literature review in the hopes of providing a complete set of methods that could serve as an entry point for this thesis as well as for the new student of NMPC.

Afterwards, we introduce the general input-parameterised frameworks in chapter 4 where a set of algorithms are provided as an initial contribution for the efficient implementation of input-parameterised NMPC using the Real-Time Iteration (RTI) Scheme. The chapter includes 3 case studies which were used to demonstrate the key properties, advantages and disadvantages of these type of methods.

Subsequently, we introduce the proposed Shifting Strategy in chapter 5; a key contribution of this thesis that can be used in a wide range of variations and/or alternatives for obtaining significantly faster solutions when compared to the standard NMPC whilst preserving nominal stability and recursive feasibility properties as proved by theorem 5.3. The chapter provides a set of algorithms and methods for its implementation under the RTI framework, including an extension of the $O(N)/O(N^2)$ algorithms from Ph.D. thesis [8] which proved to be absolutely critical for obtaining efficient real-time solutions and therefore is considered an important contribution. Moreover, the chapter included 4 case studies which demonstrate very clearly the key properties, advantages and disadvantages of the proposed method. One of the key results obtained in these case studies was in the computation times of the popular Inverted Pendulum system where solutions up to 54 times faster than the standard approach (including those of the ACADO toolkit) were obtained, thus fulfilling one of the initial goals set from the initial motivations, aims and objectives of this thesis.

Following this, chapter 6 introduces a novel "generic" dual-mode pre-stabilisation method which allows the optimisation of significantly unstable nonlinear systems that were otherwise unable to be optimised using standard condensing-based methodologies. Although this method does not particularly tackle the computational burden given it results in slightly higher computation times, it was developed as a "pre-requisite" step towards "The combined approach" presented later in chapter 8 where the computational burden was observed to be significantly reduced. Nonetheless, the method is a key

development of this thesis given it solves the "numeric conditioning" problem of condensing-based optimisations as described in the chapter, thus allowing the solution of a wide range of unstable systems with the advantage of being able to use reduced numeric precision (eg. floats instead of doubles) which overall could ultimately result in faster solutions. Moreover, the chapter included a set of algorithms for its implementation under the RTI Scheme, along with an extension to the $O(N)/O(N^2)$ algorithms from Ph.D. thesis [8] which is considered a key contribution of this chapter. The resulting algorithms and methods were tested in 4 case studies which clearly demonstrated the expected advantages and disadvantages of the proposed approach, along with important properties that supported the method such as those described by novel theorem 6.2. Among the key results, it was seen that the proposed method could result in increments in the computation times as low as +3% in the context of Interior Point methods, which were considered to be justified based on the substantial numeric conditioning advantages that were observed which ultimately allowed the solution of highly unstable systems such as the Triple Inverted Pendulum of case study 6.5 that were unable to be solved using the standard methodologies altogether. Finally, the method could be merged with dense input-parameterisation techniques such as the Laguerre polynomials of chapter 4 as discussed in the chapter, which ultimately would offer additional alternatives to the user.

Later, chapter 7 provided a brief and relatively simple solution to the infinite horizon problem that results from the Shifting Strategy of chapter 5, particularly from the implementation of the "Ideal MWB" input parameterisation defined by equation (5.11). The proposed solution could be used as a "terminal weight" for the optimisation, thus allowing the user to obtain a more relaxed stability guarantee. The application of this approach could ultimately allow the use of reduced prediction horizons which would result in faster computation times. Moreover, the chapter introduces the concept of "blocked feedback" gains, as depicted in figure 7.1, which could be used to obtain "blocked pre-stabilised" prediction models, eg. in the context of linear MPC, to acquire Maximum Admissible Sets (or more likely a set of MAS) which could be used as terminal conditions to obtain a more rigorous recursive feasibility guarantee for the approach of chapter 5. This task was left as future work.

Finally, chapter 8 presents the final contribution of this thesis; "The Combined Approach", which merges the methodologies of chapters 4, 5, 6 and 7 into a single strategy which could be used to optimise a wide range of unstable nonlinear systems whilst having a significantly reduced computational burden. The proposed approach is provided with a set of algorithms which allow its efficient implementation, including a further extension to the "already extended" $O(N)/O(N^2)$ algorithms from chapter 5, thus considered as an additional key contribution of this thesis. The chapter includes 2 different case studies which allow the demonstration of the various relevant properties inherited from the methodologies of the other chapters. One of the key results obtained from this case studies were the computation times obtained for the Inverted Pendulum of section 8.3 where solutions up to 38 times faster were obtained when compared to the standard approach. This ultimately aligns with the initial motivations of this thesis where the proposed approach could serve as a viable alternative for handling a wide range of unstable nonlinear systems with the advantage of having a significantly reduced computational burden along with desirable nominal stability and recursive feasibility properties. In the specific case where faster computation times are required, the proposed algorithms could take advantage of the special structures of some of the used matrices of the approach which could reduce the computational burden further. This task was left for future work.

In conclusion, we believe the work presented in this Ph.D. to be of great importance and relevance for the NMPC research community, both in academia and industry, and that its proper understanding could be key for enabling a wide range of applications in which real-time performance is required and hard deterministic time-constraints must be satisfied. Indeed, the author of this thesis strongly believes that even though some of the methods and algorithms may seem complex, they might be easier to implement and understand than they appear which makes them relevant for industry engineers which typically prefer avoiding too complex or theoretical procedures, often lacking detailed steps for their implementation. Moreover, the developed approaches satisfied the initial objectives of this Ph.D. which were to "tackle the computational burden whilst preserving desirable properties for the optimisation", and we believe the methods described in this thesis provided enough details and supporting evidence for them to serve as viable alternatives to the standard NMPC approaches for practical applications.

On the other hand, future work will look to merge the proposed approaches with other existing alternatives to give an ever wider range of options to the final user. As an example, one direct extension currently in the works will be to extend the proposed Closed Loop Dual Mode NMPC framework to couple it with the input parameterised approach where by calculating and embedding the unconstrained solution in the "free-response" (ie. with the decision variables set to zero), the optimisation can then use the Laguerre Polynomials, or any relevant input parameterisation to adjust the solution only when the unconstrained solution violates the constrains, similar to the work presented in [79] for Linear MPC. Similarly, the proposed setup will be considered in a robust online framework supported by multi-model unstable systems to capture the uncertainty whilst having a set "pre-stabilising" gains to "pre-condition" the resulting optimisation and improve its hot-starting properties, resulting in an online robust optimisation which again aligned with the initial motivations and objectives of this thesis. Lastly, the algorithms for chapter 8 will be adjusted to exploit the underlying structure of the operations, thus leading to a more efficient framework, and the algorithms required the calculate the MAS for the proposed Shifting Strategy case with the Ideal MWB will be developed based on the blocked feedback gains obtained in chapter 7.

Thank you for your time and effort reading through my Ph.D. thesis!

Bibliography

[1] M.A. Abbas, R. Milman, and J.M. Eklund. Obstacle avoidance in real time with Nonlinear Model Predictive Control of autonomous vehicles. *Canadian Journal of Electrical and Computer Engineering*, 40(1):1, 2017.

[2] M. Abdullah and J. Anthony Rossiter. A Formal Sensitivity Analysis for Laguerre Based Predictive Functional Control. *UKACC 12th International Conference on Control*, pages 20–25, 2018.

[3] M. Abdullah and J.A. Rossiter. Alternative Method for Predictive Functional Control to Handle an Integrating Process. *UKACC 12th International Conference on Control*, pages 26–31, 2018.

[4] M. Abdullah and J.A. Rossiter. Using Laguerre functions to improve the tuning and performance of predictive functional control. *International Journal of Control*, 94(1):202–214, 2021.

[5] M. Alamir. Fast NMPC: A reality-steered paradigm: Key properties of fast NMPC algorithms. *European Control Conference, ECC*, (4):2472–2477, 2014.

[6] M. Alamir and A. Murilo. Swing-up and stabilization of a Twin-Pendulum under state and control constraints by a fast NMPC scheme. *Automatica*, 44(5):1319–1324, 2008.

[7] A. Alessio and A. Bemporad. *A Survey on Explicit Model Predictive Control*, pages 345–369. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[8] J. Andersson. *A General-Purpose Software Framework for Dynamic Optimization*. Ph.d. thesis, KU Leuven, 2013.

[9] A. Arpornwichanop and P. Kittisupakorn. Dual mode NMPC for regulating the concentration of exothermic reactor under parametric uncertainties. *Journal of Chemical Engineering of Japan*, 37(6):698–710, 2004.

[10] C. Auger, A. Merigaud, and J.V. Ringwood. Receding-Horizon Pseudo-spectral Control of Wave Energy Converters Using Periodic Basis Functions. *IEEE Transactions on Sustainable Energy*, 10(4):1644–1652, 2019.

[11] L. Balbis. *Nonlinear Model Predictive Control for Industrial Applications*. Ph.d. thesis, University of Strathclyde, 2007.

[12] L. Balbis, R. Katebi, R. Dunia, A. Ordys, and M.J. Grimble. Nonlinear predictive control for real time applications. *IEEE International Conference on Control Applications*, (1):211–216, 2006.

[13] Beagleboard.org. Beaglebone Blue, 2017. URL https://beagleboard.org/blue.

[14] A. Bemporad and M. Morari. Robust model predictive control: A survey. In A. Garulli and A. Tesi, editors, *Robustness in identification and control*, pages 207–226, London, 1999.

[15] J. Benoit. Eigen, 2020. URL http://eigen.tuxfamily.org/index.php?title=Main_Page.

[16] L.T. Biegler. An overview of simultaneous strategies for dynamic optimization. *Chemical Engineering and Processing: Process Intensification*, 46(11):1043–1053, 2007.

[17] A. Budiyanto, A. Cahyadi, T.B. Adji, O. Wahyunggoro, and J. Grafika. UAV Obstacle Avoidance Using Potential Field under Dynamic Environment. *International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC)*, pages 187–192, 2015.

[18] R. Cagienard, P. Grieder, E.C. Kerrigan, and M. Morari. Move blocking strategies in receding horizon control. *Journal of Process Control*, 17(6):563–570, 2007.

[19] E.F. Camacho. *Model predictive control*. Advanced textbooks in control and signal processing. Springer, London ; New York, 2nd ed. edition, 2003.

[20] P.J. Campo and M. Morari. Robust Model Predictive Control. *Proceedings of the American Control Conference*, pages 1021–1026, 1987.

[21] Y. Chen, D. Cuccato, M. Bruschetta, and A. Beghi. An Inexact Sensitivity Updating Scheme for Fast Nonlinear Model Predictive Control based on a Curvature-like Measure of Nonlinearity. *Proceedings of the IEEE Conference on Decision and Control*, pages 4382–4387, 2017.

[22] Z. Cheng, D. Necsulescu, and B. Kim. Model predictive control and dynamic inversion for unmanned aerial vehicle. *IFAC Proceedings Volumes*, 37(8):645–650, 2004.

[23] P.S.G. Cisneros, S. Voss, and H. Werner. Efficient Nonlinear Model Predictive Control via quasi-LPV representation. *IEEE 55th Conference on Decision and Control (CDC)*, pages 3216–3221, 2016.

[24] K. Dalamagkidis, K.P. Valavanis, and L.A. Piegl. Small Unmanned Helicopter Autorotation using Non-linear Model Predictive Control. *49th IEEE Conference on Decision and Control*, pages 6350–6357, 2010.

[25] F. Debrouwere, M. Vukov, R. Quirynen, M. Diehl, and J. Swevers. *Experimental validation of combined nonlinear optimal control and estimation of an overhead crane*, volume 19. IFAC, 2014.

[26] A.P. Deshpande, S.C. Patwardhan, and S.S. Narasimhan. Intelligent state estimation for fault tolerant nonlinear predictive control. *Journal of Process Control*, 19(2):187–204, 2009.

[27] G. Di Francesco and M. Mattei. Modeling and Incremental Nonlinear Dynamic Inversion Control of a Novel Unmanned Tiltrotor. *Journal of Aircraft*, 53(1):73–86, 2016.

[28] M. Diehl and S. Gros. *Numerical Optimisation of Dynamic Systems (Draft)*. Syscop, University of Freiburg, 2016.

[29] M. Diehl, L. Magni, and G. De Nicolao. Online NMPC of a looping kite using approximate infinite horizon closed loop costing. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 36(18): 519–524, 2003.

[30] M. Diehl, L. Magni, and G. De Nicolao. Efficient NMPC of unstable periodic systems using approximate infinite horizon closed loop costing. *Annual Reviews in Control*, 28(1):37–45, 2004.

[31] M. Diehl, H.G. Bock, and J.P. Schlöder. A Real-Time Iteration Scheme for Nonlinear Optimization in Optimal Feedback Control. *SIAM Journal on Control and Optimization*, 43(5):1714–1736, 2005.

[32] M. Diehl, R. Findeisen, F. Allgöwer, H.G. Bock, and J.P. Schlöder. Nominal stability of real-time iteration scheme for nonlinear model predictive control. *IEE Proceedings-Control Theory and Applications*, 152(3):296–308, 2005.

[33] J. El Hadef, S. Olaru, P. Rodriguez-Ayerbe, G. Colin, Y. Chamaillard, and V. Talon. Nonlinear model predictive control of the air path of a turbocharged gasoline engine using Laguerre functions. *17th International Conference on System Theory, Control and Computing (ICSTCC)*, pages 193–200, 2013.

[34] E. Fabregas, J. Chacón, S. Dormido-Canto, G. Farias, and S. Dormido. Virtual Laboratory of the Ball and Plate System. *IFAC-PapersOnLine*, 48(29):152–157, 2015.

[35] M. Faessler, A. Franchi, and D. Scaramuzza. Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories. *IEEE Robotics and Automation Letters*, 2018.

[36] Y. Fang and A. Armaou. A Formulation of Advanced-step Bilinear Carleman Approximation-based Nonlinear Model Predictive Control. *IEEE Conference on Decision and Control*, 2016.

[37] H.J. Ferreau, H. Bock, and M. Diehl. An online active set strategy to overcome the limitations of Explicit MPC. *International Journal of Robust and Nonlinear Control*, 18:816–830, 2008.

[38] H.J. Ferreau, C. Kirches, A. Potschka, H.G. Bock, and M. Diehl. qpOASES: a parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4): 327–363, 2014.

[39] A. Forsgren and P.E. Gill. Primal-dual interior methods for nonconvex nonlinear programming. *SIAM Journal on Optimization*, 8(4):1132–1152, 1998.

[40] J.V. Frasch, S. Sager, and M. Diehl. A parallel quadratic programming method for dynamic optimization problems. *Mathematical Programming Computation*, 7(3):289–329, 2015.

[41] D. Garcia-Violini and J.V. Ringwood. Energy maximising robust control for spectral and pseudospectral methods with application to wave energy systems. *International Journal of Control*, 0(0):1–12, 2019.

[42] G. Garimella, M. Sheckells, and M. Kobilarov. Robust obstacle avoidance for aerial platforms using adaptive model predictive control. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 5876–5882, 2017.

[43] R. Genest and J.V. Ringwood. A critical comparison of model-predictive and pseudospectral control for wave energy devices. *Journal of Ocean Engineering and Marine Energy*, 2(4):485–499, 2016.

[44] R. Genest and J.V. Ringwood. Receding Horizon Pseudospectral Control for Energy Maximization with Application to Wave Energy Devices. *IEEE Transactions on Control Systems Technology*, 25(1):29–38, 2017.

[45] T. Glück, A. Eder, and A. Kugi. Swing-up control of a triple pendulum on a cart with experimental validation. *Automatica*, 49(3):801–808, 2013.

[46] R. Gondhalekar and J.i. Imura. Least-restrictive move-blocking model predictive control. *Automatica*, 46(7):1234–1240, 2010.

[47] O. Gonzalez and A. Rossiter. Fast hybrid dual mode NMPC for a parallel double inverted pendulum with experimental validation. *IET Control Theory & Applications*, 2020.

[48] O.J. Gonzalez Villarreal and J.A. Rossiter. A Time-Varying Shifting Strategy for Block Based MPC Solutions using a RTI Scheme. *IFAC NMPC Conference Paper*, 2018.

[49] O.J. Gonzalez Villarreal, J.A. Rossiter, and H. Shin. Laguerre-Based Adaptive MPC for Attitude Stabilization of Quad-Rotor. *UKACC 12th International Conference on Control*, pages 360–365, 2018.

[50] O.J. Gonzalez Villarreal and A. Rossiter. Shifting strategy for efficient block-based non-linear model predictive control using real-time iterations. *IET Control Theory and Applications*, 14(6): 865–877, 2020.

[51] A. Govindarajan. *Accelerating Convergence of Leapfrogging Optimization - Applications to Nonlinear Process Modeling and Nonlinear Model Predictive Control.* Ph.d. thesis, Oklahoma State University, 2014.

[52] A. Grancharova, J. Kocijanb, and T.A. Johansend. Explicit output-feedback nonlinear predictive control based on black-box models. *Engineering Applications of Artificial Intelligence*, 24(2):388–397, 2011.

[53] A. Grancharova, T.A. Johansen, and V. Petrova. Distributed Nonlinear Model Predictive Control by Sequential Linearization and Accelerated Gradient Method. *IFAC-PapersOnLine*, 49(7):597–602, 2016.

[54] S. Gros, M. Vukov, and M. Diehl. A Real-time MHE and NMPC Scheme for the Control of Multi-Mega Watts Wind Turbines. *Conference on Decision and Control*, pages 1007–1012, 2013.

[55] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl. From linear to nonlinear MPC: bridging the gap via the real-time iteration. *International Journal of Control*, pages 1–19, 2016.

[56] S. Gros. An economic NMPC formulation for Wind Turbine Control. *Proceedings of the IEEE Conference on Decision and Control*, pages 1001–1006, 2013.

[57] S. Gros, R. Quirynen, and M. Diehl. Aircraft control based on fast non-linear MPC & multiple-shooting. *Proceedings of the IEEE Conference on Decision and Control*, (1):1142–1147, 2012.

[58] L. Grune, J. Pannek, M. Seehafer, and K. Worthmann. Analysis of unconstrained nonlinear MPC schemes with time varying control horizon. *Proceedings of the IEEE Conference on Decision and Control*, (0):2605–2610, 2012.

[59] J. Guerrero Fernandez, O.J. Gonzalez Villarreal, and J.A. Rossiter. Model Predictive Control for Wave Energy Converters: A Moving Window Blocking Approach. *IFAC World Conference Paper*, 2020.

[60] J. Hanema, R. Tóth, and M. Lazar. Stabilizing Non-linear MPC Using Linear Parameter-Varying Representations. *IEEE 56th Annual Conference on Decision and Control*, pages 0–5, 2017.

[61] D. He, L. Wang, and L. Yu. Multi-objective nonlinear predictive control of process systems: A dual-mode tracking control approach. *Journal of Process Control*, 25:142–151, 2015.

[62] P.H. Heins, B.L. Jones, and D.J. Taunton. Design and validation of an unmanned surface vehicle simulation model. *Applied Mathematical Modelling*, 48:749–774, 2017.

[63] D.R. Herber and J.T. Allison. Wave energy extraction maximization in irregular ocean waves using pseudospectral methods. *Proceedings of the ASME Design Engineering Technical Conference*, 2013.

[64] B. Houska, H.J. Ferreau, and M. Diehl. ACADO toolkit-An open-source framework for automatic control and dynamic optimization. *Optimal Control Applications and Methods*, 32(3):298–312, 2011.

[65] B. Houska, H.J. Ferreau, and M. Diehl. An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range. *Automatica*, 47(10):2279–2285, 2011.

[66] B. Houska, H.J. Ferreau, and M. Diehl. Autogenerating microsecond solvers for nonlinear MPC: A tutorial using ACADO integrators. *Optimal Control Applications and Methods*, 36:685–704, 2015.

[67] R. Huang, L.T. Biegler, and S.C. Patwardhan. Fast Offset-Free Nonlinear Model Predictive Control Based on Moving Horizon Estimation. *Industrial & Engineering Chemistry Research*, 49 (17):7882–7890, 2010.

[68] R. Huang, S.C. Patwardhan, and L.T. Biegler. *Adaptive quasi-infinite horizon NMPC of a continuous fermenter*, volume 9. IFAC, 2010.

[69] J. Huusom, N. Poulsen, S. Jorgensen, and J. Jorgensen. Tuning of methods for offset free MPC based on ARX model representations. *American Control Conference (ACC)*, pages 2355–2360, 2010.

[70] S. Iplikci. Runge–Kutta model-based adaptive predictive control mechanism for non-linear processes. *Transactions of the Institute of Measurement and Control*, 35(2):166–180, 2013.

[71] A.M.Z. Jasour and M. Farrokhi. Adaptive Neuro-NMPC Control of Redundant Robotic Manipulators for Path Tracking and Obstacle Avoidance. *Proceedings of the European Control Conference*, 2009.

[72] T.A. Johansen and T.I. Fossen. Automatica Control allocation — A survey. *Automatica*, 49: 1087–1103, 2013.

[73] J. Kalmari, J. Backman, and A. Visala. A toolkit for nonlinear model predictive control using gradient projection and code generation. *Control Engineering Practice*, 39:56–66, 2015.

[74] M. Kamel, K. Alexis, M. Achtelik, and R. Siegwart. Fast nonlinear model predictive control for multicopter attitude tracking on SO(3). *IEEE Conference on Control and Applications, CCA*, (3):1160–1166, 2015.

[75] S.M. Kargar, K. Salahshoor, and M.J. Yazdanpanah. Integrated nonlinear model predictive fault tolerant control and multiple model based fault detection and diagnosis. *Chemical Engineering Research and Design*, 92(2):340–349, 2014.

[76] S.M. Kargar, K. Salahshoor, and M.J. Yazdanpanah. Multiple Model-Based Fault Detection and Diagnosis for Nonlinear Model Predictive Fault-Tolerant Control. *Arabian Journal for Science and Engineering*, 39(10):7433–7442, 2014.

[77] M. Kelly. An introduction to trajectory optimization: How to do your own direct collocation . *SIAM Review*, 59(4):849–904, 2017.

[78] B. Khan and J.A. Rossiter. Alternative parameterisation within predictive control: A systematic selection. *International Journal of Control*, 86(8):1397–1409, 2013.

[79] B. Khan. *Efficient Parameterise solutions of predictive control.* Ph.d. thesis, The University of Sheffield, 2013.

[80] H.S. Khan and M.B. Kadri. Attitude and Altitude Control of Quadrotor by Discrete PID control and Non-linear Model Predictive Control. *International Conference on Information and Communication Technologies (ICICT)*, 2015.

[81] R. Khan, P. Williams, R. Hill, and C. Bil. Fault tolerant flight control system design for UAV's using Nonlinear Model Predictive Control. *Australian Control Conference*, pages 297–302, 2011.

[82] R. Khan, P. Williams, P. Riseborough, A. Rao, and R. Hill. Designing a Nonlinear Model Predictive Controller for Fault Tolerant Flight Control. *arXiv:1609.01529*, 1:1–29, 2016.

[83] C. Kirches, L. Wirsching, H.G. Bock, and J.P. Schlöder. Efficient direct multiple shooting for nonlinear model predictive control on long horizons. *Journal of Process Control*, 22(3):540–550, 2012.

[84] D. Kouzoupis, H.J. Ferreau, H. Peyrl, and M. Diehl. First-order methods in embedded nonlinear model predictive control. *European Control Conference, ECC*, pages 2617–2622, 2015.

[85] D. Kouzoupis, R. Quirynen, J.V. Frasch, and M. Diehl. Block Condensing for Fast Nonlinear MPC with the Dual Newton Strategy. *IFAC-PapersOnLine*, 48(23):26–31, 2015.

[86] D. Kouzoupis, R. Quirynen, B. Houska, and M. Diehl. A block based ALADIN scheme for highly parallelizable direct Optimal Control. *Proceedings of the American Control Conference*, pages 1124–1129, 2016.

[87] D. Kufoalor and T. Johansen. Reconfigurable fault tolerant flight control based on Nonlinear Model Predictive Control. *Proceedings of the American Control Conference*, (978):5128–5133, 2013.

[88] K. Kunz, S.M. Huck, and T.H. Summers. Fast Model Predictive Control of Miniature Helicopters. *European Control Conference*, pages 1377–1382, 2013.

[89] D. Lee, H. Lim, and H.J. Kim. Obstacle avoidance using image-based visual servoing integrated with nonlinear model predictive control. *IEEE Conference on Decision and Control and European Control Conference*, pages 5689–5694, 2011.

[90] G. Li and M.R. Belmont. Model predictive control of a sea wave energy converter: A convex approach. In *IFAC Proceedings Volumes (IFAC-PapersOnline)*, volume 19, pages 11987–11992. 2014.

[91] J. Li, J. Lv, and J. Jian. A globally and superlinearly convergent primal-dual interior point method for general constrained optimization. *Numerical Mathematics*, 8(3):313–335, 2015.

[92] J.A.J. Ligthart, P. Poksawat, and L. Wang. Experimentally Validated Model Predictive Controller for a Hexacopter. *IFAC-PapersOnLine*, 50(1):4137–4142, 2017.

[93] C. Liu, W.H. Chen, and J. Andrews. Explicit non-linear model predictive control for autonomous helicopters. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 226(9):1171–1182, 2012.

[94] J. Löfberg. Oops! i cannot do it again: Testing for recursive feasibility in MPC. *Automatica*, 48 (3):550–555, 2012.

[95] R. Lopez-Negrete, F.J. D'Amato, L.T. Biegler, and A. Kumar. Fast nonlinear model predictive control: Formulation and industrial process applications. *Computers & Chemical Engineering*, 51:55–64, 2013.

[96] J. Lu, R. Wei, X. Wang, and Z. Wang. Backstepping control of discrete-time chaotic systems with application to the Henon system. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 48(11):1359–1363, 2001.

[97] T. Madani and A. Benallegue. Backstepping Control for a Quadrotor Helicopter. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3255–3260, 2006.

[98] S.H. Mathisen, T.I. Fossen, and T.A. Johansen. Non-linear model predictive control for guidance of a fixed-wing UAV in precision deep stall landing. *International Conference on Unmanned Aircraft Systems, ICUAS*, pages 356–365, 2015.

[99] J. Mattingley, Y. Wang, and S. Boyd. Code generation for receding horizon control. In *IEEE International Symposium on Computer-Aided Control System Design*, pages 985–992. 2010.

[100] D.R. McCullough, O.J. Gonzalez V., B.L. Jones, and J.A. Rossiter. Towards Control of Autonomous Surface Vehicles in Rough Seas. *IFAC World Conference*, 2020.

[101] Q. Mei, F. Xu, H. Chen, Z. Li, and Y. Hu. Fast Model Predictive Control Based on Multiscale System Theory. *12th World Congress on Intelligent Control and Automation (WCICA)*, pages 2517–2522, 2016.

[102] H. Merabti, I. Bouchachi, and K. Belarbi. Nonlinear Model Predictive Control of a Quadcopter. *International Conference on Sciences and Techniques of Automatic Control and Computer Engineering*, 16(1):41–50, 2015.

[103] A. Mérigaud and J.V. Ringwood. Towards realistic non-linear receding-horizon spectral control of wave energy converters. *Control Engineering Practice*, 81:145–161, 2018.

[104] a. Mills, A. Wills, and B. Ninness. Nonlinear model predictive control of an inverted pendulum. *American Control Conference*, pages 2335–2340, 2009.

[105] M.A. Naidoo, R. Padhi, and I.K. Craig. A New Nonlinear Suboptimal Control Design Approach for Milling Circuits. *In Proceedings of IFAC World Congress*, 47:9804–9809, 2014.

[106] T.P. Nascimento, A.G. Conceiçaõ, and A.P. Moreira. Multi-Robot nonlinear model predictive formation control: The obstacle avoidance problem. *Robotica*, 34(3):549–567, 2016.

[107] M. Neunert, C. De Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli. Fast nonlinear Model Predictive Control for unified trajectory optimization and tracking. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1398–1404, 2016.

[108] A. Nurkanović, A. Zanelli, S. Albrecht, and M. Diehl. The advanced step real time iteration for nmpc. In *IEEE 58th Conference on Decision and Control (CDC)*, pages 5298–5305, 2019.

[109] K. Ogata. *Discrete-time control systems*. Prentice Hall, Englewood Cliffs, N.J., 2nd ed. edition, 1995.

[110] C.J. Ong, Z. Wang, and M. Dehghan. Model Predictive Control for Switching Systems With Dwell-Time Restriction. *IEEE Transactions on Automatic Control*, 61(12):4189–4195, 2016.

[111] P. Orlowski. Convergence of the Discrete-Time Nonlinear Model Predictive Control with Successive Time-Varying Linearization along Predicted Trajectories. *Research Journal Elektronika Ir Elektrotechnika*, 2011.

[112] V.G. Palma, A. Suardi, and E.C. Kerrigan. Sensitivity-based multistep MPC for embedded systems. *IFAC-PapersOnLine*, 48(23):360–365, 2015.

[113] J.M. Park, D.W. Kim, Y.S. Yoon, H.J. Kim, and K.S. Yi. Obstacle avoidance of autonomous vehicles based on model predictive control. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 223(12):1499–1516, 2009.

[114] R. Quirynen, S. Gros, and M. Diehl. Efficient NMPC for nonlinear models with linear subsystems. *Proceedings of the IEEE Conference on Decision and Control*, pages 5101–5106, 2013.

[115] R. Quirynen, M. Vukov, and M. Diehl. Multiple Shooting in a Microsecond. In *Multiple Shooting and Time Domain Decomposition Methods*, volume 9, pages 183–202. Springer, Cham, 2015.

[116] C.V. Rao, S.J. Wright, and J.B. Rawlings. Application of interior-point methods to model predictive control. *Journal of optimization theory and applications*, 99(3):723–757, 1998.

[117] J. Rawlings, D. Mayne, and M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2017.

[118] S. Revollar and P. Vega. Economic optimization of wastewater treatment plants using Non Linear Model Predictive Control. *19th International Conference on System Theory, Control and Computing (ICSTCC)*, pages 583–588, 2015.

[119] G. Rigatos and P. Siano. Control of quadrotors using differential flatness theory and the derivative-free nonlinear Kalman filter. *AIP Conference Proceedings*, 1558(2013):2545–2550, 2013.

[120] I. Rivas-Cambero and J.M. Sausedo-Solorio. Dynamics of the shift in resonance frequency in a triple pendulum. *Meccanica*, 47(4):835–844, 2012.

[121] J.A. Rossiter. *Model-based predictive control : a practical approach*. Control series. CRC Press, Boca Raton, 2003.

[122] J.A. Rossiter. *A first course in predictive control*. CRC Press, Taylor & Francis, Boca Raton, 2nd ed. edition, 2018.

[123] J.A. Rossiter, B. Kouvaritakis, and M.J. Rice. A numerically robust state-space approach to stable-predictive control strategies. *Automatica*, 34(1):65–73, 1998.

[124] J.A. Rossiter, J. Sheng, T. Chen, and S.L. Shah. Interpretations of and options in dual-rate predictive control. *Journal of Process Control*, 15(2):135–148, 2005.

[125] J.A. Rossiter, L. Wang, and G. Valencia-Palomo. Efficient algorithms for trading of feasibility and performance in predictive control. *International Journal of Control*, 83(4):789–797, 2010.

[126] S. Sabatelli, M. Galgani, L. Fanucci, and A. Rocchi. A double-stage kalman filter for orientation tracking with an integrated processor in 9-D IMU. *IEEE Transactions on Instrumentation and Measurement*, 62(3):590–598, 2013.

[127] B. Sabetghadam. Fast Nonlinear Model Predictive Control for Teleoperation Systems using Computationally Efficient Optimization Techniques. *22nd Iranian Conference on Electrical Engineering (ICEE)*, pages 1371–1376, 2014.

[128] P. Sarhadi, K. Salahshoor, and A. Khaki-Sedigh. Robustness analysis and tuning of generalized predictive control using frequency domain approaches. *Applied Mathematical Modelling*, 36(12): 6167–6185, 2012.

[129] H. Seki, S. Ooyama, and M. Ogawa. Nonlinear Model Predictive Control Using Sucessive Linearization - Application to Chemical Reactors. *Trans. of the Society of Instrument and Control Engineers*, E-3(1):66–72, 2004.

[130] R.C. Shekhar and C. Manzie. Optimal move blocking strategies for model predictive control. *Automatica*, 61:27–34, 2015.

[131] C. Shen, B. Buckham, and Y. Shi. Modified C/GMRES Algorithm for Fast Nonlinear Model Predictive Tracking Control of AUVs. *IEEE Transactions on Control Systems Technology*, 25(5): 1896–1904, 2017.

[132] H. Shi, C. Su, J. Cao, P. Li, J. Liang, and G. Zhong. Nonlinear adaptive predictive functional control based on the takagi-sugeno model for average cracking outlet temperature of the ethylene cracking furnace. *Industrial and Engineering Chemistry Research*, 54(6):1849–1860, 2015.

[133] M. Short. Move Suppression Calculations for Well-Conditioned MPC. *ISA Transactions*, 67: 371–381, 2017.

[134] S.H. Son, T.H. Oh, J.W. Kim, and J.M. Lee. Move blocked model predictive control with improved optimality using semi-explicit approach for applying time-varying blocking structure. *Journal of Process Control*, 92:50–61, 2020.

[135] J. Sowman, D.S. Laila, and S. Longo. Real-time approximate explicit nonlinear model predictive control for the swing-up of a reaction wheel pendulum. *Proceedings of the IEEE Conference on Decision and Control*, pages 4308–4313, 2015.

[136] S. Subramanian, S. Lucia, and S. Engell. Economic multi-stage output nonlinear model predictive control. *IEEE Conference on Control Applications, CCA*, pages 1837–1842, 2014.

[137] M.H. Tanveer, D. Hazry, S.F. Ahmed, M.K. Joyo, F.A. Warsi, H. Kamaruddin, M. Zuradzman, K. Wan, and A.B. Shahriman. NMPC-PID Based Control Structure Design for Avoiding Uncertainties in Attitude and Altitude Tracking Control of Quadrotor. *IEEE 10th International Colloquium on Signal Processing & its Applications*, pages 7–9, 2014.

[138] J. Thomas. Analytical non-linear model predictive control for hybrid systems with discrete inputs only. *IET Control Theory & Applications*, 6(8):1080–1088, 2012.

[139] C. Tian, J. Wang, Z. Yin, and G. Yu. Integral backstepping based nonlinear control for quadrotor. *Chinese Control Conference, CCC*, pages 10581–10585, 2016.

[140] G. Torrisi, S. Grammatico, D. Frick, T. Robbiani, R.S. Smith, and M. Morari. Low-Complexity First-Order Constraint Linearization Methods for Efficient Nonlinear MPC. *IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 4376–4381, 2017.

[141] L.N. Trefethen. *Spectral methods in MATLAB*. Software, environments, tools. Society for Industrial and Applied Mathematics, Philadelphia, Pa., 2000.

[142] T. Utstumo, T.W. Berge, and J.T. Gravdahl. Non-linear Model Predictive Control for constrained robot navigation in row crops. *IEEE International Conference on Industrial Technology (ICIT)*, (7491):357–362, 2015.

[143] R. Verschueren, G. Frison, D. Kouzoupis, N. van Duijkeren, A. Zanelli, R. Quirynen, and M. Diehl. Towards a modular software package for embedded optimization. *IFAC-PapersOnLine*, 51(20):374–380, 2018.

[144] O.J.G. Villarreal. A Fast Dual Mode NMPC for Parallel Double Inverted Pendulum. Code Ocean, Sept 2019. Available in https://doi.org/10.24433/CO.8048147.v1.

[145] M. Vukov, A. Domahidi, H.J. Ferreau, M. Morari, and M. Diehl. Auto-generated algorithms for nonlinear model predictive control on long and on short horizons. *52nd IEEE Conference on Decision and Control*, pages 5113–5118, 2013.

[146] L. Wang. *Model Predictive Control System Design and Implementation Using Matlab*. Springer, 2009.

[147] Q. Weiwei, H. Bing, L. Beixuan, Y. Renping, and L. Gang. An improved dual-mode robust nonlinear MPC with one-step set optimization. *Chinese Control Conference, CCC*, pages 4004–4009, 2015.

[148] P. Williams. Quadrature discretization method in tethered satellite control. *Applied Mathematics and Computation*, 217(21):8223–8235, 2011.

[149] L. Wirsching, J. Albersmeyer, P. Kühl, M. Diehl, and H. Bock. An Adjoint-based Numerical Method for Fast Nonlinear Model Predictive Control. *IFAC Proceedings Volumes*, 41(2):1934–1939, 2008.

[150] M.H. Wright. The interior-point revolution in optimization: History, recent developments, and lasting consequences. *Bulletin of the American Mathematical Society*, 42(1):39–56, 2005.

[151] B. Xu, F. Sun, C. Yang, D. Gao, and J. Ren. Adaptive discrete-time controller design with neural network for hypersonic flight vehicle via back-stepping. *International Journal of Control*, 84(9):1543–1552, 2011.

[152] A. Zanelli, R. Quirynen, and M. Diehl. An Efficient Inexact NMPC Scheme with Stability and Feasibility Guarantees. *IFAC-PapersOnLine*, 48(18):53–58, 2016.

[153] A. Zanelli, R. Quirynen, G. Frison, and M. Diehl. A partially tightened real-time iteration scheme for nonlinear model predictive control. *IEEE 56th Annual Conference on Decision and Control, CDC*, pages 4388–4393, 2017.

[154] A. Zannelli, G. Horn, G. Frison, and M. Diehl. Nonlinear Model Predictive Control of a Human-sized Quadrotor. *European Control Conference*, 16(1):41–50, 2018.

[155] C. Zhang, H. Hu, D. Gu, and J. Wang. Cascaded control for balancing an inverted pendulum on a flying quadrotor. *Robotica*, 35(6):1263–1279, 2017.

[156] M. Zhang, X. Li, J. Liu, and H. Su. Dual-mode LQR-feedforward optimal control for non-minimum phase boost converter. *IET Power Electronics*, 10(1):92–102, 2017.

[157] Q. Zhang, Q. Wang, and G. Li. Nonlinear modeling and predictive functional control of Hammerstein system with application to the turntable servo system. *Mechanical Systems and Signal Processing*, 72-73:383–394, 2016.

Appendix A

FLOPS Comparison

Table A.1 presents a Floating Point Operation comparison between the Standard approach and the proposed Closed Loop Dual Mode of chapter 6. The algorithms are introduced in their order of use in their respective final RTI algorithms.

Method	Algorithm	Number of Multiplications
Standard	Calculate H (alg. 3.5)	$0.5N_p n_x^2 n_u(N_p-1)$
	Calculate E (alg. 3.2)	$0.5N_p n_x n_u[N_p(2n_x+n_u)+n_u]$
	Calculate D (alg. 3.6)	$N_p n_x^2$
	Calculate f (alg. 3.3)	$N_p n_x n_u(2n_x+n_u)-n_x^2 n_u$
	Calculate $\tilde{X}$ (alg. 3.7)	$(N_p-1)n_x^2+N_p n_x n_u$
Closed Loop Dual Mode	Calculate H/F (alg. 6.4)	$0.5N_p n_x n_u(N_p-1)(n_x+n_u)$
	Calculate E (alg. 6.2)	$0.5N_p n_u[N_p(2n_x^2+2n_x n_u+n_u^2)+n_u^2]$
	Calculate D/S (alg. 6.5)	$N_p n_x(n_x+n_u)$
	Calculate f (alg. 6.1)	$N_p(n_x^2 n_u+n_x n_u^2+n_u^3)+(n_x n_u^2+n_x^2 n_u)(N_p-1)$
	Calculate $\tilde{X}/\tilde{U}$ (alg. 6.6)	$N_p n_x(n_x+2n_u)-n_x^2-n_x n_u$
Method	Algorithm	Number of Summations
Standard	Calculate H (alg. 3.5)	$0.5N_p n_x n_u(N_p-1)(n_x-1)$
	Calculate E (alg. 3.2)	$0.5N_p n_u[N_p(2n_x^2+n_x n_u-n_x-n_u)+n_x+n_u]$
	Calculate D (alg. 3.6)	$N_p n_x^2$
	Calculate f (alg. 3.3)	$N_p n_u(2n_x^2+n_x n_u-n_x-n_u+1)-n_x^2 n_u$
	Calculate $\tilde{X}$ (alg. 3.7)	$(N_p-1)n_x^2+N_p n_x(n_u-1)$
Closed Loop Dual Mode	Calculate H/F (alg. 6.4)	$0.5N_p n_u(N_p-1)(n_u+n_x)(n_x-1)$
	Calculate E (alg. 6.2)	Expression is too long
	Calculate D/S (alg. 6.5)	$N_p(n_x^2+n_x n_u-n_u)$
	Calculate f (alg. 6.1)	$N_p n_u(3n_x^2+n_x n_u+n_u^2-n_x-n_u)-2n_x^2 n_u$
	Calculate $\tilde{X}/\tilde{U}$ (alg. 6.6)	$N_p(n_x^2+2n_x n_u+n_x)-n_x(n_x+n_u)$

Table A.1: Floating Point Operations (FLOPS) of the main algorithms from the standard and closed loop dual mode approaches which are relevant for real-time implementation.

Appendix B

Infinite Horizon Feedback Gains for NMPC Pre-stabilisation: A Comparison

This chapter is meant to show a simple and easy to replicate comparison between using infinite horizon feedback gains in the general NMPC "prestabilisation" method established in chapter 6 for each of the linearised state space system matrices (ie. A_k, B_k), and using those proposed in the method of chapter 6. Although their usage may be intuitive given the general stability properties they embed on the A_k matrix, it is important to recognise that the multiplication between matrices with eigenvalues inside the unit circle DOES NOT guarantee the resulting matrix will also have the eigenvalues in the unit circle. Moreover, if the system undergoes through uncontrollable/unstabilisable periods or sections along the prediction horizon, the standard DARE procedure wouldn't be able to obtain a control law for those specific time steps. Moreover, their calculation takes significantly more time than the proposed approach. All of these represent important disadvantages which can be overcome by the general "pre-stabilisation" procedure introduced in chapter 6.

Let us begin by considering a small unstable and nonlinear "generic" state space system defined by:

$$x_{k+1} = \begin{bmatrix} 2.2 & -1.2 \\ 1 & 0 \end{bmatrix} x_k + \begin{bmatrix} b_1 \\ 0 \end{bmatrix} u_k \tag{B.1}$$

where the coefficient b_1 is time-varying, defined as $b_1 = 0.1 + 0.1 \sin\left(\frac{2\pi k}{N_p}\right)$ which contains the non-linearity where the coefficient is varying with in a smooth/simple sinusoidal manner along the prediction horizon (N_p). This results in an unstable system with eigenvalues in $eig(A) = [1, 1.2]$.

Consider now an optimisation with a prediction horizon of $N_p = 100$, a state penalisation weight of $q_k = diag([1, 1])\ \forall k = [1 \rightarrow N_p]$, and an input penalisation weight of $r_k = diag([0.1])\ \forall k = [0 \rightarrow N_p - 1]$. Note that the b_1 coefficient is 0 for $k = 75$, which would lead to an unstable/uncontrollable system in which no infinite horizon gain can be obtained in general. For the purpose of this example, the feedback gain at that time step will be set to zero. This is automatically achieved when using the proposed approach as the matrix B_k becomes zero which consequently makes the expression of K_k from equation (6.2b) to be zero.

Table B.1 summarises the condition numbers obtained for the Hessian (E) when using different version of pre-stabilisation including: no pre-stabilisation (standard approach), pre-stabilising by using infinite horizon gains for each A_k, B_k pair in algorithm 6.4, and pre-stabilising by using the proposed approach, ie. the one defined by algorithms 6.3 and 6.4. From this table it can be appreciated that the system is indeed heavily unstable, reaching a condition number with up to 16 orders of magnitude. On the other hand, it can also be seen that although the infinite horizon gain pre-stabilisation approach does indeed result in a low condition number of 46.776, the proposed approach gives a better result with a condition number 0.4543, clearly showing the deficiencies of using infinite horizon gains for this particular.

Method	Condition Number
Standard (No prestabilisation)	$4.1132E+16$
Pre-stabilised w/Infinite Horizon Gain	46.776
Pre-stabilised w/proposed approach	0.4543

Table B.1: Comparison of Condition Number of E when using different versions of pre-stabilisation.

This provides a small example comparison where the proposed approach was able to outperform the infinite horizon gains. However, there may be other cases where the latter (or indeed any other approach) performs better, and a definitive statement of superiority cannot yet be made. Nonetheless, if the system enters the terminal region, an infinite horizon cost can be embedded in the final weight (q_{N_p}) which would cause the immediately preceding feedback gain (K_{N_p-1}) to be the infinite horizon gain for the linear region, followed by the rest of the feedback gains embedded in the infinite horizon framework.

In general, because of the significantly increased computation times that would be required to compute the infinite horizon gains online, it is recommended to use the proposed approach which was found to give satisfactory performance with the benefit of freedom in selecting the tuning for the "pre-stabilisation" procedure, as well as avoiding any potential issue due to uncontrollable/unstabilisable regions around the nonlinear trajectory.

Appendix C

Interior Point Method

One of the methods that was studied as part of the learning of this Ph.D. were Interior Point methods. This allowed a deeper understanding of the underlying procedures required by this type of Quadratic Programs which ultimately were used to evaluate the computational performance, particularly when using the method presented in chapter 6 for the Inverted Pendulum case study 6.4 as seen in table 6.6. In this case study, we were interested in how fast can the resulting Quadratic Programs of chapter 6 be "run" using general Primal-Dual Interior Point methods as the ones described in [39, 91, 150], as well as how do they compare when used for standard NMPC QPs.

Consider a general Quadratic Program of the form:

$$J = \frac{1}{2}\hat{Z}^T E \hat{Z} + \hat{Z}^T f \qquad M\hat{Z} \geq \gamma \tag{C.1}$$

The optimal solution via Primal-Dual Interior Point methods reduces to the repeated solution of the linear system of equations as (C.2) until the Karush-Kush-Tucker (KKT) conditions are satisfied ($\delta\hat{Z} = 0$, $\delta\lambda = 0$, $\lambda > 0$, $M\hat{Z} \geq \gamma$), with the logarithm barrier term decreasing to ($\mu = \mu_{ini} \to 0$). The system is given by:

$$\begin{bmatrix} E & -M^T \\ \Lambda M & C \end{bmatrix} \begin{bmatrix} \delta\hat{Z}^* \\ \delta\lambda^* \end{bmatrix} = \begin{bmatrix} -f - E\bar{Z} + M^T\bar{\lambda} \\ \mu\mathbf{1} - C\bar{\lambda} \end{bmatrix} \tag{C.2}$$

where $\bar{Z}$ and $\bar{\lambda}$ are guesses of the optimal solution; $\delta\hat{Z}^*$ and $\delta\lambda^*$ are optimal deviations from the guesses towards the optimal solution; $\Lambda = diag(\bar{\lambda})$; $C = diag(M\bar{Z} - \gamma)$; and $\mathbf{1}$ is a vector of ones which multiplies the logarithm barrier weight (μ). Note that the constraints must be in the form $M\hat{Z} \geq \gamma$ for proper operation operation of the interior point, ie. both M and γ from the general QP frameworks (3.33, 3.31, 4.4, 4.5, 6.14) presented throughout the thesis must be negated.

The reader can verify that an efficient solution to (C.2) can be obtained by:

$$\delta\hat{Z}^* = (E - M^T C^{-1}\Lambda M)^{-1}(-f - E\bar{Z} + M^T C^{-1}\mu\mathbf{1}) \tag{C.3a}$$

$$\delta\lambda^* = -\bar{\lambda} + C^{-1}(\mu\mathbf{1} - \Lambda M\delta\hat{Z}^*) \tag{C.3b}$$

Having defined this, the solution of the QP using this Interior Point method reduces to 3 particular tasks: the selection of the initial guess for $\bar{Z}^{[0]}$ and $\bar{\lambda}^{[0]}$ required to be feasible (inside the interior of the barriers), the selection of appropriate step-size that satisfies $M(\bar{Z}^{[i]} + \alpha\delta\hat{Z}^{*}) - \gamma \geq \epsilon$ and $\bar{\lambda}^{[i]} + \alpha\delta\lambda^{*} \geq \epsilon$, and the systematic reduction of the logarithm barrier weight (μ). In this case, we took a simple approach for these 3 tasks on ours tests for the case study of section 6.4 such as exponential decrease of (μ), ratio-test based selection of α [38, 40], and use of heavily penalised slacks (soft-constraints) for calculating initial guesses which satisfied the basic requirements and were observed to obtain proper optimal solutions. Regardless, the main interest of its implementation was not to obtain an actual solution but to quantify the computational burden related to the repeated solution of (C.3), as discussed in the aforementioned case study.

Appendix D

Publications

This appendix includes all the publications and submissions related to the work of this thesis, namely:

- A 2018 IFAC NMPC Conference (Abstract-only) Paper [48]: "A Time-Varying Shifting Strategy for Block Based MPC Solutions using a RTI Scheme", related to the contents of chapter 5.
- A 2018 UKACC Conference Paper [49]: "Laguerre-based Adaptive MPC for Attitude Stabilisation of Quadrotor", related to the contents of chapter 4 experimentally applied to the physical Quadrotor visible in (https://youtu.be/RSe35TjjBPI).
- A 2020 IET Control Theory and Applications journal paper [50]: "Shifting Strategy for Efficient Block-based Non-linear Model Predictive Control using Real-Time Iterations", related to the contents of chapter 5.
- A 2020 IET Control Theory and Applications journal paper [47]: "Fast Hybrid Dual-Mode NMPC for a Parallel Double Inverted Pendulum with Experimental Validation", related to the contents of chapter 6 based on the experimental application of the proposed approach visible in (https://youtu.be/7E-SXi3YKQo).
- A 2020 IFAC World Congress publication [59]: "Model Predictive Control for Wave Energy Converters: A Moving Window Blocking Approach", related to the contents of chapter 5.
- A 2020 IFAC World Congress publication [100]: "Towards Control of Autonomous Surface Vehicles in Rough Seas", related to the contents of chapter 4.
- A 2020 IEEE Transactions on Automatic Control submission: "Dual Mode Stable Prediction Models for Numerically Robust Fast Nonlinear Model Predictive Control using Real-Time Iterations", related to the contents of chapter 6.

A Time-Varying Shifting Strategy for Block Based MPC Solutions using a RTI Scheme

O. J. Gonzalez Villarreal* **J. A. Rossiter***

* *The University of Sheffield, Dept. ACSE, Sheffield, UK (e-mail: ojgonzalezvillarreal1@sheffield.ac.uk, j.a.rossiter@sheffield.ac.uk)*

Abstract: The solution of Nonlinear Model Predictive Control problems in real-time demands efficient solutions and formulations that give approximate/suboptimal solutions which have good performance whilst preserving important properties such as stability and recursive feasibility guarantees. This paper presents a time-varying shifting strategy for block based solutions, which allow to reduce the computational burden while maintaining recursive feasibility guarantees that are not present in standard blocked solutions. The formulation is validated through the simulation of an inverted pendulum and the results indicate that, not only does it recover the recursive feasibility guarantees but also, it delivers better performance by performing a well-posed optimization.

Keywords: Nonlinear Model Predictive Control (NMPC), Real-Time Iterations (RTI), Blocking, Recursive Feasibility, Well-Posed Optimization

1. INTRODUCTION

Over the last decade, Nonlinear Model Predictive Control (NMPC) has been gaining attention given the increasing computing power available nowadays which allows its online implementation in fast systems. Its popularity is mainly due to its ability to handle constraints and complex or highly nonlinear multivariable systems.

One of the most prominent approaches for fast online optimisation in the context of NMPC is the Real Time Iteration (RTI) Scheme (Diehl et al., 2005). An excellent review of this method is presented in Gros et al. (2016). The ACADO toolkit (Houska et al., 2015) is an open-source software capable of automatic code generation that allows implementation of this method for solving Optimal Control Problems (OCP) for different types of systems, such as continuous, discrete, ODEs, DAEs, etc. (Quirynen et al., 2013). In the case of continuous time systems, it uses single or multiple shooting to reduce the OCP to an approximate but tractable non-linear program (NLP). Depending on the requirements of the user, as well as the size of the system itself, the optimization can be done using condensing approaches or sparse solutions. An earlier study (Vukov et al., 2013) concluded the condensing approach is faster for small to medium systems, whereas the sparse solution is preferred for larger scale systems.

Another popular approach to reduce the computational burden within NMPC is that of blocked solutions, however a common problem with this type of approach is recursive feasibility. In Cagienard et al. (2007), a "Moving Window Blocking" strategy is presented that solves this by using a set of blocking matrices, similar to the concept presented in this paper but with significant differences. The work in Kouzoupis et al. (2015) presented the application of block condensing using the ACADO toolkit, but critically, with no consideration of the recursive feasibility problems.

This paper focuses on condensed solutions for the solution of NMPC problems (although the proposed approach is equally applicable to sparse solutions) and explores block based solutions for the reduction of the computation burden related to the optimization. To solve the recursive feasibility problems present in this type of formulation, we propose a time-varying shifting strategy which relies on fixing the points of interest in the time domain and shifting the optimization accordingly. To validate the concept, the paper presents a simulation of the swing-up and stabilization of an inverted pendulum, a common benchmark for NMPC. The results show that recursive feasibility is recovered and the proposed approach gives significantly better performance than previous alternatives. Additionally, restrictions in the block size (Kouzoupis et al., 2015) are now not required.

The paper is organized as follows: Section 2 presents the NMPC formulation based on Taylor linearisation for discrete systems, derives the optimization matrices, give two types of solution, the incremental and absolute and introduces the concepts used by the RTI Scheme. Section 3 presents the blocked solution which embeds an input-structure into the optimization. Section 4 presents the proposed time-varying shifting strategy for block based solutions to retain recursive feasibility problem and embeds a well-posed optimization giving better performance and more stable and reliable solutions. Section 5 presents numerical results obtained from an inverted pendulum simulation where the proposed strategy clearly outperforms earlier non time-varying shifting strategies. Finally, section 6 contains conclusions and future work.

2. NONLINEAR MODEL PREDICTIVE CONTROL

2.1 Modeling

Throughout this paper, discrete-time nonlinear dynamics of the following form will be considered:

$$\begin{aligned} x_{k+1} &= f(x_k, u_k) \\ y_k &= g(x_k, u_k) \end{aligned} \tag{1}$$

where x_k are the states, u_k are the controls or inputs of the system and y_k are the outputs. For continuous-time models, *direct methods* such as *single* or *multiple shooting* can be used to discretize the system and reduce the infinite Optimal Control Problem (OCP) to an approximate but tractable and finite NLP and simulate the nominal trajectories and linearisation matrices used by the formulation.

2.2 Prediction

By expanding a Taylor series up to first order terms, the system (1) can then be approximated by:

$$\begin{aligned} x_{k+1} &= f(\bar{x}_k, \bar{u}_k) + \frac{\partial f(\bar{x}_k, \bar{u}_k)}{\partial \bar{x}_k}\delta x_k + \frac{\partial f(\bar{x}_k, \bar{u}_k)}{\partial \bar{u}_k}\delta u_k \\ &= \bar{x}_{k+1} + A_k \delta x_k + B_k \delta u_k \end{aligned} \tag{2}$$

where $\delta x_k = x_k - \bar{x}_k$ and $\delta u_k = u_k - \bar{u}_k$ represent the deviation of the state and input from the nominal point at time step $t = k$, and $A_k = \frac{\partial f(x_k, u_k)}{\partial x_k}$ and $B_k = \frac{\partial f(x_k, u_k)}{\partial u_k}$ represent the partial derivatives of the system dynamics. Notice the deviation $\delta x_{k+1} = x_{k+1} - \bar{x}_{k+1}$ at time step $t = k + 1$ is then approximated by:

$$\delta x_{k+1} = A_k \delta x_k + B_k \delta u_k \tag{3}$$

Given that the nominal points $\bar{x}_{k+1}$ and linearization matrices (A_k, B_k) of (2) depend parametrically on $\bar{x}_k$ and $\bar{u}_k$, and that at a given sampling time $t = k$ the value of $\bar{x}_k$ is already given either by measurements or by state estimation, the value of $\bar{x}_{k+1}$ can only be obtained by assuming the value of $\bar{u}_k$. If values are assumed not only for $\bar{u}_k$ but for all the **future nominal input trajectory** $\bar{U} = \begin{bmatrix} \bar{u}_k & \bar{u}_{k+1} & \cdots & \bar{u}_{k+N_p-1} \end{bmatrix}^T$, this allows us to simulate the system forward and generate the nominal points $\bar{X} = \begin{bmatrix} \bar{x}_{k+1} & \bar{x}_{k+2} & \cdots & \bar{x}_{k+N_p} \end{bmatrix}^T$ and linearisation matrices A_k and B_k at future time steps $t = k+1, k+2, \cdots, k+N_p$. From now on, $\bar{X}$ will be referred as the **predicted nominal state trajectory**.

Once the predicted nominal state trajectory is obtained by using the future nominal input trajectory, the prediction equation (3) can be shifted forward by:

$$\delta x_{k+2} = A_{k+1} \delta x_{k+1} + B_k \delta u_{k+1} \tag{4}$$

Substituting equation 3 in 4 gives:

$$\begin{aligned} \delta x_{k+2} &= A_{k+1}(A_k \delta x_k + B_k \delta u_k) + B_{k+1} \delta u_{k+1} \\ &= A_{k+1} A_k \delta x_k + A_{k+1} B_k \delta u_k + B_{k+1} \delta u_{k+1} \end{aligned} \tag{5}$$

By repeating the process N_p steps ahead and considering only the output of the system, the deviations from the predicted nominal output trajectory can be **condensly** represented by:

$$\delta \hat{Y} = G \delta x_k + H \delta \hat{U} \tag{6}$$

where $\delta \hat{Y} = \hat{Y} - \bar{Y}$ and matrices G and H are given by:

$$G = \begin{bmatrix} C_1 A_0 \\ C_2 A_1 A_0 \\ \vdots \\ C_{N_p} A_{N_p-1} \cdots A_1 A_0 \end{bmatrix} \tag{7}$$

$$H = \begin{bmatrix} C_1 B_0 & O & \cdots & \cdots \\ C_2 A_1 B_0 & C_2 B_1 & O & \cdots \\ C_3 A_2 A_1 B_0 & C_3 A_2 B_1 & C_3 B_2 & \cdots \\ \vdots & \vdots & \vdots & \ddots \\ C_{N_p} A_{N_p-1} \cdots A_1 B_0 & C_{N_p} A_{N_p-1} \cdots A_2 B_1 & \cdots & \cdots \end{bmatrix} \tag{8}$$

where $C_k = \frac{\partial g(\bar{x}_k, \bar{u}_k)}{\partial \bar{x}_k}$ is the partial derivative w.r.t the nominal state in (1), and O represents a matrix of zeros with the same dimensions of $C_k B_k$. Notice the k notation in A_k and B_k has been dropped for simplicity.

2.3 Optimization

Once the prediction is formulated, a quadratic cost function penalizing the predicted errors between the **reference trajectory** Y_r and the predicted output trajectory $\hat{Y}$ and the input trajectory $\hat{U}$ magnitude can be formulated as:

$$J = \frac{1}{2}(Y_r - \hat{Y})^T Q (Y_r - \hat{Y}) + \frac{1}{2}\hat{U}^T R \hat{U} \tag{9}$$

where Q is a positive-definite matrix penalizing the predicted errors with dimensions $n_y N_p \times n_y N_p$ and R is a positive-semi definite matrix penalizing input deviations with dimension $n_u N_u \times n_u N_u$.

In the following, two types of solutions will be formulated: the first one giving increments to the nominal input trajectory $\delta \hat{U}$ and the second one giving the input trajectory directly $\hat{U}$. Although both solutions give exactly the same answer, the inequality constraints are expressed differently. Let us first reformulate cost function (9) by expressing it in the standard QP form:

$$J = \frac{1}{2} z^T E z + f^T z \quad s.t \quad Mz \leq \gamma \tag{10}$$

where z is the decision variable to be optimized depending on which formulation (incremental or absolute) is chosen, M is the linearisation matrix of the inequality constraints and γ is a time varying vector with the constraints. The equality constraints can be implemented by selecting the upper and lower limits of the inequality constraints to be the same value.

Incremental Formulation

Substituting expressions (6) and the definition of $\hat{Y}$ and $\hat{U}$ into cost function (9) gives:

$$\begin{aligned} J = \frac{1}{2}(Y_r - \bar{Y} - G\delta x_k - H\delta \hat{U})^T Q (Y_r - \bar{Y} - G\delta x_k - H\delta \hat{U}) \\ + \frac{1}{2}(\bar{U} + \delta \hat{U})^T R (\bar{U} + \delta \hat{U}) \end{aligned} \tag{11}$$

By optimizing w.r.t $\delta \hat{U}$, the optimization is of the standard QP form (10) where $E = H^T Q H + R$ is formally known as the Hessian and $f = -(H^T Q (Y_r - \bar{Y} - G\delta x_k) - R\bar{U})$ is formally known as the gradient. In this case, both input and output inequality constraints must be expressed relative to the nominal trajectory:

$$M = \begin{bmatrix} I \\ -I \\ H \\ -H \end{bmatrix} \qquad \gamma = \begin{bmatrix} U_{max} - \bar{U} \\ -(U_{min} - \bar{U}) \\ Y_{max} - \bar{Y} - G\delta x_k \\ -(Y_{min} - \bar{Y} - G\delta x_k) \end{bmatrix} \tag{12}$$

Absolute Formulation

Similarly, substituting expression (6) and the definition of $\hat{Y}$ and $\delta\hat{U}$ into cost function (9) gives:

$$J = \frac{1}{2}(Y_r - \bar{Y} - G\delta x_k - H\hat{U} + H\bar{U})^T Q$$
$$(Y_r - \bar{Y} - G\delta x_k - H\hat{U} + H\bar{U}) + \frac{1}{2}\hat{U}^T R\hat{U} \tag{13}$$

Once again, by optimizing w.r.t $\hat{U}$, the optimization is of the standard QP form (10) where $E = H^T QH + R$ and $f = -H^T Q(Y_r - \bar{Y} - G\delta x_k + H\bar{U})$. In this case, the input constraints are expressed in absolute values and the output constraints remain relative:

$$M = \begin{bmatrix} I \\ -I \\ H \\ -H \end{bmatrix} \qquad \gamma = \begin{bmatrix} U_{max} \\ -U_{min} \\ Y_{max} - \bar{Y} - G\delta x_k + H\bar{U} \\ -(Y_{min} - \bar{Y} - G\delta x_k + H\bar{U}) \end{bmatrix} \tag{14}$$

Having defined this, any QP solver such as qpOASES can be used to solve the optimization. For our simulation, the quadprog function from Matlab was used.

2.4 Real-Time Iteration Scheme

The RTI Scheme is a method developed by Diehl et al. (2005) for Nonlinear Optimization in Optimal Feedback Control that is capable of giving real-time performance based on the following strategies:

(1) **Initial Value Embedding**
It uses the solution found in the previous step in a *shifted* version, typically duplicating the last input variable $u_{k+N_p} = u_{k+N_p-1}$, to obtain the trajectory over which the formulation will linearise and optimize. Additionally, in the case of constrained optimization, it also uses a *shifted* version of the Lagrange multipliers λ found in the previous optimization.

(2) **Computation Divisions**
It separates the computations into feedback and preparation phases. The *preparation phase* uses a *predicted state evolution* obtained from the last *input trajectory* in its shifted version to linearise and prepare a QP. Once the state becomes available, the *feedback phase* quickly delivers an approximate solution.

(3) **Approximate QP Solution**
To further reduce the computational burden, it performs only one QP iteration and accepts an approximate solution the decreases the cost function J.

The main relation between the proposed approach and the RTI Scheme lies on the modification of the shifting strategy used in the IVE.

3. BLOCKED SOLUTION

A popular method for reducing the computational burden further is by using blocked solutions where the inputs or decision variables are blocked in sections and assumed to have the same value, e.g. $u_k = u_{k+1} = \cdots = u_{k+N_B-1}$ where N_B is the block size, thus reducing the number of degrees of freedom and consequently the optimization time. This blocked solution can be represented by an input structure of the form:

$$\delta\hat{U} = \mathbb{N}\delta\hat{\mathbb{U}} \tag{15}$$
$$\hat{U} = \mathbb{N}\hat{\mathbb{U}} \tag{16}$$

where $\hat{\mathbb{U}}$ (or $\delta\hat{\mathbb{U}}$) is the blocked decision variable and $\mathbb{N}$ is the blocking matrix defined as:

$$\mathbb{N} = \begin{bmatrix} \mathbb{I} & \mathbb{O} & \cdots & \mathbb{O} \\ \mathbb{O} & \mathbb{I} & \cdots & \mathbb{O} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{O} & \mathbb{O} & \cdots & \mathbb{I} \end{bmatrix} \tag{17}$$

$\mathbb{I}$ is a matrix containing N_B vertically blocked identity matrices of n_u dimension and $\mathbb{O}$ is a matrix of zeros of the same dimension. By substituting (15) and (16) into cost functions (11) and (13) respectively, the following modified Hessians and gradients can be found.

Incremental Formulation

$$E^* = \mathbb{N}^T(H^T QH + R)\mathbb{N} \tag{18}$$
$$f^* = -\mathbb{N}^T(H^T Q(Y_r - \bar{Y} - G\delta x_k) - R\bar{U}) \tag{19}$$

Absolute Formulation

$$E^* = \mathbb{N}^T(H^T QH + R)\mathbb{N} \tag{20}$$
$$f^* = -\mathbb{N}^T H^T Q(Y_r - \bar{Y} - G\delta x_k + H\bar{U}) \tag{21}$$

Efficient code can be generated to compute both previous solutions which save a significant amount of computations and reduce the required memory of the relevant matrices.

Definitions (15) and (16) can be used to recover the solution in the original variables. This is also relevant when representing the constraints as in (12) or (14). Finally, it should be noted that in contrast to the unblocked case where both formulations (absolute or incremental) result in the same solution, in this case, they can present different solutions given that it is conceptually different to embed this blocked structure in either increments or absolute variables. However, if the time-varying shifting strategy presented next is used, they do give the same results which is a consequence of deploying a consistent optimization.

4. TIME-VARYING SHIFTING STRATEGY

One potential problem of using the blocked solution above is the lack of recursive feasibility guarantees. Using the definition of recursive feasibility discussed in Cagienard et al. (2007), an optimization is said to be recursively feasible if and only if, for any feasible solution found at time step $t = k$, all solutions found at future steps $t = k+1, k+2, \cdots, k+\infty$ are also feasible. A simple way of guaranteeing recursive feasibility is to include the solution at the previous sampling (also known as the "tail") in the possible input choices at the current time step (Rossiter, 2018). The problem with a standard blocking approach is that it fails this simple test; the input trajectory from the previous sample is not available in the choices at the current sample and, as such, a recursive feasibility assurance cannot be given, in general.

One possible work around (Mendez et al., 2000) is to utilize the tail explicitly. If the input choice was feasible previously and one cannot find a feasible solution at the current step, then a shifted version of the previous decision should still be feasible (nominal case). However, such a choice is, in effect, open-loop and thus will not be rejecting any uncertainty in the system. This observation presents a mayor problem for the RTI Scheme with standard blocking given that a well-posed optimization (Rossiter, 2018) is premised on the fact that a shifting of the previous solution will be included automatically in the optimization.

This section proposes a time-varying shifting strategy with analogies to multi-rate approaches (Rossiter et al., 2005) that fixes the recursive feasibility and ill-posed optimization problems present in simplistic blocking. The proposed strategy fixes the attention of the optimization to absolute rather than relative points in time at which either inputs or decisions must be made, and constraints must be satisfied. This method is relevant when the points of interest occur at a significantly lower frequency than the sampling time. Notice that although in this paper the formulation is presented for discrete time models, this concept is equally applicable to shooting methods where the system must be discretized. The concept is illustrated in figure (1) where the red asterisk represents the optimization at time $t = k$ looking at 5 shooting nodes or points of interest sampled at $T_s = 0.4$ (s). Assuming that the actual sampling time of the system is $T_s = 0.2$ (s), the usual shifting strategy of the RTI used with the shooting concepts would look into the black crossed nodes and forget about the red asterisk nodes of the previous optimization. In contrast, the proposed approach (TV) represented by the green squares remain looking at the same red points of the previous optimization and add the last node to keep a constant prediction horizon.

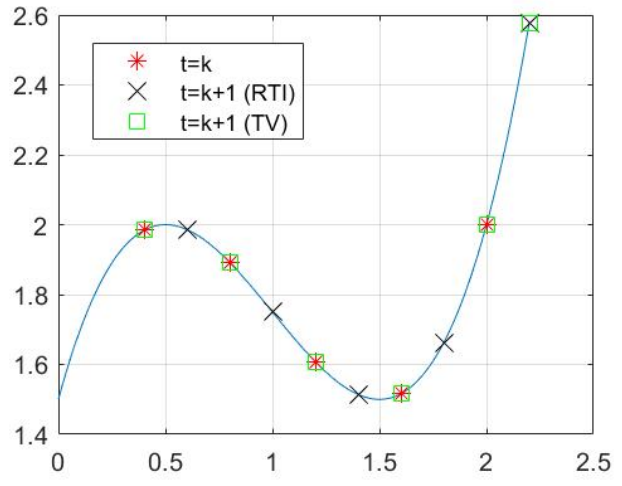


Fig. 1. Time-Varying Strategy Illustration

When thinking about the decision variable, the concept of time-varying shifting illustrated in figure (1), leads to a time-varying blocking structure $\mathbb{N}$ that is used sequentially as the optimization progresses. To understand this, consider a small system of prediction horizon $N_p = 4$ with a single input $n_u = 1$ and a block size $N_B = 2$. For this system, the standard time-invariant blocking is given by:

$$\mathbb{N} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \tag{22}$$

However, notice that in order to **make the decisions on the same points in time** matrix (22) must be shifted in time accordingly, hence giving the next two possible representations:

$$\mathbb{N}_1 = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \qquad \mathbb{N}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{23}$$

It can be shown that for an input-structure with N_B blocks, there will be N_B possible representations that maintain consistency along time. Notice this may cause the modified Hessians E^* to vary in length from $\lceil \frac{N_p}{N_B} \rceil$ to $\lceil \frac{N_p}{N_B} + 1 \rceil$ as shown in the example above for $\mathbb{N}_2$. An alternative would be to have a time-varying horizon or to select an horizon that keeps this lengths fixed, e.g. $N_p = 51, N_B = 2$.

Finally, **the points of interest related to the outputs must also be fixed in time**, as illustrated in figure (1), for the optimization to be consistent. This means that the optimization will select a **time-varying subset of rows** from H, $H_{\mathbb{N}}$, G, M, γ, $\bar{Y}$ and Y_r matrices and vectors, that ensures the same points are always optimised.

5. SIMULATIONS

To test the proposed formulation, a challenging benchmark problem was selected such as the nonlinear inverted pendulum which is a under-actuated single-input multiple-output non-minimum phase nonlinear system where the task is to swing-up and stabilize the system. The latter was achieved in a real inverted pendulum system by Mills et al. (2009) using modest hardware. The idea now is to see if the optimisation can be done using much simpler algorithms, thus allowing faster computation times.

5.1 System Modeling

The simplified nonlinear differential equations describing the dynamics of an inverted pendulum can be found in Alamir (2014) and are given by:

$$\begin{bmatrix} \ddot{\theta} \\ \ddot{p} \end{bmatrix} = \begin{bmatrix} -b\dot{\theta} + g\sin\theta + \cos\theta u \\ u \end{bmatrix} \tag{24}$$

Assuming the state $x = \begin{bmatrix} \theta & \dot{\theta} & p & \dot{p} \end{bmatrix}^T$, the system was simulated using a forward euler integration method which leads to the following linearized state space model:

$$\delta x_{k+1} = A_k \delta x_k + B \delta u_k \tag{25}$$

$$A_k = \begin{bmatrix} 1 & T & 0 & 0 \\ T\alpha_k & (1 - Tb) & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad B_k = \begin{bmatrix} 0 \\ Tc\theta_k \\ 0 \\ T \end{bmatrix} \tag{26}$$

where T is the sampling time, $c\theta_k = \cos\theta_k$, $s\theta_k = \sin\theta_k$ and $\alpha_k = gc\theta_k - s\theta_k u_k$. It is important to remember that the dynamics were still simulated using the forward euler integration to generate the nominal outputs $\bar{Y}$, namely the position p and angle θ of the system.

For our simulation, the parameters were $b = 0.3$ and $g = 9.806655$, and the simulation was done using a sampling time of $T = 0.025$ (s). Furthermore, the prediction horizon was set at $T_p = 1.25$ (s) (i.e $N_p = 50$) and the tuning parameters were $R = 0.1I$ and $Q = I$. Following the ideas presented in Mills et al. (2009), a terminal weight of $Q = 500$ was added to the end of both position and angle of the pendulum to aid with the convergence properties of the optimisation, given the required swing-up is highly non-linear. Finally, the system was constrained to positions $-1 < p < 1$ and inputs/accelerations of $-10 < u < 10$.

5.2 Results and Discussion

The system was simulated for blocking sizes up to 12 starting with the pendulum at the lower equilibrium position in the center of the sliding bar ($x_0 = [\pi \ 0 \ 0 \ 0]^T$) and with a nominal input trajectory of zeros. Table (1) presents a cost comparison for each blocking size using 4 different blocked methods, namely absolute and incremental with and without the proposed time-varying (TV) shifting. Additionally, whenever the system encountered a recursive feasibility violation (ie. an infeasible flag) or a non-convex QP (ie. a non-convex flag) from quadprog, the previous solution (ie. the tail) was used to retain feasibility. The number of recursive feasibility violations were counted and are shown in brackets in the same table. If no violations were recorded, the brackets were omitted.

From Table (1) it can be seen that the TV formulations have identical values for J as expected, and they present an overall better cost indicated by the final ('Total') row. Furthermore, although the non TV incremental formulation presents slightly better performance in some cases (specifically $N_B = [2, 3, 6, 9]$), notice it has a significant problem with recursive feasibility and an overall unstable behavior as the block size increases. In the case of the absolute formulation, the recursive feasibility problems only occurs once for this scenario, however, it delivers significantly worse behavior and in most cases is not able to control the system effectively as seen in figure (2). Finally, an interesting case is visible at $N_B = 10$ were the optimization wasn't able to swing-up the pendulum in "one shot", thus giving significantly sub-optimal performance. This is unsurprising and linked to the obvious observation that there is a upper limit to sensible block sizes.

N_B	TV-Inc	TV-Abs	Inc	Abs
1	1101	1101	1101	1101
2	1104	1104	1103 [10]	1128 [2]
3	1111	1111	1109 [8]	1288
4	1125	1125	1149 [16]	2532
5	1219	1219	1239 [16]	4149
6	1376	1376	1229 [13]	4116
7	1145	1145	2495 [12	4157
8	1238	1238	2573 [29]	3960
9	1360	1360	1227 [17]	3904
10	2560	2560	2552 [20]	3480
11	1326	1326	4261 [28]	3458
12	1109	1109	4076 [38]	3979
Total	15773	15773	24115 [207]	37253 [2]

Table 1. Cost (J) comparison alongside number of optimisation failures.

Furthermore, because the optimisation is only taking into account intermediate points defined by the block size, and the fact that the predicted corrections are approximations given by equation (3), there exists the possibility of small constraint violations, as illustrated in figure (3), where the position presented a small constraint violation at $t \approx 0.8(s)$. Notice that the violation presented in figure (3) measures only 10 time-steps in length (less than the block size $N_B = 12$) and it wasn't detected as a recursive feasibility problem in table (1), most likely because the points of interest did respect the constraint at $t = 0.6$ (s) $(2N_BT_s)$ and $t = 0.9$ (s) $(3N_BT_s)$ (see close up in figure). A simple solution for this is to use a small slack variable in the constraints to protect the solution from this small violations. To quantify this, the relative distance between the constraint limit and the violation was accumulated and is presented in table (1) where it can be seen that the TV-Inc formulation presented the least overall constraint violation, and naturally, the worst is the Inc solution as expected from Table (1).

N_B	TV-Inc	TV-Abs	Inc	Abs
1	0.000	0.000	0.000	0.000
2	0.003	0.003	0.004	0.003
3	0.005	0.005	0.006	0.000
4	0.017	0.017	0.016	0.000
5	0.000	0.000	0.097	0.107
6	0.061	0.061	0.093	0.213
7	0.210	0.210	0.280	0.324
8	0.074	0.074	0.496	0.517
9	0.082	0.082	0.192	0.211
10	0.341	0.341	0.458	0.000
11	0.108	0.108	0.280	0.000
12	0.415	0.415	1.480	0.279
Total	1.3169	1.3169	3.4014	1.6529

Table 2. Constraint Violation Comparison

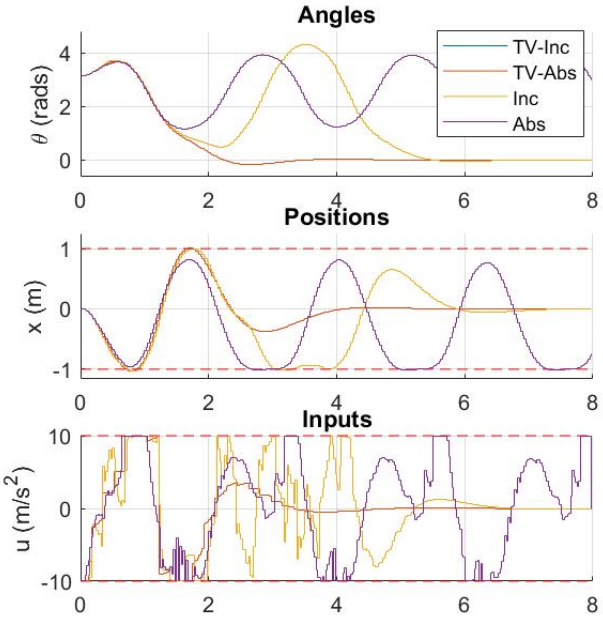


Fig. 2. Example Performance Comparison with $N_B = 7$

Figure (2) shows a comparison between all the 4 strategies using a block size of $N_B = 7$. This figure clearly illustrates

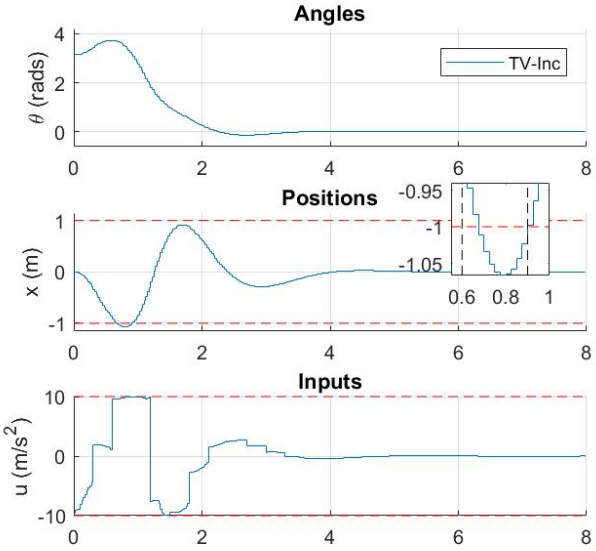


Fig. 3. Example TV-Inc Performance with $N_B = 12$

the benefit of using the proposed time-varying method leading to precise and visibly noticeable blocked actions required for the swing-up and stabilization. To further emphasize the resulting blocking concept, figure (3) clearly exhibits a blocked input structure after $t \approx 2.5(s)$ when the system is stabilized within the prediction horizon. Finally, notice the system is able to get a sub-optimality of $\Delta J < 0.8\%$ with this block size ($N_B = 12$) where sub-optimality is defined as $\Delta J = (J_{N_B}/J_1 - 1) \times 100$.

N_B	TV-Inc	TV-Abs	Inc	Abs
1	1044	1044	1044	1044
2	1075	1075	1077	1072
3	1089	1089	1095	1082
4	1096	1096	1105	1087
5	1101	1101	1113	1093
6	1103	1103	1124	1101
7	1106	1106	1141	1114
8	1122	1122	1132	1138
9	1119	1119	1128	1171
10	1122	1122	1142	1214
11	1130	1130	1150	1295
12	1169	1169	1165	1353
Total	13275	13275	13416	13763

Table 3. Unconstrained Optimization Cost (J) Comparison using Saturated Control

Finally, table (3) presents the unconstrained case of the solution where the proposed strategies (TV-Inc or TV-Abs), outperform the others in the overall cost and in some cases ($N_B = 12$) giving a difference of up to $\Delta J_{TV} - \Delta J_{Abs} = 17.62\%$ sub-optimality reduction.

6. CONCLUSION

This paper proposed a time-varying shifting strategy for blocked-based solutions of NMPC problems which have problems with recursive feasibility guarantees and ill-posed optimization. The formulation relies on fixing the attention to absolute points in time in which decisions must be made or constraints must be enforced, thus giving a set of optimisation routines that run consistently from one sample to the next.

The results suggest the proposed approach gives much better performance and ensures that decisions can be made whilst preserving recursive feasibility. In this work, only the nominal case was studied and further testing is required to analyse the performance under with uncertainty. However, it is important to recall that more rigorous approaches require substantially heavier computational loads which are not aligned with the objectives of this study.

Future work will include analysing the computational load of this formulation, which is expected to be significantly reduced given the reduced number of degrees of freedom and constraints that are imposed in the optimisation. Furthermore, it will be merged with the ACADO toolkit and bench-marked using qpOASES as QP the solver.

REFERENCES

Alamir, M. (2014). Fast NMPC: A reality-steered paradigm: Key properties of fast NMPC algorithms. *2014 European Control Conference, ECC 2014*, (4), 2472–2477.

Cagienard, R., Grieder, P., Kerrigan, E.C., and Morari, M. (2007). Move blocking strategies in receding horizon control. *Journal of Process Control*, 17(6), 563–570.

Diehl, M., Bock, H.G., and Schlöder, J.P. (2005). A Real-Time Iteration Scheme for Nonlinear Optimization in Optimal Feedback Control. *SIAM Journal on Control and Optimization*, 43(5), 1714–1736.

Gros, S., Zanon, M., Quirynen, R., Bemporad, A., and Diehl, M. (2016). From linear to nonlinear MPC: bridging the gap via the real-time iteration. *International Journal of Control*, 7179(November), 1–19.

Houska, B., Ferreau, H.J., and Diehl, M. (2015). Auto-generating microsecond solvers for nonlinear MPC: A tutorial using ACADO integrators. *Optimal Control Applications and Methods*, 36, 685–704.

Kouzoupis, D., Quirynen, R., Frasch, J.V., and Diehl, M. (2015). Block Condensing for Fast Nonlinear MPC with the Dual Newton Strategy. *IFAC-PapersOnLine*, 48(23), 26–31.

Mendez, J., Kouvaritakis, B., and Rossiter, J. (2000). State Space approach to interpolation in MPC. *International journal of robust nonlinear control*, 10, 27–38.

Mills, a., Wills, A., and Ninness, B. (2009). Nonlinear model predictive control of an inverted pendulum. *2009 American Control Conference*, 2335–2340.

Quirynen, R., Gros, S., and Diehl, M. (2013). Efficient NMPC for nonlinear models with linear subsystems. *Proceedings of the IEEE Conference on Decision and Control*, 5101–5106.

Rossiter, J.A. (2018). *A First Course in Predictive Control*. CRC Press, 2nd edition.

Rossiter, J., Sheng, J., Chen, T., and Shah, S. (2005). Interpretations of and options in dual-rate predictive control. *Journal of Process Control*, 15, 135–148.

Vukov, M., Domahidi, A., Ferreau, H.J., Morari, M., and Diehl, M. (2013). Auto-generated algorithms for nonlinear model predictive control on long and on short horizons. *52nd IEEE Conference on Decision and Control*, 5113–5118.

Proceedings Paper:
Gonzalez Villarreal, O., Rossiter, J.A. orcid.org/0000-0002-1336-0633 and Shin, H. (2018) Laguerre-based adaptive MPC for attitude stabilization of quad-rotor. In: Proceedings of 2018 UKACC 12th International Conference on Control (CONTROL). Control 2018: The 12th International UKACC Conference on Control, 05-07 Sep 2018, Sheffield, UK. IEEE , pp. 360-365. ISBN 978-1-5386-2864-5

https://doi.org/10.1109/CONTROL.2018.8516876

Laguerre-based Adaptive MPC for Attitude Stabilization of Quad-rotor

1st O. J. Gonzalez Villarreal
Dept. of ACSE
The University of Sheffield
Sheffield, UK
ojgonzalezvillarreal1@sheffield.ac.uk

2nd J. A. Rossiter
Dept. of ACSE
The University of Sheffield
Sheffield, UK
j.a.rossiter@sheffield.ac.uk

3rd H. Shin
Center for A.C.P.S
Cranfield University
Cranfield, UK
h.shin@cranfield.ac.uk

***Abstract*—The application of predictive control methods in real-time to fast systems, such as quad-rotors, remains a challenge for its implementation in low-power embedded systems. This paper presents the application of an Adaptive Laguerre-based Model Predictive Controller (MPC) to the Attitude Stabilization of a Quadrotor. The formulation uses an Online System Identification algorithm based on Recursive Least Squares (RLS) with forgetting factor for parameter estimation, and a Laguerre-based Model Predictive Controller for achieving real-time calculation/update of the control law. The developed control system was experimentally tested in a real quad-rotor, and the results demonstrate its real-time applicability in a low-power embedded platform.**

***Index Terms*—MPC, Laguerre, Adaptive, UAV, Quadrotor**

I. INTRODUCTION

Over the last decade, quad-rotors have become a very popular research topic, given their relatively low-cost, complex nonlinear dynamics and high maneuverability. This has led to all kinds of different applications, such as surveillance, aerial photography, surface mapping, search and rescue, inspection, transport, military and the recently, more popular, FPV racing [1]. Apart from their own inherent complexity, these vehicles are constantly affected by external disturbances, such as wind, as well as changes in the systems' dynamics, which can affect their performance and may require re-tuning to achieve good stability characteristics. As an example, the payload in transportation quad-rotors affects the inertia and mass properties. Similarly, it is common to constantly attach/detach components such as cameras, batteries and external sensors to the vehicle, once again, changing the vehicle's dynamics. These challenges require the implementation of flexible control schemes that are capable of dealing with these uncertainties and sudden changes, and are therefore the motivation behind using the adaptive predictive control scheme presented in this paper.

Model Predictive Control is an advanced optimization-based control strategy that uses an inner model of the system to predict and optimize its future behavior [2], [3]. With the advances in computation platforms and optimization algorithms, the implementation of predictive control algorithms to fast system is now looking more feasible [1]. In the area of UAVs, several authors have implemented MPC, or even Nonlinear MPC, both in simulation and real experiments [1]. In [4], a Laguerre-based MPC was developed and experimentally validated for an hexacopter based on the methodology described in [5], but with no parameter estimation. Other authors, such as [6] and [7] looked at the combination of nonlinear MPC with parameter and state estimation techniques for improving the systems' performance, but only in simulation.

The main contribution of this paper is the formulation of a simple SISO model of a quad-rotor, which differs from models presented in [1] by including actuator (or possibly sensor) dynamics. This model is then used by a real-time feasible Laguerre-based MPC control law, able to match a PD control law with the main advantage of including the desired frequency content of the Laguerre Polynomials in the design. Furthermore, this is combined with a computationally inexpensive Online System Identification algorithm for estimation of 3 parameters that define the systems dynamics with the goal of achieving auto-tuning. Moreover, the entire formulation is experimentally tested in a quad-rotor UAV, and the tests demonstrate successful implementation in the relatively new Beaglebone Blue board, which is a low-power embedded platform. The entire formulation is available from https://github.com/OscarJGV26/LaguerreMPC using object oriented programming and Matlab codes. In summary, the paper presents the application of a novel Adaptive Laguerre-based Model Predictive Controller (MPC) for Attitude Stabilization, experimentally tested in a Quad-rotor using relatively new hardware.

Section II presents the full nonlinear attitude model of a quad-rotor, and derives a linear SISO model to be used by the formulation presented in this paper with its respective assumptions. Section III presents the formulated Online System Identification using Recursive Least Squares (RLS) with a forgetting factor and discusses important aspects to be considered for its implementation. Section IV outlines the Laguerre-based MPC formulation and discusses some important remarks. Section V presents the results obtained from real experiments where the system dynamics were automatically excited using a sinusoidal signal for auto-tuning, and the convergence and computational benefits of the overall control systems are discussed. Finally, Section VI gives conclusions and future work.

II. Quadrotor Modeling

It is well known that the attitude model of a quad-rotor has several sources of nonlinear dynamics such as quaternion/euler dynamics, thrust relations and coriolis-centripetal crossed-coupled angular velocity effects [6]. Nevertheless, many authors have simplified them into linear models [4]. This section presents the fully nonlinear attitude model of a quad-rotor, and derives the associated simple linear SISO model to be used for the control design presented in this paper.

Recalling the modeling done in [6], [4] and [2]; the full nonlinear attitude dynamics of a quad-rotor can be represented by:

$$\begin{bmatrix}\dot{q}_0\\ \dot{q}_1\\ \dot{q}_2\\ \dot{q}_3\end{bmatrix} = \frac{1}{2}\begin{bmatrix}0 & -p & -q & -r\\ p & 0 & r & -q\\ q & -r & 0 & p\\ r & q & -p & 0\end{bmatrix}\begin{bmatrix}q_0\\ q_1\\ q_2\\ q_3\end{bmatrix} \tag{1}$$

$$\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}\frac{k_T l}{I_{xx}} & 0 & 0\\ 0 & \frac{k_T l}{I_{yy}} & 0\\ 0 & 0 & \frac{k_\tau}{I_{zz}}\end{bmatrix}\begin{bmatrix}1 & -1 & 1 & -1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\end{bmatrix}\begin{bmatrix}\omega_1^2\\ \omega_2^2\\ \omega_3^2\\ \omega_4^2\end{bmatrix} + \begin{bmatrix}\frac{I_{yy}-I_{zz}}{I_{xx}}qr\\ \frac{I_{zz}-I_{xx}}{I_{yy}}pr\\ \frac{I_{xx}-I_{yy}}{I_{zz}}pq\end{bmatrix} \tag{2}$$

where $[q_0, q_1, q_2, q_3]^T$ is the quaternion in the inertial frame, $[p, q, r]^T$ are the angular velocities in the body axes frame, $[\omega_1, \omega_2, \omega_3, \omega_4]^T$ are the propellers' angular velocities, I_{xx}, I_{yy}, I_{zz} are the vehicle's inertias, k_T, k_τ are constants relating the propellers' angular velocities and thrust/torque respectively, and l is the distance parallel to the respective axis from center of gravity (CG) to the propeller.

The dynamics of the quaternion (1) are affected by the data-fusion method used to correct drift, e.g. using accelerometer data [8]. Therefore, in order to avoid being affected by this in the Online System Identification phase, the formulation will focus on the rate dynamics (2) and use a cascade proportional control loop as common control loops. This can be improved further by using a quaternion based control such as in [9] or [10].

Now, assuming the cross-coupled angular velocity terms are negligible around the operating point $p \approx q \approx r \approx 0$, and assuming that at this operation point there exist a linear relationship between the input signal u_i (i.e. the signal that goes into the Electronic Speed Controllers (ESCs)) and the propellers' angular velocities ω_i, the rate dynamics can now be represented by:

$$\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = K\begin{bmatrix}L\\ M\\ N\end{bmatrix} \tag{3}$$

where K is a diagonal matrix with the gains of each axis given by:

$$K = \begin{bmatrix}k_x & 0 & 0\\ 0 & k_y & 0\\ 0 & 0 & k_z\end{bmatrix} \tag{4}$$

L, M, N represent the allocated "virtual moments" as in [11] given by:

$$\begin{bmatrix}L\\ M\\ N\\ Z\end{bmatrix} = \frac{1}{4}\begin{bmatrix}1 & -1 & 1 & -1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\\ 1 & 1 & 1 & 1\end{bmatrix}\begin{bmatrix}u_1\\ u_2\\ u_3\\ u_4\end{bmatrix} \tag{5}$$

and the resulting control allocation is given by:

$$\begin{bmatrix}u_1\\ u_2\\ u_3\\ u_4\end{bmatrix} = \begin{bmatrix}1 & 1 & 1 & 1\\ -1 & 1 & -1 & 1\\ 1 & -1 & -1 & 1\\ -1 & -1 & 1 & 1\end{bmatrix}\begin{bmatrix}L\\ M\\ N\\ Z\end{bmatrix} \tag{6}$$

We introduce Z to represent a potential offset of "thrust" to be produced by each of the 4 motors. The derivation of the control allocation matrices is out of the scope of this paper.

These dynamics represent a simple integrator and implementing an MPC directly onto then result in a proportional controller which is unlikely to give the desired tracking performance in the rate dynamics whilst also being robust to external disturbances. A key assumption for this model is that the propeller angular velocities are "instantaneously" achieved, which is not true, and nevertheless, has been used by many authors (see [1], [4]) Based on this potential problem, the model was further augmented with unit-gain first-order dynamics representing the actuators (or possibly even the sensors) dynamics. Combining both models gives the following second order dynamics with an integrator and with $\tau > 0$ related to the time constant of the first order; these will be used for the control law formulation.

$$\begin{bmatrix}\ddot{p}\\ \ddot{q}\\ \ddot{r}\end{bmatrix} = -\frac{1}{\tau}\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} + K\begin{bmatrix}L\\ M\\ N\end{bmatrix} \tag{7}$$

III. Online System Identification

Although the parameters of the dynamic model of a vehicle can be calculated and pre-stored off-line using for example CAD or system identification methods, the application of online system identification allows quick recalculation of the system's true dynamic model and therefore, can enhance the system performance. Additionally, it can be used for supervisory control and fault-detection/isolation, as well as fault-tolerant control; however this is outside the scope of this paper.

A. Algorithm and Modeling

In this case, the Online System Identification was based on the Recursive Least Squares (RLS) with forgetting factor

formulation presented in [12]. The algorithm equations are given by:

$$\begin{aligned} e_k &= z_k - \Psi^T \Theta \\ K &= \tfrac{P\Psi}{1+\Psi^T P \Psi} \\ \Theta_k &= \Theta_{k-1} + K e_k \\ P &= \tfrac{I - K\Psi^T}{\lambda} P \end{aligned} \tag{8}$$

where e_k is the prediction error of the mode, Ψ is known as the regressors vector, Θ is the parameters vector, P is the covariance matrix, K is a Kalman-Filter-type gain and λ is the forgetting factor. For our system, a forgetting factor of $\lambda = 0.999$ was selected.

This formulation was combined with the assumed model (7) given in the previous section. Although a classic approach would formulate this model as a standard ARX model represented by a discrete difference equation of the form:

$$y_k = a_1 y_{k-1} + a_2 y_{k-2} + b_1 u_{k-1} + b_2 u_{k-2} \tag{9}$$

which has $n_a = 2$ recursive terms and $n_b = 2$ exogenous terms. In our case, we implemented the Δ modeling and identification approach given in [13]. This allows the embedding of the desired model structure into the system whilst also improving the precision of the coefficients by requiring 2 less coefficients to be estimated. Moreover, the learned system must consider possible disturbances or un-modeled biased errors. Therefore the parameters to be estimated were augmented with a constant disturbance. The algorithm entry-data (z_k, Ψ) are then given by:

$$\begin{aligned} z_k &= y_k - 2y_{k-1} + y_{k-2} \\ \Psi &= \begin{bmatrix} y_{k-1} - y_{k-2} \\ u_{k-1} \\ 1 \end{bmatrix} \quad \Theta = \begin{bmatrix} a \\ b \\ d \end{bmatrix} \end{aligned} \tag{10}$$

where y is a general output-variable, which in this case represents the angular velocity of the vehicle, and u_k is a general input-variable, which in this case represent the "virtual moments". Once the model is learned, by rearranging the equations, the state space model to be used later (see eqn. (13)) can be found.

B. Execution Rules and Parameter Constraints

The performance of the algorithm presented above is known to converge to the real parameters if and only if: (i) the assumed model (7), or in general, the entry data (10) can actually represent the system dynamics, and (ii) if the system is under persistent excitation periods [12]. Therefore, in order to prevent unwanted adaptation and learning actions in period of low excitation, several execution rules and parameter constraints were implemented and are listed and explained below based on the ideas discussed in [12], with the selected thresholds for our system.

1) Run the RLS algorithm only when the "angular acceleration" $y_{k-1} - y_{k-2} > 5$ is greater than a threshold to provide sufficient excitation and avoid running the algorithm when steady, e.g. when hovering. Additionally, the criteria found in [12] based on the normalized information was also used:
$$\frac{||P\Psi||_1}{||P||_1 ||\Psi||_1} < 0.1 \tag{11}$$
2) Limit the trace of the co-variance matrix (P) with
$$P = \frac{k_{lim}}{tr(P)} P \quad \textit{if} \quad tr(P) > k_{lim} \tag{12}$$
to prevent it from becoming ill-conditioned and have better control on the rate of convergence. A threshold of $k_{lim} = 10$ was selected for our system.
3) Limit the range of parameters a and b to an expected range to prevent incorrect modeling. For our system, the thresholds $-0.7 < a < 0$, $0.05 < b < 0.3$ for roll/pitch axis, and $0.005 < b < 0.05$ for yaw axis, were selected and can be obtained considering variation in the time constant and gain of the system.
4) Only update the control law (i.e. adapt) if the overall uncertainty of your first two parameters/coefficients a, b, which can be considered as the summation of $P_{1,1} + P_{2,2}$, is sufficiently small. This not only ensures that adaptations are made when you can actually trust your coefficients, but also when they are moving slowly. For this system, a maximum trace of $k_{max} = 0.0001$ was selected.

Remark 1 (Saturation): One thing to be careful with when using this formulation is the saturation of the actuators. Whenever this happens, the allocated control values do not represent the same values of the "virtual moments" in matrix (6). Therefore, in this case, the system must recalculate the actual allocated "virtual moments" values with matrix (5) after saturation.

IV. Laguerre-based Model Predictive Control

Laguerre-based MPC uses a set of discrete orthonormal basis functions embedded into the design. The main motivations behind using Laguerre are: (i) the possible recovery of the fully optimal solution; (ii) the acceleration of the computation times and (iii) a better frequency control on the system's response. Furthermore, by imposing a fast zero-decaying structure, it prevents the optimization from becoming ill-conditioned in case of plants with unstable/conditionally stable dynamics. The methodology can be found in [14] and [5].

A. Model

One important difference in this implementation is the model to be used. Most MPC implementations use the Δ modeling approach where the system is augmented with an integrator. However, based on the results from [15], it was chosen to use a disturbance estimation model which indeed gave better performance as well as improved control over disturbance rejection, thus motivating its use for unbiased-predictions. Furthermore, in order to consider a possible match to a proportional-derivative (PD) controller, the model was

transformed to an equivalent by using a simple backward-forward euler integration method. Finally, in order to be able to formulate the unbiased optimization-index (16), the model had to be represented with difference inputs (Δu_k), rather than the absolute values (u_k). Thus, the state space model used for this formulation is given by:

$$\begin{aligned} x_{k+1} &= Ax_k + B\Delta u_k \\ y_k &= Cx_k \end{aligned} \tag{13}$$

where:

$$A = \begin{bmatrix} 1 & T_s(1+a) & T_s & b \\ 0 & (1+a) & 1 & \frac{b}{T_s} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} b \\ \frac{b}{T_s} \\ 0 \\ 1 \end{bmatrix}$$
$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$$

T_s is the sampling time and the state defined as $x_k = \begin{bmatrix} y_k & \frac{y_k - y_{k-1}}{T_s} & d_k & u_{k-1} \end{bmatrix}^T$, with d_k being the disturbance to be estimated. From now on, we will refer to $\dot{y}_k = \frac{y_k - y_{k-1}}{T_s}$.

To estimate the disturbance a full Kalman Filter can be employed as in [15], or alternatively, the disturbance can be simply filtered with a unit-gain first-order discrete filter given by:

$$\begin{aligned} d_k = \alpha d_{k-1} + \\ (1-\alpha)(\dot{y}_k - (1+a)\dot{y}_{k-1} + \tfrac{b}{T_s}u_{k-1}) \end{aligned} \tag{14}$$

where $0 < \alpha < 1$ is the tuning constant. For our system, a value of $\alpha = 0.98$ was selected.

B. Control Law

The derivation of the control law is based on finite horizon optimal control using a relatively long horizon which is known to give good stability characteristics [3], [16]. Given that the purpose of the optimization is to calculate a control law in real-time whilst preserving computational simplicity, dual-mode approaches were avoided.

In this section, it will be shown that the formulation leads to a control law of the form:

$$u_k = u_{k-1} - K_x x_k + K_r r_k + K_{\dot{r}} \dot{r}_k \tag{15}$$

where K_x, K_r and $K_{\dot{r}}$ vary depending on the system dynamics.

To achieve this, a standard unbiased-optimization index of the form:

$$J = (\boldsymbol{r} - \hat{\boldsymbol{y}})^T (\boldsymbol{r} - \hat{\boldsymbol{y}}) + \boldsymbol{\Delta\hat{u}}^T R \boldsymbol{\Delta\hat{u}} \tag{16}$$

is defined, where $\hat{\boldsymbol{y}} = \begin{bmatrix} y_{k+1} & y_{k+2} & \cdots & y_{k+N_p} \end{bmatrix}^T$ and $\boldsymbol{\Delta\hat{u}} = \begin{bmatrix} \Delta u_k & \Delta u_{k+1} & \cdots & \Delta u_{k+N_p-1} \end{bmatrix}^T$ represent the vectors of predicted outputs and future input-increments trajectories respectively, N_p is the prediction horizon, and $\boldsymbol{r} = \begin{bmatrix} r_{k+1} & r_{k+2} & \cdots & r_{k+N_p} \end{bmatrix}^T$ represents a reference trajectory, typically a constant r_k.

The unbiased-predicted output $\hat{\boldsymbol{y}}$ is represented by:

$$\hat{\boldsymbol{y}} = Gx_k + H\boldsymbol{\Delta\hat{u}} \tag{17}$$

with

$$G = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^{N_p} \end{bmatrix} \quad H = \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ CA^{N_p-1}B & \cdots & \cdots & CB \end{bmatrix} \tag{18}$$

Now, the dynamics of the Laguerre Polynomials can be found in [5] and are given by,

$$\begin{bmatrix} L(1) \\ L(2) \\ \vdots \\ L(N_L) \end{bmatrix}_{k+1} = \begin{bmatrix} a_L & 0 & \cdots & 0 \\ \beta & a_L & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \cdots & \cdots & \cdots & a_L \end{bmatrix} \begin{bmatrix} L(1) \\ L(2) \\ \vdots \\ L(N_L) \end{bmatrix}_k \tag{19}$$

where, a_L is the decay-rate, $\beta = (1 - a_L^2)$ and N_L is the number of Laguerre coefficients.

By taking $L_0 = \sqrt{\beta}\begin{bmatrix} 1 & -a_L & \cdots & (-1)^{N_L-1} a_L^{N_L-1} \end{bmatrix}^T$ as initial condition and iterating system (19) forward N_p times, the following input structure can be embedded into the optimization

$$\boldsymbol{\Delta\hat{u}} = L\eta \tag{20}$$

where $L = \begin{bmatrix} L_0^T & L_1^T & \cdots & L_{N_p-1}^T \end{bmatrix}^T$ and $\eta = \begin{bmatrix} c_1 & c_2 & \cdots & c_{N_L} \end{bmatrix}^T$ are the Laguerre coefficients [5]. For our system, the number of Laguerre coefficients was fixed with $N_L = 2$ and the decay-rate was fixed at $a_L = 0.5$, which contains a desirable frequency response of the input. Figure (1) shows the embedded input parameterization.

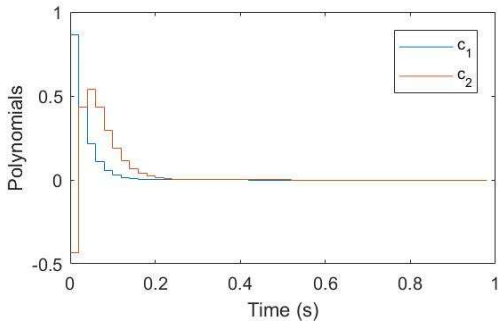


Fig. 1. First two Laguerre Polynomials.

By substituting equations (20) and (17) in (16), it can then be derived that the optimization is of the standard Quadratic Problem (QP) form:

$$J = \frac{1}{2}\eta^T E \eta + f^T \eta \tag{21}$$

where $E = L^T H^T H L + L^T R L$, $f^T = -L^T H^T (r - Gx_k)$. This optimization will give the optimal Laguerre coefficients η which can then be used to recover the solution in the original space using (20). Recalling the receding horizon strategy [5], only the first input is applied and the optimization is recalculated in the following next step. If the system dynamics

are linear time-invariant, then, the unconstrained solution of this QP is of the form of (15) with:

$$K_x = -\begin{bmatrix}1 & 0 & \cdots & 0\end{bmatrix} LE^{-1}L^T H^T G \tag{22}$$

and it can be shown that $K_r = K_x(1)$. To further match a PD controller and give better trajectory tracking, the control law included a desired angular acceleration, $\dot{r}_k = \frac{r_k - r_{k-1}}{T_s}$. Thus, the final control law is given by (15) with $K_{\dot{r}} = K_x(2)$.

Remark 2 (Coding): Significant computation savings can be achieved by proper coding and memory allocation of this optimization, e.g. by using recursive information [2]. Efficient Matlab code for this is available from https://github.com/oscarjgv26.

Remark 3 (Saturation): Because it is unconstrained, the saturation of the actuators will be done using anti-windup techniques as discussed in [4] as, in such cases, Δ based control laws are known to have good anti-windup properties.

V. Experimental Results

This formulation was tested in an F450 quad-rotor frame with EMAX MT2213 brushless motors, ESCs operating at 400 Hz and 1045 ABS propellers. The flight control system running this formulation was a Beaglebone Blue running @ 1000 MHz and the formulations were compiled using -O3 C flag. The filter of both on-board gyroscope and accelerometer was set at 10 Hz, and the resolutions were set at 2000 (deg/s) and $4G$ respectively. The accelerometer was fused using the double-stage Kalman Filter presented in [8] but only performing accelerometer correction with $Q = 10^{-6}I$ and $R_{acc} = 0.1$. The sampling time of the control system was $T_s = 20(ms)$ (50 Hz) and the prediction horizon was fixed at $N_p = 50$, i.e. $T_p = 1\ (s)$. The outer loop proportional gains for the Roll and Pitch axis were both set on $K_p = 4$ and left constant throughout the tests. The input range of each of the motors was $u_i = [0, 1000]$, and the allocated moments (v_i) were saturated at approximately 60% of the overall hovering throttle Z which depends on the mass, thus time-varying but for our system, approximately $v_i = [-300, 300]$.

One of the most important, and also difficult things to achieve was the stability of the combined online system identification algorithm and control law update. This is because if the standard RLS algorithm is applied, the system can diverge during periods of poor excitation which leads to the need for the rules presented in section III-B. Another important part was that, regardless of the identified model, or in general, the performance of the online system identification algorithm, there are still three constants that determine the performance of the system, namely the input-weight R, the disturbance estimation filter α and the Laguerre decay-rate a_L. For the purpose of this paper, these constants were fixed at the values $R = 0.1, \alpha = 0.98, a_L = 0.5$, where "stability" and "smoothness" were observed across the tests. However, in general, these parameters do need to be assigned carefully, in particular taking into account the possible embedded frequency response of both Laguerre decay-rate and disturbance estimation constant α which could have a substantial effect on the system.

A. Flight Tests

To test the formulation, the system was excited in the roll axis, starting from a poorly tuned control law, and moving iteratively towards the optimal value. Figure (2) shows approximately 7 seconds of the flight test data. At the beginning ($t < 1$), the system is showing the performance of the poorly tuned control law. At $t \approx 1$, an automatically generated sinusoidal signal of $\phi = 50\sin(4\pi)$ in the roll angle is implemented for approximately 3 seconds to provide initial excitation and the online system identification starts (see figure (3)). The control law starts updating at $t \approx 4$ (see figure (4)) and maintains the same values after the pilot terminates the sinusoidal excitation to test the performance. As it can be seen, the "chattering" of the inputs was reduced to ± 10, thus giving very precise corrections and reducing actuator wear. Once again, the flight data is available at https://github.com/OscarJGV26/LaguerreMPC.

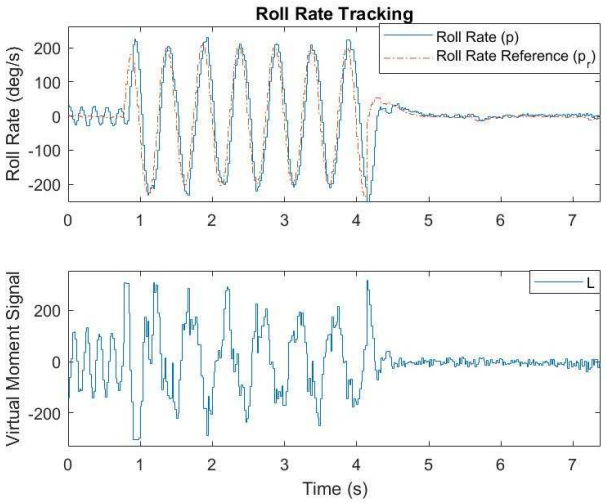


Fig. 2. Excitation Data - Roll Axis

Furthermore, to test the full learning capabilities, the RLS algorithm started from an empty parameter vector $\Theta = \begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$ during this test. The variation of the principal estimated coefficients (a, b) can be seen in figure (3). As it can be seen, they move near to the final value within less than 3 seconds whilst also staying within the constraints, and move smoothly afterwards as the algorithm iterates .

During the same test, figure (4) shows the movement of the control law parameters which can be seen to be moving equally smoothly and do not change during the first 3 seconds of the RLS algorithm where the uncertainty is still high. Similar results were obtained for the other 2 axis (pitch/yaw) and the formulation was able to tune all the axis simultaneously with the sinusoidal signal within 10 sec.

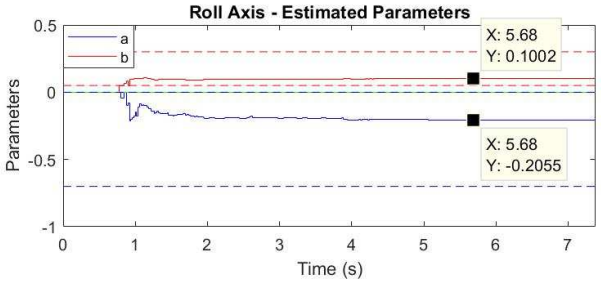


Fig. 3. Online System Identification - Estimated Coefficients

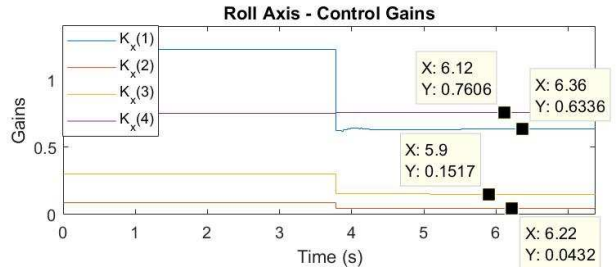


Fig. 4. Control Law Gains

B. Computation Times

One of the requirements of this formulation is that it is able to be run in real-time whilst performing other tasks such as data-fusion, telemetry, data-logging and so forth. Table I shows the computation times in miliseconds (t_c) of the formulation, where updates of the control law in 235 microseconds, and execution of the online system identification in 4.8 microseconds (per axis) can be seen, leaving more than enough time for other tasks. Furthermore, it includes the computation times of 30 and 50 iterations for solving the Discrete Algebraic Ricatti Equation (DARE) for comparison.

TABLE I
ONLINE COMPUTATION TIMES

Task	$t_c(ms)$
Control Law Execution on 3 axis	0.0033
Online System Identification on 3 axis (RLS)	0.0144
Laguerre MPC ($N_L = 2$)	0.235
LQR (30 iterations of DARE)	0.165
LQR (50 iterations of DARE)	0.273

Although 30 iterations of DARE for computing the LQR control law are faster, the tuning process of LQR didn't give good performance given the lack of frequency content embedded into the optimization. Thus, the motivation behind using Laguerre Polynomials was validated.

VI. Conclusion

This paper presented the implementation of a Laguerre-based Model Predictive Control formulation coupled with Recursive Least Squares with forgetting factor as an Online System Identification method for achieving adaptive self-tuning control of a quad-rotor UAV in real-time. This formulation was experimentally tested in a quad-rotor and could equally be applied to other UAVs with similar dynamics using the control allocation concept.

The developed control system combined the standard outer loop proportional cascade loop control with a control allocation strategy, an online system identification method and a Laguerre-based Model Predictive Control.

The performance of the developed formulation was tested using an automated sinusoidal signal for exciting the system's dynamics and the results demonstrate the capability of the formulation to achieve self-tuning of the system in real-time within less than 3 seconds per axis with overall smooth performance of the parameters. A demonstration of this implementation will be given at the conference.

Future work related to this formulation will be the extension to the multi-variable system identification case and consideration of other types of UAVs.

References

[1] A. Zannelli, G. Horn, G. Frison, and M. Diehl, "Nonlinear Model Predictive Control of a Human-sized Quadrotor," *European Control Conference*, vol. 16, no. 1, pp. 41–50, 2018.

[2] R. Quirynen, S. Gros, and M. Diehl, "Efficient NMPC for nonlinear models with linear subsystems," *Proceedings of the IEEE Conference on Decision and Control*, pp. 5101–5106, 2013.

[3] J. A. Rossiter, *A first course in predictive control: 2nd edition*. CRC Press, 2018.

[4] J. A. J. Ligthart, P. Poksawat, and L. Wang, "Experimentally Validated Model Predictive Controller for a Hexacopter," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 4137–4142, 2017.

[5] L. Wang, *Model Predictive Control System Design and Implementation Using Matlab*. Springer, 2009.

[6] M. H. Tanveer, D. Hazry, S. F. Ahmed, M. K. Joyo, F. A. Warsi, H. Kamaruddin, M. Zuradzman, K. Wan, and A. B. Shahriman, "NMPC-PID Based Control Structure Design for Avoiding Uncertainties in Attitude and Altitude Tracking Control of Quadrotor," *IEEE 10th International Colloquium on Signal Processing & its Applications*, pp. 7–9, 2014.

[7] S. Iplikci, "RungeKutta model-based adaptive predictive control mechanism for non-linear processes," *Transactions of the Institute of Measurement and Control*, vol. 35, no. 2, pp. 166–180, 2013.

[8] S. Sabatelli, M. Galgani, L. Fanucci, and A. Rocchi, "A double-stage kalman filter for orientation tracking with an integrated processor in 9-D IMU," *IEEE Transactions on Instrumentation and Measurement*, vol. 62, no. 3, pp. 590–598, 2013.

[9] S. Bouhired, M. Bouchoucha, and M. Tadjine, "Quaternion-based global attitude tracking controller for a quadrotor UAV," *2013 3rd International Conference on Systems and Control, ICSC 2013*, pp. 933–938, 2013.

[10] E. Fresk and G. Nikolakopoulos, "Full Quaternion Based Attitude Control for a Quadrotor," *European Control Conference*, pp. 3864–3869, 2013.

[11] T. A. Johansen and T. I. Fossen, "Automatica Control allocation A survey," *Automatica*, vol. 49, no. November, pp. 1087–1103, 2013.

[12] S. F. Campbell, N. T. Nguyen, J. Kaneshige, and K. Krishnakumar, "Parameter Estimation for a Hybrid Adaptive Flight Controller," *AIAA Infotech@ Aerospace Conference, Seattle, WA*, no. April, pp. 1–27, 2009.

[13] S. R. Anderson and V. Kadirkamanathan, "Modelling and identification of non-linear deterministic systems in the delta-domain," *Automatica*, vol. 43, no. 11, pp. 1859–1868, 2007.

[14] B. Khan and J. Rossiter, "Alternative Parameterisation within predictive Control: a systematic selection," *IJC*, vol. 86, no. 8, pp. 1397–1409, 2013.

[15] J. Huusom, N. Poulsen, S. Jorgensen, and J. Jorgensen, "Tuning of methods for offset free MPC based on ARX model representations," *American Control Conference (ACC), 2010*, pp. 2355–2360, 2010.

[16] E. F. Camacho, *Model predictive control*. Advanced textbooks in control and signal processing, London ; New York: Springer, 2nd ed. ed., 2003.

IET Control Theory & Applications

Research Article

IET Journals
The Institution of Engineering and Technology

Shifting strategy for efficient block-based non-linear model predictive control using real-time iterations

ISSN 1751-8644
Received on 1st April 2019
Revised 2nd September 2019
Accepted on 24th December 2019
E-First on 13th March 2020
doi: 10.1049/iet-cta.2019.0369
www.ietdl.org

Oscar Julian Gonzalez Villarreal[1] ✉, *Anthony Rossiter*[1]

[1]*ACSE, The University of Sheffield, Sheffield, UK*

✉ *E-mail: ojgonzalezvillarreal1@sheffield.ac.uk*

Abstract: Non-linear model predictive control requires the use of efficient solutions and strategies for its implementation in fast/real-time systems. A popular approach for this is the real-time iteration scheme, which uses a shifting strategy, namely the initial value embedding, that shifts the solution from one sampling time to the next. However, this strategy together with other efficient strategies such as move blocking, present a recursive feasibility problem. This study proposes a novel modified shifting strategy which preserve both recursive feasibility and stability properties, as well as achieves a significant reduction in the computational burden associated with the optimisation. The proposed approach is validated through a simulation of an inverted pendulum where it clearly outperforms other standard solutions in terms of performance and recursive feasibility properties. Additionally, the approach was tested on two computing platforms: a laptop with an i7 processor and a Beaglebone Blue Linux-based computer for robotic systems, where computational gains compared to existing approaches are shown to be as high as 100 times faster.

1 Introduction

Non-linear model predictive control (NMPC) is an advanced non-linear optimisation method for an optimal feedback control that uses a mathematical model of a dynamical system to predict and optimise its future performance. Its popularity comes from its ability to handle constraints explicitly, as well as complex multivariable non-linear dynamic systems [1]. One of the main challenges is that the required optimisation represents a significant computational burden, which has limited its application to relatively slow systems such as chemical reactors [2]. However, given the improvements in electronics in the last two decades, its application to fast real-time systems is now looking more feasible [3].

A key requirement for the implementation in real-time of NMPC is the use of efficient solutions. Efficiency may come in many different forms, from approximate solutions and inexact mathematical representations to tailored coding, special hardware such as field programmable gate arrays [4] and other strategies.

One of the most popular NMPC approaches nowadays is the real-time iteration (RTI) scheme proposed in [5], where a set of strategies are used to achieve real-time performance, namely the initial value embedding (IVE), the approximated single SQP solution, and the computations separation, offering solutions in the microseconds range [6–8] whilst preserving stability guarantees [9]. A tutorial-like paper of the latter is presented in [10]. Among other solutions based on inexact and approximate solutions, the authors of [11] propose an inexact mathematical representation, which avoids having time-varying matrices and preserves stability and recursive feasibility guarantees for a non-negligible region of the state space. In [12], a similar approach based on adjoints and inexact Jacobians is presented. In [13], the authors propose an inexact updating scheme that allows a reduction in the number of sensitivity updates required at each sampling time, by using a curvature-like measure of non-linearity. The authors of [14] propose a partially tightened NMPC that uses a Riccati-like recursive equation combined with an interior-point like method that uses logarithmic barriers to remove the constraints at later stages of the prediction horizon. Finally, a common method for achieving faster computation times is by reducing the number of degrees of freedom; this can be done using input-parameterised solutions such as move blocking and Laguerre polynomials [15] where an input-structure is embedded into the decision variables. The authors of [16] proposed methods for solving the blocked optimisation using highly parallelisable algorithms, which allow for even faster computation times. However, most works in this area, including [3, 16–18] have used blocking for NMPC with no regard to the recursive feasibility problem it presents.

In the case of tailored coding, a wide variety of toolkits exist that facilitate the implementation of NMPC. One of the most popular is the ACADO toolkit [19], an open source code capable of exporting efficient automatically generated code. The authors of [20] provide a tutorial-like paper for this toolkit. In [21], a toolkit named VIATOC that also exports automatically generated code is presented. Other toolkits such as CasADI and GRAMPC are also discussed in [21]. Another important area of research is the development of efficient QP solvers. Several solvers are available nowadays such as qpOASES, FORCES, CVXGEN and qpDUNES [20, 22, 23]. Furthermore, the solution of NMPC problems can be done using simultaneous or sequential approaches as discussed in [24] resulting in sparse or condensed QPs. The authors of [7] concluded that using condensed QPs is computationally faster for small- to medium-sized optimisations, whereas sparse solutions have better performance for medium to large. However, one of the main benefits of simultaneous approaches is that they present better stability characteristics for unstable systems [8]. Finally, different methods can be used for discretising an optimal control problem (OCP) such as direct single/multiple-shooting and direct collocation [6].

This paper focuses on a single-shooting condensed/sequential NMPC approach. It uses the RTI scheme as a base line methodology in which the proposed shifting strategy is embedded, thus re-formulating the optimisation. Although the proposed methodology is formulated using this specific approach, it is possible to apply it using other approaches such as multiple-shooting (sequential or simultaneous), as well as in combination with other solutions and methods such as [16] or [25]. The proposed approach uses concepts from the block-based solution for linear MPC presented in [26] which are extended to an NMPC formulation and merged with the RTI scheme. The key contribution of this paper, which differs from both aforementioned works, is the modified shifting strategy. By conceptualising the optimisation in an absolute-time-frame, this allows the reduction of both the number of degrees of freedom and constraints, whilst also

preserving recursive feasibility guarantees and stability properties. The proposed approach is shown to give computational benefits up to 100 times faster than the standard RTI-NMPC solution.

The remaining part of this paper is organised as follows: Section 2 presents a detailed description of the modelling, prediction and optimisation methods to be used such as the RTI scheme and the block-based solutions. Section 3 develops the main ideas of the proposed shifting strategy presented in this paper and gives a simple overall generic example. Section 4 gives clear insight into important coding aspects for implementing the proposed approach and an example code applied to the benchmark problem of Section 5 is provided in [27, 28]. Section 5 presents a benchmark of this method applied to a fully non-linear inverted pendulum model and focuses mainly on the overall performance and recursive feasibility properties of the different strategies that were tested. Additionally, it presents the computation times of the proposed approach implemented in two different systems: a laptop with an i7 processor, and the aforementioned Beaglebone Blue Linux-board [29]. Finally, Section 6 contains conclusions, summarises the contribution of the paper and describes future work.

2 Non-linear model predictive control

2.1 Modelling

Throughout this paper, discrete-time non-linear dynamics of the following form will be considered:

$$\begin{aligned} x_{k+1|k} &= f(x_{k|k}, u_{k|k}) \\ y_{k|k} &= g(x_{k|k}, u_{k|k}) \end{aligned} \tag{1}$$

where x_k are the states, u_k are the controls or inputs of the system and y_k are the outputs, with n_x, n_u and n_y the number of states, inputs and outputs, respectively. The notation $k+1|k$ reads 'value at $k+1$ predicted at sample time k', and will only be used in full when needed for clarity.

Remark 1: Continuous-time models, can be discretised using any integration method to reduce the infinite OCP to an approximate but tractable and finite non-linear problem. This allows simulating the system forward using the future nominal input trajectory, and linearising along the resulting trajectory.

2.2 Prediction

By expanding a Taylor series up to first order terms, the system (1) can be approximated by

$$\begin{aligned} x_{k+1} &= f(\bar{x}_k, \bar{u}_k) + \frac{\partial f(\bar{x}_k, \bar{u}_k)}{\partial \bar{x}_k}\delta x_k + \frac{\partial f(\bar{x}_k, \bar{u}_k)}{\partial \bar{u}_k}\delta u_k \\ &= \bar{x}_{k+1} + A_k \delta x_k + B_k \delta u_k \end{aligned} \tag{2}$$

where $\delta x_k = x_k - \bar{x}_k$ and $\delta u_k = u_k - \bar{u}_k$ represent the state and input deviations from the nominal points at time step $t = k$, respectively, and $A_k = \partial f(x_k, u_k)/\partial x_k$ and $B_k = \partial f(x_k, u_k)/\partial u_k$ represent the partial derivatives of the system dynamics. Notice the deviation $\delta x_{k+1} = x_{k+1} - \bar{x}_{k+1}$ at time step $t = k+1$ can be approximated by

$$\delta x_{k+1} = A_k \delta x_k + B_k \delta u_k \tag{3}$$

Given that the nominal point $\bar{x}_{k+1}$ and linearisation matrices (A_k, B_k) of (2) depend parametrically on $\bar{x}_k$ and $\bar{u}_k$, and that at a given sampling time $t = k$ the value of $\bar{x}_k$ is already given either by measurements or by state estimation, the value of $\bar{x}_{k+1}$ can only be obtained by assuming (or guessing) a value for $\bar{u}_k$. If values are assumed/guessed for the future nominal input trajectory $\bar{U} = \begin{bmatrix} \bar{u}_k^T & \bar{u}_{k+1}^T & \cdots & \bar{u}_{k+N_p-1}^T \end{bmatrix}^T$, this allows the computation of the predicted nominal state trajectory $\bar{X} = \begin{bmatrix} \bar{x}_{k+1}^T & \bar{x}_{k+2}^T & \cdots & \bar{x}_{k+N_p}^T \end{bmatrix}^T$, and linearisation matrices A_k and B_k at future time steps $t = k+1, k+2, \ldots, k+N_p$, where N_p is known as the prediction horizon. This prediction assumption is known as single-shooting [20]. Common MPC strategies such as GPC, also use a control horizon N_u where the inputs after $k + N_u - 1$ are enforced to be the same [30]. Let us consider $N_u = N_p$ for now as this is just a special case of the blocked solution presented in Section 2.5.

Once $\bar{X}$ is obtained using $\bar{U}$, the prediction equation (3) can be shifted forward

$$\delta x_{k+2} = A_{k+1}\delta x_{k+1} + B_{k+1}\delta u_{k+1} \tag{4}$$

Thus, substituting (3) into (4) gives

$$\begin{aligned} \delta x_{k+2} &= A_{k+1}(A_k \delta x_k + B_k \delta u_k) + B_{k+1}\delta u_{k+1} \\ &= A_{k+1}A_k \delta x_k + A_{k+1}B_k \delta u_k + B_{k+1}\delta u_{k+1} \end{aligned} \tag{5}$$

By repeating the above process recursively for N_p steps and considering only the output of the system, the predicted deviations from the nominal output trajectory can be represented in condensed form by

$$\delta \hat{Y} = G\delta x_k + H\delta \hat{U} \tag{6}$$

where $\delta \hat{Y} = \hat{Y} - \bar{Y} = \begin{bmatrix} \delta y_{k+1}^T & \delta y_{k+2}^T & \cdots & \delta y_{k+N_p}^T \end{bmatrix}^{\mathrm{T}}$ are the output deviations, $\delta \hat{U} = \hat{U} - \bar{U} = \begin{bmatrix} \delta u_k^T & \delta u_{k+1}^T & \cdots & \delta u_{k+N_p-1}^T \end{bmatrix}^{\mathrm{T}}$ are the input deviations, and matrices G and H are given by

$$G = \begin{bmatrix} C_1 A_0 \\ C_2 A_1 A_0 \\ \vdots \\ C_{N_p} A_{N_p-1} \cdots A_1 A_0 \end{bmatrix} \tag{7}$$

$$H = \begin{bmatrix} C_1 B_0 & O & \cdots & \cdots \\ C_2 A_1 B_0 & C_2 B_1 & O & \cdots \\ C_3 A_2 A_1 B_0 & C_3 A_2 B_1 & C_3 B_2 & \cdots \\ \vdots & \vdots & \vdots & \ddots \\ C_{N_p} A_{N_p-1} \cdots A_1 B_0 & C_{N_p} A_{N_p-1} \cdots A_2 B_1 & \cdots & \cdots \end{bmatrix} \tag{8}$$

where G has dimensions of $N_p n_y \times n_x$, H has dimensions of $N_p n_y \times N_p n_u$, $C_k = \partial g(\bar{x}_k, \bar{u}_k)/\partial \bar{x}_k$ is the partial derivative w.r.t the nominal state in (1), and O represents a matrix of zeros with the same dimensions of $C_k B_k$. Notice the k notation in A_k and B_k has been dropped for simplicity.

2.3 Optimisation

Once the prediction is formulated, a quadratic cost function penalising the predicted errors between the reference trajectory Y_r and the predicted output trajectory $\hat{Y}$, as well as penalising the input trajectory $\hat{U}$, can be formulated as

$$J = \frac{1}{2}(Y_r - \hat{Y})^T Q (Y_r - \hat{Y}) + \frac{1}{2}\hat{U}^T R \hat{U} \tag{9}$$

where Q is a positive semi-definite matrix penalising the predicted errors with dimensions $n_y N_p \times n_y N_p$ and R is a positive definite matrix penalising input deviations with dimension $n_u N_p \times n_u N_p$. The latter represents an unbiased performance index for the inverted pendulum system when no disturbances are present, given it stabilises at $u_k = 0$. If other types of system are used, the cost function (9) must be reformulated slightly, e.g. using unbiased costs [31].

In the following, two types of solutions will be formulated: the first one based on deviations of the nominal input trajectory $\delta \hat{U}$ and the second one based on the input trajectory $\hat{U}$ directly. Although both solutions give the same result, the inequality constraints are

expressed differently. Let us first reformulate cost function (9) by expressing it in the standard QP form

$$J = \frac{1}{2} z^T E z + f^T z \quad \text{s.t} Mz \le \gamma \tag{10}$$

where z is the decision variable to be optimised depending on which formulation (deviations or absolute) is chosen, E is a symmetric matrix formally known as the Hessian with dimensions $N_E = N_p n_u \times N_p n_u$, f is a column-vector with dimension $N_p n_u$ typically referred as the linear term, and M and γ are a matrix and vector, respectively, related to the inequality constraints. Equality constraints can be implemented by selecting the upper and lower limits of the inequality constraints to be the same.

2.3.1 Deviations formulation: Substituting expression (6) and the definitions of $\hat{Y}$ and $\hat{U}$ into cost function (9) gives

$$\begin{aligned} J = \frac{1}{2}(Y_r - \bar{Y} - G\delta x_k - H\delta\hat{U})^T Q(Y_r - \bar{Y} - G\delta x_k - H\delta\hat{U}) \\ + \frac{1}{2}(\bar{U} + \delta\hat{U})^T R(\bar{U} + \delta\hat{U}) \end{aligned} \tag{11}$$

By optimising w.r.t $\delta\hat{U}$, the optimisation has the standard QP form (10) where $E = H^T Q H + R$ and $f = -(H^T Q(Y_r - \bar{Y} - G\delta x_k) - R\bar{U})$. Input and output inequality constraints are expressed relative to the nominal trajectory as

$$M = \begin{bmatrix} I \\ -I \\ H \\ -H \end{bmatrix}, \quad \gamma = \begin{bmatrix} U_{\max} - \bar{U} \\ -(U_{\min} - \bar{U}) \\ Y_{\max} - \bar{Y} - G\delta x_k \\ -(Y_{\min} - \bar{Y} - G\delta x_k) \end{bmatrix} \tag{12}$$

2.3.2 Absolute formulation: Similarly, substituting expression (6) and the definition of $\hat{Y}$ and $\delta\hat{U}$ into cost function (9) gives

$$\begin{aligned} J = \frac{1}{2}(Y_r - \bar{Y} - G\delta x_k - H\hat{U} + H\bar{U})^T Q \\ (Y_r - \bar{Y} - G\delta x_k - H\hat{U} + H\bar{U}) + \frac{1}{2}\hat{U}^T R\hat{U} \end{aligned} \tag{13}$$

Once again, by optimising w.r.t $\hat{U}$, the optimisation has the standard QP form (10) where $E = H^T Q H + R$ and $f = -H^T Q(Y_r - \bar{Y} - G\delta x_k + H\bar{U})$. The input and output constraints are expressed as follows:

$$M = \begin{bmatrix} I \\ -I \\ H \\ -H \end{bmatrix}, \gamma = \begin{bmatrix} U_{\max} \\ -U_{\min} \\ Y_{\max} - \bar{Y} - G\delta x_k + H\bar{U} \\ -(Y_{\min} - \bar{Y} - G\delta x_k + H\bar{U}) \end{bmatrix} \tag{14}$$

Having defined the QP problem, any QP solver such as qpOASES [22, 23] or quadprog MATLAB function can be used to find the solution. In this paper, the Hildreth's primal-dual quadratic programming procedure presented in [32] was used for the simulations of Section 5.2 given its simplicity and 'hot-starting' capabilities. A fixed amount of iterations were done to obtain predictable timings and accurate comparisons between the different approaches.

Remark 2: For rigorous closed-loop stability guarantees, suitable terminal costs or zero-terminal constraints must be added by modifying the relevant matrices appropriately [6, 20, 33].

2.4 RSI scheme

The RTI scheme is a method developed by Diehl *et al.* [5] for non-linear optimisation in optimal feedback control that is capable of giving real-time performance based on strategies summarised in the following subsections.

2.4.1 Initial value embedding: IVE uses the solution found in the previous step in a shifted version, typically duplicating the last input variable $u_{k+N_p|k+1} = u_{k+N_p-1|k}$, to obtain the nominal trajectory over which the formulation will linearise and optimise. Additionally, in the case of QPs with 'hot-start' capabilities such as an active-set, it also uses a shifted version of the Lagrange multipliers λ found in the previous optimisation.

2.4.2 Single SQP: One can further reduce the computational burden and achieve predictable timings, by performing only a single SQP, i.e. only linearise the optimisation once instead of re-linearising over and over until convergence. This is reasonable given that the optimisation is 'hot-started' from the previous solution, which is expected to be close to the optimal solution, provided no significant disturbances have entered the system. Additionally, because the problem is forced to finish solving the linearised QP rather fast to give a quick feedback correction, the number of iterations or allowed time for solving the QP must be limited. In general, the solution of the problem is not given exactly but as an approximation that is expected to decrease the cost J at each iteration. Moreover, one must be satisfied with finding a local minimum and the solution can be subject to small approximation errors given only one re-linearisation is done.

2.4.3 Computation separation: Computation separation is arguably the most important strategy. It separates the computations into feedback and preparation phases. A timing diagram illustrating this can be found in [10, 34].

(a) *Preparation phase*: The preparation phase uses a predicted state $\hat{\hat{x}}_{k|k-1}$ as a starting point obtained from the last nominal input trajectory in its shifted version to linearise and prepare a QP. The standard RTI scheme only performs the aforementioned tasks and solves the QP in the feedback phase, however, in this work a small modification is used where the QP is iterated during preparation assuming $\delta x_k = 0$ to find the vector of Lagrange multipliers λ which is then used to compute the solution as given in (15) or (16).
(b) *Feedback phase*: Once the state measurement becomes available, the feedback phase quickly delivers an approximate solution by calculating the predicted state deviation $\delta x_k = x_k - \hat{\hat{x}}_{k|k-1}$ and computing the 'feedback phase' parts of (15) or (16), depending on which type of solution is being used. This allows the optimisation to have robustness against noise, disturbances and uncertainty. Because the state deviation δx_k has an effect, not only on the linear term f, but also in constraint vector γ (see (11)–(14)), the standard RTI scheme recomputes them before solving the QP.

To further elaborate on the strategy of computation separation, notice that the value of δx_k in cost functions (11) and (13) only makes sense to be used in the context of the RTI scheme, in particular, in the feedback phase. Assuming the vector of Lagrange multipliers (λ), has been found by an appropriate QP in the preparation phase, the solution for both types (deviations and absolute), can be expressed as the summation of the QP result which is calculated in the preparation phase, and the effect of δx_k which is calculated in the feedback phase of the RTI. From the QP procedure presented in [32], it can be shown that the both solutions are given by the expressions below:

(i) *The deviations solution is given as*

$$\hat{U} = \bar{U} - E^{-1}\left[\underbrace{\overbrace{-(H^TQ(Y_r - \bar{Y}) - R\bar{U})}^{\text{Unconstrained}}}_{\text{Preparation phase}} \underbrace{\overbrace{+M^T\lambda}^{\text{Constrained}} + H^TQG\delta x_k}_{\text{Feedback phase}}\right] \tag{15}$$

(ii) *The absolute solution is given as*

$$\hat{U} = -E^{-1}\left[\underbrace{\overbrace{-H^TQ(Y_r - \bar{Y} + H\bar{U})}^{\text{Unconstrained}}}_{\text{Preparation phase}} \underbrace{\overbrace{+M^T\lambda}^{\text{Constrained}} + H^TQG\delta x_k}_{\text{Feedback phase}}\right] \tag{16}$$

A general drawback of NMPC methods based on the RTI scheme is that the predictions can be subject to approximation errors given small deviation models are used and only one SQP iteration is done, i.e. re-linearising the system only once.

2.5 Blocked solutions

A popular method for reducing the computational burden further is by using blocked solutions, where the inputs or decision variables are blocked in sections and assumed to have the same value [16, 17, 26, 35]. This allows a reduction in the number of degrees of freedom and consequently the optimisation time. To achieve this, an equality constraint of block size N_B is embedded into the optimisation as $u_k = u_{k+1} = \cdots = u_{k+N_B-1}$ across all the prediction horizon. The latter can be represented by an input structure of the form

$$\delta\hat{U} = \mathbb{N}\delta\hat{\mathbb{U}} \tag{17}$$

$$\hat{U} = \mathbb{N}\hat{\mathbb{U}} \tag{18}$$

where $\hat{\mathbb{U}}$ (or $\delta\hat{\mathbb{U}}$) is the blocked decision variable and $\mathbb{N}$ is the blocking matrix defined as

$$\mathbb{N} = \begin{bmatrix} \mathbb{I} & \mathbb{O} & \cdots & \mathbb{O} \\ \mathbb{O} & \mathbb{I} & \cdots & \mathbb{O} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{O} & \mathbb{O} & \cdots & \mathbb{I} \end{bmatrix} \tag{19}$$

with dimensions $N_p n_u \times \lceil N_p/N_B \rceil n_u$ where the operator $\lceil x \rceil$ rounds the result towards infinity, $\mathbb{I}$ is a matrix containing N_B vertically blocked identity matrices of n_u dimension and $\mathbb{O}$ is a matrix of zeros of the same dimension. Obviously with $N_B = 1$, the blocking structure is just an identity and represents the same QP as (11) or (13). Substituting (17) and (18) into cost functions (11) and (13), respectively, leads to:

(i) *Blocked deviations formulation*

$$J = \frac{1}{2}(Y_r - \bar{Y} - G\delta x_k - H\mathbb{N}\delta\hat{\mathbb{U}})^TQ(Y_r - \bar{Y} - G\delta x_k - H\mathbb{N}\delta\hat{\mathbb{U}}) + \frac{1}{2}(\bar{U} + \delta\hat{\mathbb{U}})^TR(\bar{U} + \delta\hat{\mathbb{U}}) \tag{20}$$

(ii) *Blocked absolute formulation*

$$J = \frac{1}{2}(Y_r - \bar{Y} - G\delta x_k - H\mathbb{N}\hat{\mathbb{U}} + H\bar{U})^TQ (Y_r - \bar{Y} - G\delta x_k - H\mathbb{N}\hat{\mathbb{U}} + H\bar{U}) + \frac{1}{2}(\mathbb{N}\hat{\mathbb{U}})^TR(\mathbb{N}\hat{\mathbb{U}}) \tag{21}$$

Optimising w.r.t the decision variables, ($\delta\hat{\mathbb{U}}$) and ($\hat{\mathbb{U}}$), results in the modified Hessian (22) for both modified cost functions, (20) and (21), respectively,

$$E_{\mathbb{N}} = \mathbb{N}^T(H^TQH + R)\mathbb{N} = \mathbb{N}^TE\mathbb{N} \tag{22}$$

which has reduced dimensions of $N_{E_{\mathbb{N}}} = \lceil N_p/N_B \rceil n_u \times \lceil N_p/N_B \rceil n_u$, and the linear terms can be found to be, respectively:

(i) *Blocked deviations linear term*

$$f_{\mathbb{N}} = -\mathbb{N}^T(H^TQ(Y_r - \bar{Y} - G\delta x_k) - R\bar{U}) = -\mathbb{N}f \tag{23}$$

(ii) *Blocked absolute linear term*

$$f_{\mathbb{N}} = -\mathbb{N}^TH^TQ(Y_r - \bar{Y} - G\delta x_k + H\bar{U}) = -\mathbb{N}f \tag{24}$$

Finally, the constraint matrix M is modified to

$$M_{\mathbb{N}} = \begin{bmatrix} \mathbb{N}^T & -\mathbb{N}^T & (H\mathbb{N})^T & -(H\mathbb{N})^T \end{bmatrix}^{\mathrm{T}} \tag{25}$$

whereas the constraint vectors γ remain the same.

The modified Hessian $E_{\mathbb{N}}$, linear term $f_{\mathbb{N}}$ and constraint matrix $M_{\mathbb{N}}$ can be used to compress/decompress a pre-prepared QP, e.g. to use the strategy with a given toolkit such as ACADO.

Remark 3: $\mathbb{N}$ has N_B vertically blocked identity matrices, so the number of constraints related to the input can be reduced provided the respective rows in γ of a given vertically blocked section are equal. This is not the case when the solutions are based on deviations to a shifted blocked input trajectory (IVE strategy of RTI). In fact, unlike the unblocked case where both formulations (absolute or deviations) result in exactly the same solution, in this case, they will have different solutions because it is conceptually different to embed a blocked structure into either the deviations or absolute variables. However, if the shifting strategy proposed in this paper is used, it will be seen that they give the same results as a direct consequence of performing a consistent optimisation.

After the optimisation is solved, definitions (17) and (18) can be used to recover the solution in the original variables. This results in solutions (15) and (16) presented in the Section 2.4 to change to:

(i) *Blocked deviations solution*

$$\hat{U} = \bar{U} - \mathbb{N}E_{\mathbb{N}}^{-1}\mathbb{N}^T\left[\underbrace{\overbrace{-(H^TQ(Y_r - \bar{Y}) - R\bar{U})}^{\text{Unconstrained}}}_{\text{Preparation phase}} \underbrace{\overbrace{+M^T\lambda}^{\text{Constrained}} + H^TQG\delta x_k}_{\text{Feedback phase}}\right] \tag{26}$$

(ii) *Blocked absolute solution*

$$\hat{U} = -\mathbb{N}E_{\mathbb{N}}^{-1}\mathbb{N}^T\left[\underbrace{\overbrace{-H^TQ(Y_r - \bar{Y} + H\bar{U})}^{\text{Unconstrained}}}_{\text{Preparation phase}} \underbrace{\overbrace{+M^T\lambda}^{\text{Constrained}} + H^TQG\delta x_k}_{\text{Feedback phase}}\right] \tag{27}$$

One of the advantages of blocking is that the problem or system itself may require control actions in the future, and not all congested in the beginning of the horizon as with standard GPC approaches. This benefit can be seen in Fig. 1 where the predicted optimal trajectory of both approaches is given for the inverted pendulum problem presented in Section 5 and compared to the one using the full decision vector. Notice although they all present differences in the input solution, the solutions of position and angle trajectories are nearly indistinguishable for the blocked and full solutions, whereas the GPC solution clearly results in a different trajectory. As expected, the predicted costs of all cases, full decision, blocked and standard GPC were $J_{\text{full}} = 1872$, $J_{\text{blk}} = 1878$ and $J_{\text{std-gpc}} = 1994$, respectively, which clearly shows the superiority of blocking over the standard GPC approach. Obviously, the GPC solution would adjust the input as the horizon is moved forward (receding horizon) and might be able to perform similarly in closed-loop. However, it is the inconsistency/ill-posedness of the problem within each prediction that may negatively affect the overall closed-loop solution in the long term, especially when constraints come into play. This is discussed in Section 3.2.

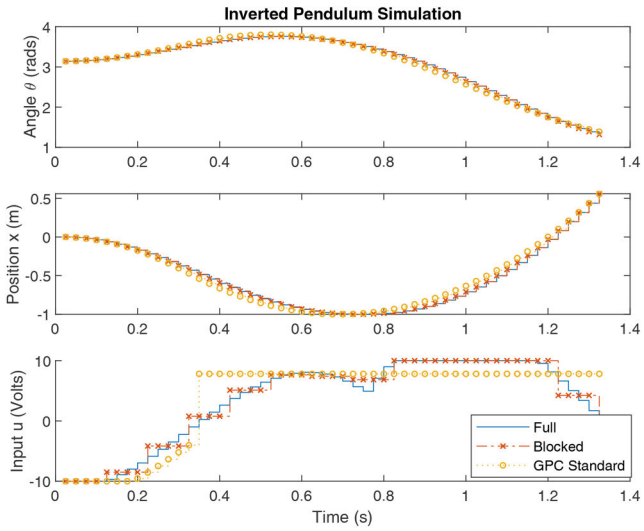


Fig. 1 *Blocking versus GPC standard comparison with* $N_B = 4$

3 Shifting strategy

This section presents the shifting strategy proposed in this paper which represents the main contribution and has the main goal of achieving faster computation times whilst preserving the stability and recursive feasibility properties of the standard RTI scheme.

3.1 Sub-strategies

The three sub-strategies summarised next are used in the proposed approach. These sub-strategies are explained further in the following subsections and an overall example is given in Section 3.6.

3.1.1 Reducing the number of degrees of freedom: The method uses an embedded input-structure, in this case blocked, for reducing the degrees of freedom of the optimisation. Other input-structures such as Laguerre or Kautz polynomials [15] were not considered, but represent potential alternatives, although they would present different recursive feasibility properties and overall performance in the general case.

3.1.2 Reducing the number of shooting points: It selects a reduced number of points of interest which sometimes are referred to as shooting points. These shooting points represent the constraints and future errors that are included in the optimisation, and are not necessarily at every sampling time but rather spread across the prediction horizon.

3.1.3 Absolute-time-frame shifting: It shifts the shooting points and blocked-inputs structure in an absolute-time-frame, rather than in a relative-time-frame (as implemented in standard NMPC/RTI methods) to maintain consistency along the optimisation.

3.2 Consistent optimisation and recursive feasibility

One of the most important properties to maintain in an optimisation is recursive feasibility [36]. This property is present in an optimisation *if and only if* for a given feasible solution at time $t = k$, all subsequent solutions at future times $t = k+1, k+2, \ldots, k+\infty$, remain feasible [26, 35]. At this point, it is emphasised that recursive feasibility is not related to how an initial feasible solution is found but rather, maintaining feasibility. In the case of a RTI scheme, it is typically assumed that the solution is initially close to an optimal and feasible solution for nominal stability [9], thus initial feasibility is implied. Moreover, strong recursive feasibility guarantees can be given when the optimisation explicitly includes the solution from the previous sampling time as a possible solution of the current optimisation, also known as the tail [30, 31]. This is a direct result of performing a consistent optimisation where at each time, the latter improves or 'builds on top of' the previous solution. This property is not naturally present in blocked solutions and instead, at each sampling time, the optimisation is forced to disregard the previous solution and find a new one which is the main reason they may lack recursive feasibility guarantees [15]. In other words, at each sampling time the optimiser makes a plan which is then immediately forced to disregard at the next sampling time.

3.3 Shifting strategy applied to blocked solutions

To solve the aforementioned problems, we propose the following blocked solution which shifts the blocks in an absolute-time-frame to maintain consistency in the 'breaking points' of the blocked input. A similar approach is presented in [26], however, an important difference with the proposed approach in this paper is that they do not apply it in the context of the RTI where the IVE approach is used, nor conceptualise it in an absolute-time-frame. Moreover, they do not formulate it in the context of NMPC for both types of solutions given above. Additionally, they use a time varying horizon, whereas in the proposed approach, the horizon is maintained constant through the use of the ideal horizon. Finally, although the approach has similarities with lifted systems [37], it has important differences given both measurements and control actions are available at the all times and the strategy is applied to reduce computation times whilst maintaining consistency and recursive feasibility.

In simple terms, the proposed approach applies a set of blocking structures sequentially, which guarantees that the previous solution (i.e. the tail) is always included in the optimisation. By using this method, the solution based on deviations can now be applied consistently as it now represents the exact same solution given by the absolute blocked formulation; this will be seen later in the results Section 5.2.

Definition 1 (Shifting blocked sections): The proposed strategy can be formally represented with the input equalities (28) and (29) below, defined for time steps $[k, k+N_B-1]$ with an horizon N_p, 'resetting' at time step $k+N_B$ and repeating infinitely

$$\begin{aligned} u_{k+i+(n-1)N_B|k+j} &= u_{k+nN_B-1|k+j} \\ \forall j = [0, N_B-1]; \quad \forall n &= \left[1, \left\lceil \frac{N_p - N_B + j}{N_B} \right\rceil\right] \\ \begin{cases} \forall i = [j, N_B-1] \text{if} n = 1 \\ \forall i = [0, N_B-1] \text{if} 1 < n < \left\lceil \dfrac{N_p - N_B + j}{N_B} \right\rceil + 1 \end{cases} \end{aligned} \tag{28}$$

and for the last block ($n = \lceil (N_p - N_B + j)/N_B \rceil + 1$),

$$\begin{aligned} u_{k+i+(n-1)N_B|k+j} &= u_{k+i_{\max}+(n-1)N_B|k+j} \\ \forall j = [0, N_B-1]; \quad \forall i &= [0, i_{\max}] \end{aligned} \tag{29}$$

with

$$i_{\max} = N_p + j - 1 - \left\lceil \frac{N_p - N_B + j}{N_B} \right\rceil N_B$$

where j is the time step, $j = 0$ giving the standard blocking structure, n is the number of blocked sections, $n = 1$ being the first one, and i is related to the size of the given blocked section. Notice the latter changes as $j \to N_B - 1$ and the number of blocks n depends on the selected prediction horizon. This will be explored further in the following subsections.

As a quick example of definition 1, consider a simple SISO optimisation with prediction horizon $N_p = 4$ and block size $N_B = 2$. The proposed strategy would apply the two following blocking matrices $\mathbb{N}_1$ and $\mathbb{N}_2$ sequentially, in order, and repeating infinitely $(\mathbb{N}_1, \mathbb{N}_2, \mathbb{N}_1, \ldots)$.

$$\mathbb{N}_1 = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}, \mathbb{N}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{30}$$

Lemma 1 (N_B unique blocking matrices): When using Definition 1 with a blocking size of N_B, there will always be exactly N_B unique blocking matrices ($\mathbb{N}$) that include the tail, maintaining consistency over time, hence preserving recursive feasibility.

Proof: Consider only the first blocked section $n = 1$ of a given blocking structure. By applying (28) to maintain consistency at any time step $k + j$ within that first block, the latter gives

$$\begin{gathered} u_{k+i|k+j} = u_{k+N_B-1|k+j} \\ \forall\, j = [0, N_B - 1]; \quad \forall\, i = [j, N_B - 1] \end{gathered} \tag{31}$$

Thus, the number of equalities i, i.e. the size of the first blocked section $N_{B_1} = N_B - j$, decreases as $j \to N_B - 1$ leading to N_B sizes for the latter and consequently N_B unique blocking structures $\mathbb{N}$. Conceptually, the latter shrinks until reaching its limit and 'resetting' its size to N_B as in $\mathbb{N}_1$ of example (30). □

3.3.1 Ideal prediction horizon: Notice in the example (30), the dimension of the resulting Hessian $E_{\mathbb{N}}$ would vary from $N_{E_{\mathbb{N}}} = 2 \times 2$ to $N_{E_{\mathbb{N}}} = 3 \times 3$. This is undesirable behaviour given that to achieve better computing performance, dynamic memory allocation should be avoided. A work around to this problem is to use an ideal prediction horizon which allows implementation of the proposed shifting strategy without modifying the dimension of the Hessian.

Lemma 2 (The ideal prediction horizon): When using Definition 1 to maintain recursive feasibility, the selected horizon N_p must be an integer multiple of the block size plus 1 to keep the Hessian dimension $N_{E_{\mathbb{N}}}$ constant for any block size N_B:

Proof: The expected size of the Hessian $N_{E_{\mathbb{N}}}$ for any block size N_B is given by

$$N_{E_{\mathbb{N}}} = \left\lceil \frac{N_p - N_{B_1}}{N_B} \right\rceil + 1 \tag{32}$$

where N_{B_1} is the length of the first blocked section ($n = 1$ of 28) of a given blocking structure. For constant dimensions, the following must hold:

$$\begin{gathered} \left\lceil \frac{N_p - N_B}{N_B} \right\rceil + 1 = \left\lceil \frac{N_p - i}{N_B} \right\rceil + 1 \\ \forall i = [1, N_B] \end{gathered} \tag{33}$$

The latter can only be satisfied for all i and any N_B by using $N_p = nN_B + 1$, where n is an integer number. Substituting in (33) gives

$$\left\lceil \frac{nN_B + 1 - N_B}{N_B} \right\rceil + 1 = \left\lceil \frac{nN_B + 1 - i}{N_B} \right\rceil + 1 \tag{34}$$

After some algebraic manipulation

$$n - 1 + \left\lceil \frac{1}{N_B} \right\rceil = n + \left\lceil \frac{1 - i}{N_B} \right\rceil \tag{35}$$

because $\lceil 1/N_B \rceil = 1$ and $\lceil (1 - i)/N_B \rceil = 0$ for all i, (35) holds. □

Algorithm 1: Ideal prediction horizon The following steps are advised for selecting the ideal prediction horizon.

1. Select the desired block size N_B.
2. Select a desired horizon $N_{p_{\text{des}}} > N_B$.
3. The closest ideal prediction horizon is then given by $N_p = \lceil N_{p_{\text{des}}}/N_B \rceil N_B + 1$ or $N_p = \lfloor N_{p_{\text{des}}}/N_B \rfloor N_B + 1$. For better stability properties, the upper one is suggested.

Theorem 1 (Shifting strategy applied to blocked solutions - recursive feasibility): When using Definition 1 with Lemmas 1 and 2, recursive feasibility guarantees are recovered by always including the tail.

Proof: Considering an optimal feasible solution with the blocked structure given by Definition 1 at time k ($j = 0$) with block size N_B:

$$\begin{gathered} u_{k+i+(n-1)N_B|k} = u_{k+nN_B-1|k} \\ \forall n = [1, \lceil \frac{N_p}{N_B} \rceil - 1]; \quad \forall i = [0, N_B - 1] \end{gathered} \tag{36}$$

and the last block containing a single input $u_{k+N_p-1|k}$. The tail of the solution will be automatically included at the next time step $k + 1$ ($j = 1$) giving

$$\begin{gathered} u_{k+i+(n-1)N_B|k+1} = u_{k+nN_B-1|k+1} \\ \forall n = [1, \lceil \frac{N_p}{N_B} \rceil - 1]; \quad \forall i = [1, N_B - 1] \end{gathered} \tag{37}$$

and the last block containing two blocked inputs, $u_{k+N_p-1|k+1} = u_{k+N_p|k+1}$. The same is true $\forall\, j = [0, N_B - 1]$. □

3.3.2 Breaking points: shifting Lagrange multipliers: An important concept in the proposed strategy is that of the 'breaking points'. As shown in Lemma 1, the first blocked section size N_{B_1} shrinks until it reaches its limit. When this happens, we refer to it as the 'breaking point' and the optimisation must apply a blocking matrix $\mathbb{N}$ where the first blocked section 'resets' and has the original block size $N_{B_1} = N_B$ such as $\mathbb{N}_1$ in (30). This is relevant when performing a constrained optimisation in the context of the RTI scheme for hot-started solutions, as an active-set guess can be provided.

Theorem 2 (Breaking points - shifting Lagrange multipliers): When using definition 1 with Lemma 2, the Lagrange multipliers (λ) active-set guess for hot started solutions must be shifted only when the optimisation reaches the 'breaking point' to maintain consistency.

Proof: Consider the following unblocked Lagrange multipliers related to positive input constraints ($M \leq u_{\max}$):

$$\lambda = \left[\lambda_{k|k}, \lambda_{k+1|k}, \ldots, \lambda_{k+N_p-1|k}\right] \tag{38}$$

The usual shifting strategy used by the RTI is

$$\begin{gathered} \lambda_{k+i|k+i} > 0 \text{ active if } \lambda_{k+i|k+i-1} > 0 \\ \lambda_{k+i|k+i} = 0 \text{ inactive if } \lambda_{k+i|k+i-1} = 0 \\ \forall i = [0, N_p - 1] \end{gathered} \tag{39}$$

Now for simplicity, consider an ideal prediction horizon $N_p = N_B + 1$. When the proposed shifting input structure given by Definition 1 is used, it requires only two lambdas $\left[\lambda_{1|k+j}, \lambda_{2|k+j}\right]$ for the two blocked sections where

$$\lambda_{1|k+j} = \lambda_{k+i|k+j} = \lambda_{k+N_B-1|k+j}$$
$$\forall\, j = [0, N_B - 1]; \quad \forall\, i = [j, N_B - 1] \tag{40}$$

for the first blocked section with initial block size N_B, and

$$\lambda_{2|k+j} = \lambda_{k+N_B+i|k+j} = \lambda_{k+N_B+j|k+j}$$
$$\forall j = [0, N_B - 1]; \quad \forall i = [0, j] \tag{41}$$

for the second blocked section. Applying the RTI IVE shifting (39) combined with equalities (40) and (41), and considering the 'breaking point' of definition (1) happens at $k + N_B$ gives

$$\lambda_{1|k+N_B} > 0 \text{ active if } \lambda_{2|k+N_{B-1}} > 0$$
$$\lambda_{1|k+N_B} = 0 \text{ inactive if } \lambda_{2|k+N_{B-1}} = 0 \tag{42}$$

at that time step. The same holds for the rest of the blocks where

$$\lambda_{i|k+N_B} > 0 \text{ active if } \lambda_{i+1|k+N_{B-1}} > 0$$
$$\lambda_{i|k+N_B} = 0 \text{ inactive if } \lambda_{i+1|k+N_{B-1}} = 0 \tag{43}$$
$$\forall i = [1, N_{E_{\mathbb{N}}}]$$

□

3.4 Shifting strategy applied to shooting points

The concept of a reduced number of points of interest is already used in the shooting methods, e.g. [3, 18], however, they do not apply the shifting strategy proposed in this paper where the points are kept in an absolute-time-frame. The proposed approach of this paper builds on top of the conceptual 'breaking points' aforementioned for the input-blocking and selects a subset of future errors and output constraints directly at the end of each blocked input for the optimisation.

Definition 2 (Shifting shooting points): The proposed approach can be formally represented by selecting a subset of references, outputs and output constraints given by

$$e_{k+nN_B|k+j} = r_{k+nN_B|k+j} - \hat{y}_{k+nN_B|k+j}$$
$$y_{\min} \le \hat{y}_{k+nN_B|k+j} \le y_{\max} \tag{44}$$
$$\forall n = [1, N_{E_{\mathbb{N}}} - 1]; \quad \forall j = [0, N_B - 1]$$

where j is the time step, n is related to number of shooting points, and the last point of the prediction horizon

$$e_{k+N_p+j|k+j} = r_{k+N_p+j|k+j} - \hat{y}_{k+N_p+j|k+j}$$
$$y_{\min} \le \hat{y}_{k+N_p+j|k+j} \le y_{\max} \tag{45}$$

is always included. To select the points at the end of each blocked input, the time step j must be 'in phase' with the time step j used by Definition 1.

Remark 4: Selecting a given subset of output errors or output constraints can be achieved by selecting the respective rows of matrices $H, G, Y_r, \bar{Y}, M, \gamma$.

Theorem 3 (Shooting points - recursive feasibility): The tail of the optimisation is automatically included by using Definition 2, giving recursive feasibility guarantees.

Proof: Consider an optimal solution for the shooting points:

$$\hat{r} = \left[r_{k+N_B|k}, r_{k+2N_B|k}, \ldots, r_{k+(N_{E_{\mathbb{N}}}-1)N_B|k}, r_{k+N_p|k}\right]$$
$$\hat{Y} = \left[\hat{y}_{k+N_B|k}, \hat{y}_{k+2N_B|k}, \ldots, \hat{y}_{k+(N_{E_{\mathbb{N}}}-1)N_B|k}, \hat{y}_{k+N_p|k}\right] \tag{46}$$

that satisfies the constraints $Y_{\min} \le \hat{Y} \le Y_{\max}$, i.e. is feasible at time k. By using (44) and (45) for time step $k+1\,(j=1)$, the optimisation will keep looking at the same output errors and constraints at all the points (i.e. the tail), except the last one.

$$\hat{r} = r_{k+N_B|k+1}, \ldots, r_{k+(N_{E_{\mathbb{N}}}-1)N_B|k+1}, r_{k+N_p+1|k+1}$$
$$\hat{Y} = \hat{y}_{k+N_B|k+1}, \ldots, \hat{y}_{k+(N_{E_{\mathbb{N}}}-1)N_B|k+1}, \hat{y}_{k+N_p+1|k+1} \tag{47}$$

The only difference from the aforementioned variables to be used for the optimisation is the error $e_{k+N_p+1|k+1} = r_{k+N_B+1|k+1} - \hat{y}_{k+N_p+1|k+1}$. If there were no reference changes and a sufficiently big horizon is used, this error would make a negligible change to the cost J and therefore, the optimisation would be able to follow the plan obtained at the previous time step k, provided the previous decisions can be replicated (i.e. the tail of the decision variables is available). It is noted that a rigorous guarantee requires invariant sets/terminal modes [31]. Nonetheless, works such as [2, 7, 8, 10, 20] have shown excellent performance without them, both in real systems and simulations. Moreover, because only one linearisation of the optimisation is performed in the RTI scheme, the constraint satisfaction will be subject to the accuracy of the linearisation process. This is a common problem for any RTI scheme variation.

Although this proof is derived by selecting shooting points at the conceptual 'breaking points', it is a general result and holds, independently of whether blocked approaches were used or not. In other words, if a non-blocked input-parameterisation was used, it would still guarantee recursive feasibility for the shooting points provided the tail of the decision variables is always available. Without the proposed shifting strategy, no recursive feasibility guarantee can be given without the use of soft-constraints, ie. slack variables, which would relax the feasibility problem entirely. □

Theorem 4 (Shooting points - shifting Lagrange multipliers): As in Theorem 2, the Lagrange multipliers (λ) must be shifted only when the optimisation reaches the 'breaking points' to maintain consistency. This holds even for non-blocked solution. This proof is similar to Theorem 2 thus is omitted.

3.5 Stability, optimality and convergence

As discussed in Remark 2, the stability of this scheme may be guaranteed with the use of zero-terminal constraints [6], or suitable terminal weights such as infinite horizon costing [20] which would make the resulting closed-loop sequence of costs to have Lyapunov stability. Indeed, the typical zero-terminal constraint proof can be seen directly in the assumption of Theorem's 3 proof where new information would add negligible terms to the cost and remain feasible. In the case of infinite horizon costing, a local LQR control law may be used to stabilise the system in the terminal region after N_u control actions as in [20], however in some cases, the state may not be able to get inside the terminal region in one SQP iteration as performed by the RTI Scheme. Moreover, the required assumptions of the RTI Scheme discussed in [10] must be considered to achieve global optimality, including that the optimisation is initialised at the global optimum, and that there are no abrupt reference or state jumps. Finally, as per all SQP methods, the convergence of the numerical solution may be subject to appropriate step-size selection (typically full for RTI [10]), and the accuracy of the linearisation process.

3.6 Example of the overall shifting strategy

To understand the overall strategy, consider the simple generic example given in (30) with the ideal prediction horizon $N_p = 5$, and the same block size $N_B = 2$. By dropping the absolute notation $k+i|k+j$ and applying the overall shifting strategy, the selected shooting points and constraints for both blocking structures expressed relative to the 'current' time step, would lead to considering cost function (9) subject to the two set of variables and constraints defined in Table 1, used in sequence and repeating infinitely (1st, 2nd, 1st, …) for the optimisation. The optimisation can then be prepared by selecting appropriate matrices and vectors for Y_r, G, H, M, γ.

Table 1 Shifting strategy example

Seq.	1st	2nd
refs.	$Y_r = \begin{bmatrix} r_{k+2} \\ r_{k+4} \\ r_{k+5} \end{bmatrix}$	$Y_r = \begin{bmatrix} r_{k+1} \\ r_{k+3} \\ r_{k+5} \end{bmatrix}$
outputs	$\hat{Y} = \begin{bmatrix} y_{k+2} \\ y_{k+4} \\ y_{k+5} \end{bmatrix}$	$\hat{Y} = \begin{bmatrix} y_{k+1} \\ y_{k+3} \\ y_{k+5} \end{bmatrix}$
input structure	$\hat{U} = \mathbb{N}_1\hat{\mathbb{U}}$ or $\delta\hat{U} = \mathbb{N}_1\delta\hat{\mathbb{U}}$	$\hat{U} = \mathbb{N}_2\hat{\mathbb{U}}$ or $\delta\hat{U} = \mathbb{N}_2\delta\hat{\mathbb{U}}$
	$\mathbb{N}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\mathbb{N}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$
constr.	$u_- \le \begin{bmatrix} u_k \\ u_{k+2} \\ u_{k+4} \end{bmatrix} \le u_+$	$u_- \le \begin{bmatrix} u_k \\ u_{k+1} \\ u_{k+3} \end{bmatrix} \le u_+$
	$y_- \le \begin{bmatrix} y_{k+2} \\ y_{k+4} \\ y_{k+5} \end{bmatrix} \le y_+$	$y_- \le \begin{bmatrix} y_{k+1} \\ y_{k+3} \\ y_{k+5} \end{bmatrix} \le y_+$

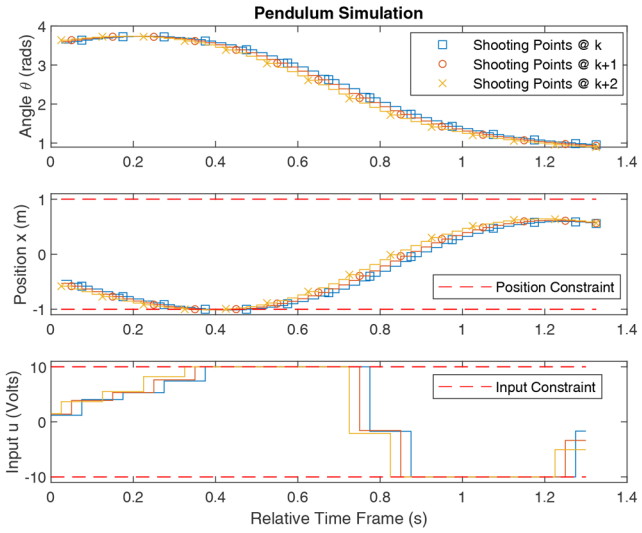


Fig. 2 *Shifting example with $N_B = 4$ starting from $j = 1$ of Definition 1, in relative time frames*

Remark 5: Notice the entire future input trajectory is constrained in the optimisation given the blocking input-structure used, however, the output constraints only constrain the shooting points. This will be seen in the results Section 5.2.

To give a comprehensive visualisation of the strategy, Fig. 2 shows the predicted trajectories in the relative time frame of three subsequent optimisation problems for the inverted pendulum problem presented in Section 5 when the strategy is using a block size of $N_B = 4$. It is noticeable how the shooting points at times approximately $0.2 \le t \le 0.4\,(s)$ are kept at the constraint limits. Moreover, notice how most of the time, the shooting points and the blocked inputs are moving left horizontally indicating that the resulting optimal of all three optimisations, in absolute time, are nearly identical.

4 Efficient coding

This section presents the algorithms and computational savings used to achieve efficient coding of the entire NMPC optimisation presented in Section 2 together with the proposed shifting strategy of this paper presented in Section 3.

Algorithm 2: Efficient H and G computation

In order to fill the H and G matrices efficiently, the following algorithm uses dummy variables T_G and T_H to calculate the required rows of the matrices recursively. Assuming $[T_G]_0 = [A_0]$ and $[T_H]_0 = [B_0]$ as starting points, it can be shown that all subsequent sub-matrices required by the H and G matrices are determined by

$$[T_G]_k = A_k[T_G]_{k-1}, [T_H]_k = \begin{bmatrix} A_k[T_H]_{k-1} & B_k \end{bmatrix} \quad \forall k = [1, \ldots, N_p - 1] \tag{48}$$

The values are then assigned into the corresponding rows and columns of H and G, and only the rows of the 'shooting points' are stored as discussed in Section 3. In the case the output has a non-linear relation to the state, matrices T_G and T_H must first be multiplied (separately) with the appropriate C_k (see (7) and (8)).

Once these matrices are filled, a significant number of computations can be avoided by taking advantage of the nature of the operations as in [7]. Furthermore, memory used should be pre-allocated avoiding dynamic memory allocation at all cost. Finally, it is important to NEVER repeat the same operation twice. Below we present a list of the computational savings that can be achieved.

1. The value of $H\mathbb{N}$ can be obtained by the summation of the respective columns in the H matrix. We will refer to this operation as $H\mathbb{N} = H_{\mathbb{N}}$.
2. The computation of the modified Hessian can be done as $E_{\mathbb{N}} = H_{\mathbb{N}}^T Q H_{\mathbb{N}} + \mathbb{N}^T R \mathbb{N}$.
3. The operation $\mathbb{N}^T R \mathbb{N}$ corresponds to the summation of penalisation values of the corresponding blocked inputs. This values can be gathered in a vector $r_{\mathbb{N}}$ and added directly to the diagonal.
4. The operation $\mathbb{N}^T R \bar{U}$ represents the summation of the inputs in the block multiplied by the respective penalisation.
5. Assuming Q is diagonal, the values on the diagonal (q) can be multiplied individually to the rows of $H_{\mathbb{N}}$ in $QH_{\mathbb{N}}$ operation of the Hessian [7]. We will refer to this operation as $H_{Q\mathbb{N}}$.
6. If Q is diagonal or symmetric, $Q^T = Q$, therefore $\mathbb{N}^T H^T Q = H_{Q\mathbb{N}}^T$.
7. Given the Hessian is symmetric, only the lower (or upper) triangular values need be calculated; the rest can be duplicated [7].
8. Given the Hessian is symmetric, a Cholesky decomposition is used to calculate the inverse of the Hessian efficiently by calculating only lower (or upper) triangular values and duplicating the rest.
9. An efficient version of Hildreth's QP provided [32] was developed, avoiding repeated computations by storing relevant results required by the optimisation.
10. The recovery of the original solution (full sized vector) from the blocked solution is done programmatically, rather than through (18) or (17).

Based on the ideas presented in this section, the proposed strategy can significantly reduce the memory required for the optimisation. In particular, this allows the reduction of matrix $E \to E_{\mathbb{N}}$, gradient $f \to f_{\mathbb{N}}$, constraint matrix $M \to M_{\mathbb{N}}$, prediction matrix $H \to H_{\mathbb{N}}$, as well as constraint vector γ, nominal output vector $\bar{Y}$ and state-to-output prediction matrix G by selecting only the rows related to the shooting points. However, if the methodology is meant to be used to compress/decompress a given QP, as explained in Section 2.5, the optimisation could add up to half the memory (depending on the block size – half at $N_B = 2$) for storing the compressed QP matrices $(E_{\mathbb{N}}, f_{\mathbb{N}}, M_{\mathbb{N}}, \gamma)$.

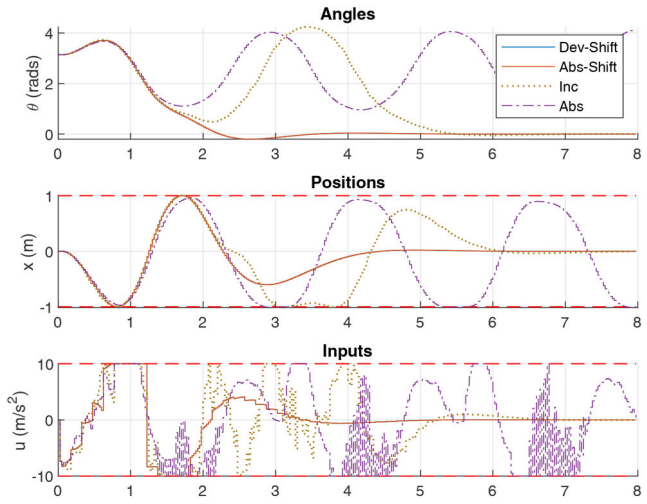


Fig. 3 *Example performance comparison with $N_B = 6$ using quadprog*

The ideas and computational savings gathered up to this section are implemented in the MATLAB and C++ codes given in [27, 28] for the benchmark presented in the following section.

5 Inverted pendulum – a benchmark

The inverted pendulum is a non-linear system widely used by academics, known to present several control challenges such as non-linear and non-minimum phase dynamics, physical constraints and under-actuation (multiple outputs–single input), where the task is to drive the pendulum to its upright position, and simultaneously control its position in a rail. This makes it an interesting and challenging benchmark for NMPC.

The application of NMPC to a real inverted pendulum was succesfully achieved in [34] using modest hardware. An impressive application to a real triple inverted pendulum is presented in [38] where a non-linear optimisation using a collocation point is to calculate the solution offline, however, they do not apply it in a receding horizon context nor do they apply it using the RTI scheme. Other authors have used it extensively for benchmark simulations such as [1, 6, 10, 13, 14, 20].

This section presents a benchmark of the proposed approach in a non-linear inverted pendulum for comparing the performance of the different strategies presented in this paper. The results show that by applying the proposed approach, the system achieves better performance and presents better recursive feasibility properties, which is a consequence of a consistent optimisation. Furthermore, it will be seen that the proposed approach can have computational gains up to 100 times faster on an i7 laptop, and up to 70 times faster on a relatively low power Linux-based embedded platform such as the aforementioned Beaglebone Blue, which would otherwise render the application of NMPC to this system unfeasible.

5.1 System modelling

Several variations of the mathematical model of an inverted pendulum have been used, some of which are more complex than others. For our simulation, we used the mathematical model presented in [33] which contains two main non-linearities, namely, the gravitational effect, $g\sin(\theta)$, and the non-linear torque-relationship $\cos(\theta)u$ of the bar-link with the input u (or car acceleration $\ddot{p} = u$). The model is given by the following differential equations:

$$\begin{bmatrix} \ddot{\theta} \\ \ddot{p} \end{bmatrix} = \begin{bmatrix} -b\dot{\theta} + g\sin\theta + \cos\theta u \\ u \end{bmatrix} \tag{49}$$

Assuming the state $x = \begin{bmatrix} \theta & \dot{\theta} & p & \dot{p} \end{bmatrix}^T$, the system was simulated using a forward Euler integration method. Considering θ and p as the relevant outputs leads to the following linearisation matrices of the state-space model (3):

$$A_k = \begin{bmatrix} 1 & T & 0 & 0 \\ T\alpha_k & (1 - Tb) & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix}, B_k = \begin{bmatrix} 0 \\ Tc\theta_k \\ 0 \\ T \end{bmatrix}$$
$$C_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{50}$$

where T is the sampling time, $c\theta_k = \cos\theta_k$, $s\theta_k = \sin\theta_k$ and $\alpha_k = gc\theta_k - s\theta_k u_k$. The parameters used were $b = 0.3$ and $g = 9.81\ \mathrm{m/s^2}$, and the simulation was done using a sampling time of $T = 0.025$ s. Only the position of the car and the angle were considered as outputs ($n_y = 2$) and the constraints of the system were considered as $-10 \le u \le 10\ (\mathrm{m/s^2})$ and $-1 \le p \le 1\ (m)$.

5.2 Simulations

All the simulation tests were done in the nominal case (no noise, no disturbances, no uncertainty) given that the prime interest is in the 'inner' recursive feasibility, stability properties and computational efficiency; disturbance rejection and noise cancellation can be addressed separately using offset-free optimisations [18, 39] and observer/estimator design or filters [30–32, 40], respectively. The simulation was initialised with the state $\boldsymbol{x} = [\pi \quad 0 \quad 0 \quad 0]$, and was run for $T_s = 8$ s, allowing the system to swing up in 'one shot' or 'two shots' (see Fig. 3). For the initial guess, a future nominal input trajectory of zeros $\bar{U} = \mathbb{O}$ was used which represents the free response of the system, a condition from which any optimisation could be initialised.

A desired prediction horizon of $N_{p_{\mathrm{des}}} = 50$ ($T_p = 1.25$ s) was selected and the ideal prediction horizon was then acquired depending on the selected block size (N_B). For a reference, the ideal horizon is displayed next to the block size in parenthesis in all the tables. Regarding the tuning parameters, the optimisation was done using $Q = I$ and an input penalisation of $R = 0.1I$. Moreover, a terminal cost (last two values in Q diagonal) of $Q_f = 500$ was used for both, angle and position, as a 'soft' zero-terminal constraint to improve the stability characteristics of the underlying optimisation.

5.2.1 Performance and recursive feasibility comparison: To assess the performance and recursive feasibility properties of the proposed approach, the system was tested with four possible types of solution (deviations/absolute with/without the proposed shifting strategy) and for different block sizes (N_B). Moreover, to compare the performance, two QP solvers were used in this comparison, namely the MATLAB R2018a quadprog function using the interior-point method and Hildreth's QP presented in [32], which is an active-set primal-dual type of QP that allows for hot-starting the solution (initial guess for λ). In the former, the solution did not have a limit in iterations or time (solved to optimallity), and the latter performed a fixed number of 20 iterations (approximated solution) when the unconstrained solution did not satisfy the constraints. In the particular case of non-shifted solution based on deviations, the constraints remained in the full-sized vector. Additionally, in the case of the MATLAB quadprog function, every time it returned an infeasibility flag, it was counted and the previous solution was used explicitly; this represents essentially open-loop control (no feedback), thus is a risk. The number of infeasibilities presented in a given type of solution is shown in brackets in Table 2. Finally, the solution of the optimisation was always saturated to respect the input constraints regardless of the result from the QP.

Table 2 gathers the comparison of the costs for all the different types of solutions where $J_{Dev-Shift}$ represents the cost of the deviations solution with the proposed shifting strategy, $J_{Abs-Shift}$ the cost of the absolute solution, and so on. For reference, costs less than 2000 swung up the system in 'one shot', costs greater

Table 2 Cost comparison for different block sizes N_B using shifting and non-shifting strategies, and using quadprog MATLAB function and Hildreth's QP for solving the optimisation

QP	Quadprog				Hildreth's			
N_B (N_p)	$J_{\text{Dev-Shift}}$	$J_{\text{Abs-Shift}}$	J_{Dev}	J_{Abs}	$J_{\text{Dev-Shift}}$	$J_{\text{Abs-Shift}}$	J_{Dev}	J_{Abs}
1 (50)	1101	1101	1101	1101	1102	1102	1102	1102
2 (51)	1099	1099	1105 [17]	1119 [8]	1102	1102	1108	1130
3 (52)	1104	1104	2732 [18]	1239	1103	1103	1114	1238
4 (53)	1138	1138	2758 [32]	2475	1140	1140	1135	2787
5 (51)	1194	1194	1224 [14]	4118	1244	1244	1218	3952
6 (55)	1168	1168	2436 [18]	4002	1177	1177	1237	3715
7 (57)	1098	1098	2438 [26]	2197	1098	1098	1245	2201
8 (57)	1183	1183	2533 [38]	2173	1192	1192	1285	2293
9 (55)	1207	1207	2361 [20]	3514	1220	1220	2697	2230
10 (51)	2524	2524	2544 [27]	3475	2555	2555	2448	3474
11 (56)	2776	2776	2284 [15]	3448	2714	2714	4392	3447
12 (61)	1244	1244	4432 [14]	3397	1247	1247	2203	3399
total	16,835	16,835	27,948 [239]	32,258 [8]	16,894	16,894	21,185	30,969

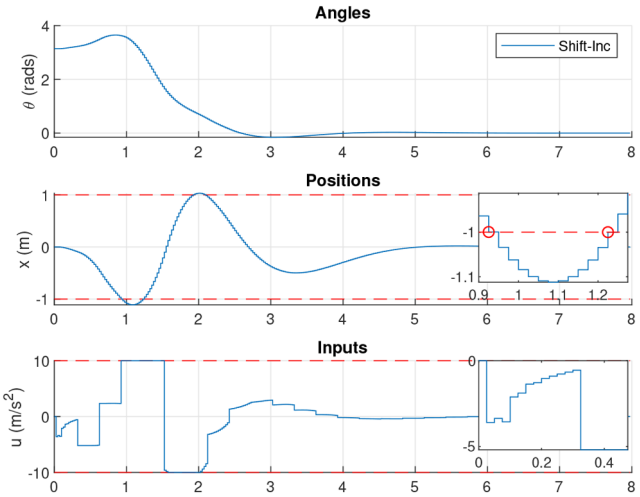


Fig. 4 *Shifting strategy response with $N_B = 12$ using Hildreth's QP*

than 2000 but less than 3000 swung up the system in 'two shots', and costs above 3000 means the optimisation was not able to stabilise the system (see Fig. 3). The following summarises the main results from Table 2.

1. In both QPs (quadprog and Hildreth's), the results from the proposed deviation and absolute formulations with the shifting strategy are exactly the same (columns 1 and 2, and columns 5 and 6 equal). This is a direct result of what has been said repeatedly throughout the paper: consistency. Moreover, no infeasibilities were recorded for both types of solutions when using quadprog.
2. The solution giving deviations without shifting presented a significant number of infeasibilities (239), and worse performance when solving to optimality (quadprog) than when using an approximate solution (Hildreth's). This is a direct result of inconsistency combined with the fact that Hildreth's does not check for infeasibility and therefore feedback is always applied.
3. Although there were some differences in the results between both QP's for the proposed approach (i.e. columns 1 and 2 $\neq$ columns 5 and 6), most likely given that Hildreth's QP did not converge to the solution in the 20 iterations (slow convergence rate of λ [32]), they gave similar results for all the cases (columns 1 and 2 $\simeq$ 5 and 6).
4. Overall, the best 'Total' cost is given by the proposed shifting strategy and the absolute non-shifted formulation gave the worse results.
5. For block sizes $N_B = 10, 11$, non of the solutions was able to swing up the system in 'one shot'. This is unsurprising and linked to the obvious observation that there are sensible block sizes.
6. In both QP's, suboptimalities $\Delta J = (J_{N_B}/J_1 - 1) \times 100 < 13.26\%$ where obtained through all the 'one shot' solutions which give acceptable performance such as the ones given in both Figs. 3 and 4.
7. Notice block sizes $N_B = 2, 7$ presented even better performance than the original full size vector $N_B = 1$. This is because in the linearisation process, the optimisation might take a different 'branch' of the solution that improved further AFTER re-linearisation. Moreover, allowing the intermediate constraints to be violated may relax the solution and lead to better performance at the cost of having to accept the violations. Finally, the optimisation is done in a finite horizon where both block sizes have slightly longer prediction horizon which could result in better overall predictions.

Fig. 3 shows an example response with block size $N_B = 6$ where it can be seen that the optimisation was able to swing up and stabilise the system in 'one shot' using the proposed shifting strategy (both deviations/absolute giving same result). In contrast, it took 'two shots' for the non-shifted solution based on deviations and the optimisation failed completely in the case of the absolute non-shifted solution.

Another value that was compared was the summation of the absolute violation to the position constraints for different block sizes (N_B). In other words

$$\sum_{\forall k} v_k, \quad \text{where } v_k = \begin{cases} |p_k| - 1 \text{ if } |p_k| > 1 \\ 0 \text{ else} \end{cases} \tag{51}$$

The results of this are gathered in Table 3, where V_T-Shift represents the total violation of the proposed shifting strategy (deviations and absolute are the same), and V_{Dev} and V_{Abs} the total violation of the deviation and absolute non-shifting solutions, respectively. Additionally, given that the proposed shifting strategy is only supposed to enforce the constraints in the shooting points, the summation of the constraint violation at this particular points was stored separately and is represented by $V_{S-\text{Shift}}$ in the table. The following summarise the main results from Table 3.

1. In the full optimisation case (quadprog), there were no violations of the constraints at the shooting points ($V_{S-Shift} = 0$) when using the proposed strategy.
2. In the approximated optimisation case (Hildreth's), only 3 significantly small (1 mm) violations occurred on the shooting points. Notice as the block size increases, the number of constraints in the optimisation is reduced. This ultimately allows the QP to find the active set in less iterations, resulting in no violations on the shooting points at bigger block sizes whilst performing slightly better because of the relaxation of the intermediate constraints.
3. Although the non-shifted solutions gave presumably 'good' results for the Hildreth's case, they present significant cost suboptimalities. Moreover, the non-shifted deviation solution

requires the full decision vector (not the blocked vector) to be constrained, thus removing part of the computational benefit.

To illustrate the concept of satisfying the constraints in the shooting points, Fig. 4 shows the response of the system with block size $N_B = 12$ where it can clearly be seen that the solution satisfies the constraints (at the very limits) at the shooting points, which in this case are at times $t = [0.925, 1.225] = [3N_B + 1, 4N_B + 1]T$. The 'extra step' in both shooting points is due to the computation separation strategy of the RTI which uses a predicted state, thus always optimising relative to 'one step ahead' and applying the feedback phase in the next sampling time when the measurement of the state is available. This can clearly be seen in the input response where the first decision is at $t = 0.025$ s instead of $t = 0$ s. Another important thing to notice is that the solution clearly exhibits the blocking structure, in particular, after $t > 3$ when the system is stabilised within the prediction horizon. Finally, a particular drawback of the proposed approach is that it only guarantees satisfying constraints at the shooting points by fixating all attention to them, therefore a small slack is required to protect the non-shooting points from small constraint violations. The selection of the slack size itself is a non-trivial task but could be selected based on Monte Carlo simulations, analysing the system from a variety of conditions, and obviously would be increased as the block size increases. In general, this is a problem from which most direct methods suffer because of the discretisation of the problem and thus is not unique to the proposal in this paper.

Table 3 Constraint violation comparison for different block sizes N_B, using shifting and non-Shifting strategies, and quadprog MATLAB function and Hildreth's QP for solving the optimisation

QP	quadprog				Hildreth's			
N_B (N_p)	$V_{T\text{-Shift}}$	$V_{S\text{-Shift}}$	V_{Dev}	V_{Abs}	$V_{T\text{-Shift}}$	$V_{S\text{-Shift}}$	V_{Dev}	V_{Abs}
1 (50)	0	0	0	0	0	0	0	0
2 (51)	0.002	0	0.004	0.006	0.002	0.001	0.002	0
3 (52)	0.009	0	0.017	0	0.008	0	0.010	0
4 (53)	0.002	0	0.064	0	0.002	0.001	0.012	0
5 (51)	0	0	0.044	0.106	0	0	0.030	0.040
6 (55)	0.003	0	0.094	0.178	0.003	0	0.040	0.301
7 (57)	0.171	0	0.300	0	0.171	0	0.089	0
8 (57)	0.114	0	0.423	0	0.108	0.001	0.021	0.094
9 (55)	0	0	0.406	0.059	0	0	0.100	0.008
10 (51)	0.032	0	0.686	0	0.026	0	0.334	0
11 (56)	0.069	0	0.313	0	0.156	0	0.214	0
12 (61)	1.055	0	0.986	0	1.033	0	0.055	0
Total	1.457	0	3.337	0.349	1.511	0.003	0.908	0.444

5.2.2 Computation time comparison: A core topic of interest is the computation time benefits achieved by the proposed strategy. Notice that the latter benefits from two main parts: (i) the first being the reduction in the number of degrees of freedom and points of interest which by itself would lend to faster unconstrained solutions, and (ii) the second being the reduction of the number of constraints which would otherwise lack the recursive feasibility guarantees if they were not reduced in the absolute-time-frame used by the proposed approach.

In order to properly test the computation times, the strategy was tailored to be executed with a given block size N_B and its ideal prediction horizon N_p which define the sizes of the matrices to be used. Each block size was programmed separately as a MATLAB function, and MATLAB C Coder was used to produce tailored C++ code to perform the optimisation. Regarding the QP iterations, the interest was on how fast the prepared QPs could be run, therefore the optimisation performed exactly 20 steps of Hildreth's QP when the unconstrained solution did not satisfy the constraints, independently of whether the active set was found or not. This is a realistic scenario given a situation may arise in a real system where the optimisation can only run for a given maximum number of iterations. Three execution times were of interest, namely:

1. *Preparation:* The total time required to compute QP matrices $E_{\mathbb{N}}, f_{\mathbb{N}}, M_{\mathbb{N}}, \gamma$; referred to as QP_p in Table 4.
2. *Unconstrained solution:* The preparation time plus obtaining the unconstrained solution and computing the feedback gain $K_x = E_{\mathbb{N}}^{-1} H_{\mathbb{N}Q}^T G$ to be used in the feedback phase; referred to as QP_u in Table 4.
3. *Constrained solution:* The unconstrained solution time plus the time required to perform exactly 20 QP iterations of Hildreth's QP; referred to as QP_c in Table 4.

The resulting C++ code can be found in [28] and was tested on two different systems: a laptop running Ubuntu 18.04 with an i7-5700HQ 64-bits processor running @3.5 GHz with 12 GB of DDR3 RAM @ 1.6 GHz; and a Beaglebone Blue embedded platform running Real-Time (RT) Debian with an ARM Cortex-A8 32-bits processor running @ 1 GHz. Given that the latter is a 32-bit system, the code was also produced and tested in 32-bits format on both systems. The simulation was done 1000 times and the minimum execution times for all cases were stored. This represents the fastest computation time that the approaches could obtain if a real time OS was used given the exact same computations/QP

Table 4 Comparison of computation times (in microseconds) for different block sizes N_B in 32/64 bit formats on two different systems: Ubuntu 18.04 (Intel i7-5700HQ 64-bit @ 3.5 GHz) & Beaglebone Blue running RT Debian (ARM Cortex-A8 32-bit @ 1 GHz)

Sys	Ubuntu 18.04 (i7-5700HQ @ 3.5 GHz)						Beaglebone Blue (ARM Cortex-A8 @ 1 GHz)					
Bits	64-bits			32-bits			64-bits			32-bits		
N_B (N_p)	QP_p	QP_u	QP_c	QP_p	QP_u	QP_c	QP_p	QP_u	QP_c	QP_p	QP_u	QP_c
1 (50)	165	276	2646	144	248	2291	6393	14326	108567	6689	14879	119865
2 (51)	41	60	427	35	53	415	2239	3459	19800	2144	3359	21586
3 (52)	26	34	185	22	30	172	1756	2204	8568	1598	2043	9211
4 (53)	21	25	85	18	22	77	1628	1863	5250	1456	1680	5100
5 (51)	17	19	57	15	17	52	1440	1575	3496	1372	1493	3493
6 (55)	19	21	52	16	17	46	1619	1729	3252	1390	1487	3077
7 (57)	19	21	45	17	18	41	1706	1794	2962	1655	1734	2996
8 (57)	18	19	39	16	17	35	1693	1761	2634	1426	1492	2439
9 (55)	17	18	33	15	15	29	1572	1625	2306	1447	1495	2235
10 (51)	14	15	26	12	13	23	1312	1352	1860	1146	1182	1717
11 (56)	17	18	29	14	15	26	1552	1593	2105	1366	1403	1938
12 (61)	20	20	32	17	18	29	1819	1861	2357	1602	1639	2172
gain	12	18	102	12	19	100	5	11	58	6	13	70

iterations are performed. The resulting computing times are gathered in Table 4 and the following summarises the main results.

1. For the constrained solution, the optimisation was able to get computation times QP_c up to 102 times faster in the laptop when using 64-bits, and 100 times faster when using 32-bits format. In the case of the embedded system (Beaglebone Blue), the optimisation was able to get computation time QP_c up to 70 times faster when using 32-bits, and only 58 times faster when using 64-bits.
2. Gains of up to 19 and 13 times faster were observed for the unconstrained solution, and up to 12 and 6 times faster for the preparation in the laptop and the embedded system, respectively.
3. The fastest execution time in all the cases is obtained with block size $N_B = 10$. This is because this block size, has the smallest ideal prediction horizon ($N_p = 51$) with the smallest Hessian size of 6×6 ($\lceil \frac{51}{10} \rceil$) and only 24 constraints (6×4). In contrast, the optimisation with block size $N_B = 1$ has a Hessian size of 50×50 and 200 constraints (50×4).
4. The block size of $N_B = 1$ would render the application unfeasible in the embedded system (Beaglebone Blue) given that the optimisation would not be able to finish within a sampling time ($T = 0.025$ s). However, in all the cases the computation time quickly drops to less than 8% with a block size of $N_B = 3$.

6 Conclusion

This paper presents a novel shifting strategy based on efficient blocked solutions for NMPC combined with the RTI scheme. The proposed strategy uses a set of blocking structures which if applied sequentially automatically include the tail of the solution hence preserving recursive feasibility guarantees whilst reducing the degrees of freedom and the input-related constraints. Additionally, the proposed approach uses a reduced amount of points of interest, sometimes called shooting points, which represent output errors and output constraints that must be satisfied, and a stability and recursive feasibility guarantee is presented for the infinite horizon case, or for when the system includes special terminal conditions such as zero-terminal constraints or infinite horizon costing. Finally, it presents a set of algorithms and computational savings that can be used to code the proposed approach efficiently.

The overall resulting strategy is tested using an inverted pendulum as a benchmark where the proposed approach clearly outperforms the standard solutions in recursive feasibility properties and general performance, giving suboptimallities $\Delta J < 13\%$ and fully satisfying the constraints at the shooting points when a full optimisation is performed. Finally, the computational benefits of the proposed approach were evaluated in two physical systems: an i7 laptop and a Beaglebone Blue embedded system, where computation times up to 100 and 70 times faster were possible.

Future work related to the proposed strategy will be to merge the approach with the ACADO toolkit allowing it to use the efficient sensitivity generation and integration methods it contains, as well as a variety of QP solvers, and ultimately, the automatic code generation for its implementation in real robotic and mechatronic systems such as UAVs using, for example, the Beaglebone Blue Linux-based computing platform.

7 Acknowledgment

This work was funded by CONACyT, Mexico.

8 References

[1] Quirynen, R., Gros, S., Diehl, M.: 'Efficient NMPC for nonlinear models with linear subsystems'. IEEE Conf. on Decision and Control, Florence, Italy, 2013, pp. 5101–5106

[2] Seki, H., Ooyama, S., Ogawa, M.: 'Nonlinear model predictive control using sucessive linearization - application to chemical reactors', *Trans. Soc. Instrum. Control Eng.*, 2004, **E-3**, (1), pp. 66–72

[3] Zannelli, A., Horn, G., Frison, G., *et al.*: 'Nonlinear model predictive control of a human-sized quadrotor'. European Control Conf., Limassol, Cyprus, 2018, vol. 16, no. 1, pp. 41–50

[4] Mei, Q., Xu, F., Chen, H., *et al.*: 'Fast model predictive control based on multiscale system theory'. World Congress on Intelligent Control and Automation, Guilin, China, 2016, pp. 2517–2522

[5] Diehl, M., Bock, H.G., Schlöder, J.P.: 'A real-time iteration scheme for nonlinear optimization in optimal feedback control', *SIAM J. Control Optim.*, 2005, **43**, (5), pp. 1714–1736

[6] Houska, B., Ferreau, H.J., Diehl, M.: 'An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range', *Automatica*, 2011, **47**, (10), pp. 2279–2285

[7] Vukov, M., Domahidi, A., Ferreau, H.J., *et al.*: 'Auto-generated algorithms for nonlinear model predictive control on long and on short horizons'. IEEE Conf. on Decision and Control, Florence, Italy, 2013, pp. 5113–5118

[8] Gros, S., Quirynen, R., Diehl, M.: 'Aircraft control based on fast non-linear MPC & multiple-shooting'. IEEE Conf. on Decision and Control, Maui, Hawaii, USA, 2012, no. 1, pp. 1142–1147

[9] Diehl, M., Findeisen, R., Allgöwer, F., *et al.*: 'Nominal stability of real-time iteration scheme for nonlinear model predictive control', *IEE Proc. Control Theory Applic.*, 2005, **152**, (3), pp. 296–308

[10] Gros, S., Zanon, M., Quirynen, R., *et al.*: 'From linear to nonlinear MPC: bridging the gap via the real-time iteration', *Int. J. Control*, **7179**, 2016, pp. 1–19

[11] Guarantees, F., Quirynen, R., Diehl, M., *et al.*: 'An efficient inexact NMPC scheme with stability and feasibility guarantees', *IFAC-PapersOnLine*, 2016, **49**, (18), pp. 53–58

[12] Wirsching, L., Albersmeyer, J., Kühl, P.: 'An adjoint-based numerical method for fast nonlinear model predictive control', *IFAC Proc. Vol.*, 2008, **41**, (2), pp. 1934–1939

[13] Chen, Y., Cuccato, D., Bruschetta, M., *et al.*: 'An inexact sensitivity updating scheme for fast nonlinear model predictive control based on a curvature-like measure of nonlinearity'. IEEE Conf. on Decision and Control, Melbourne, Australia, 2017, pp. 4382–4387

[14] Zanelli, A., Quirynen, R., Frison, G., *et al.*: 'A partially tightened real-time iteration scheme for nonlinear model predictive control'. IEEE Conf. on Decision and Control, Melbourne, Australia, 2017, vol. 1, no. 1

[15] Rossiter, J., Wang, L., Valencia-Palomo, G.: 'Efficient algorithms for trading off feasibility and performance in predictive control', *Int. J. Control*, 2010, **83**, pp. 789–797

[16] Kouzoupis, D., Quirynen, R., Houska, B., *et al.*: 'A block based ALADIN scheme for highly parallelizable direct optimal control'. American Control Conf., Boston, MA, USA, 2016, vol. 2016, pp. 1124–1129

[17] Kouzoupis, D., Quirynen, R., Frasch, J.V., *et al.*: 'Block condensing for fast nonlinear MPC with the dual newton strategy', *IFAC-PapersOnLine*, 2015, **48**, (23), pp. 26–31

[18] Huang, R., Biegler, L.T., Patwardhan, S.C.: 'Fast offset-free nonlinear model predictive control based on moving horizon estimation', *Ind. Eng. Chem. Res.*, 2010, **49**, (17), pp. 7882–7890

[19] Houska, B., Ferreau, H.J., Diehl, M.: 'ACADO toolkit-an open-source framework for automatic control and dynamic optimization', *Opt. Control Appl. Methods*, 2011, **32**, (3), pp. 298–312

[20] Houska, B., Ferreau, H.J., Diehl, M.: 'Autogenerating microsecond solvers for nonlinear MPC: a tutorial using ACADO integrators', *Optim. Control Appl. Methods*, 2015, **36**, pp. 685–704

[21] Kalmari, J., Backman, J., Visala, A.: 'A toolkit for nonlinear model predictive control using gradient projection and code generation', *Control Eng. Pract.*, 2015, **39**, pp. 56–66

[22] Ferreau, H.J., Kirches, C., Potschka, A., *et al.*: 'qpOASES: a parametric active-set algorithm for quadratic programming', *Math. Program. Comput.*, 2014, **6**, (4), pp. 327–363

[23] Diehl, M., Ferreau, H.J., Bock, H.G.: 'An online active set strategy to overcome the limitations of explicit MPC', *Int. J. Robust Nonlinear Control*, 2014, **18**, pp. 816–830

[24] Quirynen, R., Vukov, M., Diehl, M.: 'Multiple shooting in a microsecond' (*Multiple shooting and time domain decomposition methods*', vol. 9 (Springer, Cham, 2015), pp. 183–202

[25] Shen, C., Buckham, B., Shi, Y.: 'Modified C/GMRES algorithm for fast nonlinear model predictive tracking control of AUVs', *IEEE Trans. Control Syst. Technol.*, 2017, **25**, (5), pp. 1896–1904

[26] Cagienard, R., Grieder, P., Kerrigan, E.C., *et al.*: 'Move blocking strategies in receding horizon control', *J. Process Control*, 2007, **17**, (6), pp. 563–570

[27] Villarreal, O.J.G.: 'Code, MATLAB, for A Shifting Strategy for Efficient Block-based Nonlinear Model Predictive Control Using Real Time Iterations'. Available at https://doi.org/10.24433/CO.0e3ceef1-9d0d-4bbc-b3ff-9fe6213ebc12, 2018

[28] Villarreal, O.J.G.: 'C++ Code for 'A Shifting Strategy for Efficient Block-based Nonlinear Model Predictive Control Using Real Time Iterations'. Available at https://doi.org/10.24433/CO.5d68cc1d-237e-4440-8e0e-dad909605e3f, 2018

[29] Beagleboard.org, Beaglebone Blue. Available at https://beagleboard.org/blue, 2017

[30] Rossiter, J.A.: '*Model-based predictive control: a practical approach*', Control series' (CRC Press, Boca Raton, 2003)

[31] Rossiter, J.A.: '*A first course in predictive control*' (CRC Press, Taylor & Francis, Boca Raton, 2018, 2nd edn.)

[32] Wang, L.: '*Model predictive control system design and implementation using matlab*' (Springer, London, 2009)

[33] Alamir, M.: 'Fast NMPC: a reality-steered paradigm: key properties of fast NMPC algorithms'. European Control Conf., Strasbourg, France, 2014, no. 4, pp. 2472–2477

[34] Mills, A., Wills, A., Ninness, B.: 'Nonlinear model predictive control of an inverted pendulum'. American Control Conf., St. Louis, MO, USA, 2009, pp. 2335–2340
[35] Shekhar, R.C., Manzie, C.: 'Optimal move blocking strategies for model predictive control', *Automatica*, 2015, **61**, pp. 27–34
[36] Ong, C.J., Wang, Z., Dehghan, M.: 'Model predictive control for switching systems with dwell-time restriction', *IEEE Trans. Autom. Control*, 2016, **61**, (12), pp. 4189–4195
[37] Rossiter, J.A., Sheng, J., Chen, T.*, et al.*: 'Interpretations of and options in dual-rate predictive control', *J. Process Control*, 2005, **15**, (2), pp. 135–148
[38] Glück, T., Eder, A., Kugi, A.: 'Swing-up control of a triple pendulum on a cart with experimental validation', *Automatica*, 2013, **49**, (3), pp. 801–808
[39] Huusom, J., Poulsen, N., Jorgensen, S.*, et al.*: 'Tuning of methods for offset free MPC based on ARX model representations'. American Control Conf., Baltimore, MD, USA, 2010, pp. 2355–2360
[40] Camacho, E.F.: '*Model predictive control. advanced textbooks in control and signal processing*' (Springer, London, New York, 2003, 2nd edn.)

IET Control Theory & Applications

Research Article

IET Journals
The Institution of Engineering and Technology

Fast hybrid dual mode NMPC for a parallel double inverted pendulum with experimental validation

ISSN 1751-8644
Received on 30th January 2020
Revised 28th April 2020
Accepted on 29th June 2020
E-First on 9th September 2020
doi: 10.1049/iet-cta.2020.0130
www.ietdl.org

Oscar Gonzalez[1] ✉, *Anthony Rossiter*[1]
[1]*ACSE, The University of Sheffield, Sheffield, UK*
✉ *E-mail: ojgonzalezvillarreal1@sheffield.ac.uk*

Abstract: This study presents a novel fast non-linear model predictive control approach for a parallel double inverted pendulum. The approach uses dual-mode closed-loop predictions to obtain numerically robust optimal solutions. Moreover, it uses the real-time iteration (RTI) scheme to reduce the computational burden and achieve real-time performance. Furthermore, two main modifications are proposed which significantly improve the performance of the RTI scheme in the presence of large disturbances, namely; additional energy-based costs, and a hybrid switching scheme. Finally, the approach uses a non-standard discretised model combined with an online system identification scheme to address parameter uncertainty, and with an extended Kalman filter for state-estimation. The resulting performance is validated through both simulations and experimental results .

1 Introduction

Non-linear model predictive control (NMPC) is an advanced optimal control strategy able to handle complex and constrained non-linear dynamic systems [1]. Although NMPC has a long history of development, its deployment had been restricted to the process industry where relatively slow processes allowed time to compute the required control algorithms [2]. However, recent progress in computing performance has enabled its application in many systems through the use of efficient solutions [1, 3, 4]. One of the most successful and popular approaches for fast NMPC is the real-time iteration (RTI) scheme, originally developed in [5]. Its efficiency is based on the fact that NMPC is required to successively solve optimal control problems (OCPs) which are closely linked to each other [1]. An excellent tutorial-like paper detailing the main differences between the RTI NMPC and standard NMPC is given in [1]. Moreover, the efficiency of the overall approach depends largely on how the algorithms are programmed, as well as the platforms in which they are deployed, e.g. using embedded hardware such as field-programmable gate-arrays [6]. To address this, several toolkits containing efficient autogeneration routines are available such as the ACADO toolkit [3], VIATOC and CasADi [7], to name a few. Furthermore, the underlying optimisation may be solved using simultaneous or sequential approaches which lead to sparse or condensed OCPs. Authors in [4] concluded condensing-based approaches are faster for small to medium systems, whereas sparse solutions give a better overall performance for large scale optimisations and deal better with unstable systems [8]. Finally, direct methods are commonly used to discretise the problem, typically by using multiple or single-shooting discretisations [8, 9].

The (double) inverted pendulum is a complex multivariable non-linear system that presents many challenges such as input-output constraints as well as underactuated, unstable and non-minimum phase dynamics [10]. For this reason, it has been used extensively for benchmarking of NMPC, though mostly for simulation works such as [1, 3, 7], and similar systems such as cranes studied in [6]. Nonetheless, experimental contributions have been achieved in [10–12], and furthermore discussed within. In [10], a triple pendulum swing up was achieved by using a two-degrees of freedom control structure which used offline optimisation to compute a feedforward trajectory, and a feedback controller to stabilise the system along it. In [11], a fast NMPC scheme for a twin parallel pendulum was developed which used a control-parameterisation where the decision variable was able to take only 3 possible values. Finally, authors from [12] presented an NMPC for a single inverted pendulum.

In this paper, we propose a novel condensed single-shooting Dual Mode NMPC based on the RTI Scheme for a parallel double inverted pendulum, which differs from all previous works such as [1, 3, 7, 10–12] for the general inverted pendulum problem. The proposed approach cancels the unstable dynamics of the inverted pendulum through the use of closed-loop predictions [13] giving solutions with up to 6 orders of magnitude better numeric conditioning than the standard NMPC. This can be critical for matrix inversion when using low precision computing, and is something that was found to be ignored in all the works found in the literature. Indeed, no current toolkit offers support to address this for condensing based NMPC optimisation. On the other hand, we propose two main modifications to improve the performance of the RTI Scheme in the presence of large disturbances, namely; additional energy-related costs, and a hybrid switching scheme. These modifications were observed to produce much smoother responses than the standard NMPC. Finally, the entire scheme uses a non-standard discretised model which is combined with an online system identification (OSI) algorithm based on recursive least squares (RLS) and delta modelling approaches, to address parameter uncertainty. The whole approach is validated through both simulations and experimental results. The benefits when compared to [10, 11] were faster performance, and the use of online optimisations, thus allowing large disturbances and model updates. A video of the resulting performance can be found at (https://youtu.be/7E-SXi3YKQohttps://youtu.be/7E-SXi3YKQo), and the data of the experiments are available in https://doi.org/10.24433/CO.8048147.v1 [14] along with a C++ code implementing the approach using the EIGEN library.

The paper is organised as follows: Section 2 introduces the mathematical models of the pendulum along with the proposed discretisation scheme. Section 3 presents a detailed derivation of the proposed dual-mode fast NMPC approach based on the RTI Scheme along with the two aforementioned modifications required to improve the RTI performance, which overall represents the main contribution of the paper. Sections 4 and 5 present background on the extended Kalman filter (EKF) and OSI frameworks used for this work without a detailed derivation as both are well known in the literature. Section 6 presents the details of the experimental setup and discusses the experimental results of the proposed

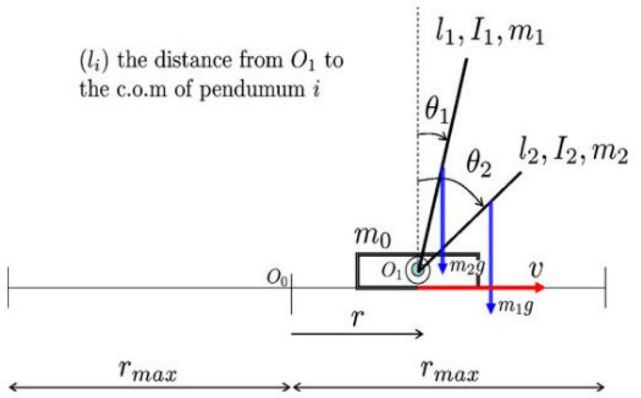


Fig. 1 *Diagram from [11]*

approach. Finally, Section 7 presents conclusions, summarises the contribution of the paper and describes future work.

2 Mathematical modelling

In this section we present the equations of motion for the parallel double inverted pendulum depicted in Fig. 1 based on the assumption that the pendulums' will have a negligible effect on the cart. Moreover, a discretisation of the system is presented based on a backward-forward Euler scheme which will be used for the NMPC, EKF and OSI, presented in the following sections.

2.1 Equations of motion

The equations of motion for the double inverted pendulum can be derived by using Lagrange formalism [10]. As this is well known, this paper uses the model from [11] with additional friction terms, given by (1a) and (1b)

$$\ddot{p} = f_m \dot{p} + ku \tag{1a}$$

$$\begin{aligned} &\ddot{\theta}_i = a_i \dot{\theta}_i + b_i \sin\theta_i + c_i \cos\theta_i (f_m \dot{p} + ku) \\ &\forall i = [1, 2] \end{aligned} \tag{1b}$$

where p is the position of the car; u is an input signal to the system which in this case is a pulse width modulated (PWM) signal for the motor driver; the i index represents the variable or parameter related to the ith pendulum; k is a constant that relate the PWM with the force and mass of the system; θ_i are the angles of the ith pendulum; f_m and a_i are viscous friction constants; $b_i = (m_i l_i g)/(m_i l_i^2 + I_i)$ are the pendulums' gravity related terms; $c_i = b_i/g$ are the acceleration-torque related constants; g is the gravity constant; and m_i, l_i, I_i, are the mass, length and moment of inertia of each pendulum, respectively. In this paper, the relevant coefficients of the model will be found by the OSI algorithm presented in Section 5. Note that the sign of certain coefficients might be subject to the specific experimental setup depending on orientation, e.g. positive cart motion to the left or positive angle rotation CCW, however, all the viscous friction constants (f_m and a_i) must always be negative. Moreover, it should be noted that though counter-intuitive, the lengths of the arms should be different to achieve better controllability of the system [11], particularly in the presence of noise which causes a significantly increased amount of input shattering as the lengths become closer. This was validated through simulations to select appropriate length differences for our particular system.

This model is valid for our particular experimental setup given two conditions are true; both pendulum's masses are much lower than the cart, and the motor driver used has a regenerative braking feature which further cancels out any possible uncontrolled movement of the cart. In the case where the cart motion is indeed affected by the pendulum's motion, a subordinate controller can be developed to cancel this effects as in [10], or the full non-linear model can be included in the general NMPC framework as it has been shown in [3].

2.2 Discretisation

This paper uses a 'direct approach' which requires to 'first discretise, then optimise' [6]. Thus, we now look to discretise the equations of motion (1a) and (1b) which will allow us to simulate and linearise the system for both NMPC and EKF frameworks. This can typically be done using some form of integration method such as the explicit Euler method or explicit Runge Kutta methods [1].

For this work, a forward Euler method was considered at first, following similar works as in [10, 12], however, based on the observation that only position and angles are measured by the system, this scheme was modified to a forward-backward Euler scheme including an extra previous input u_{k-1}, thus augmenting the state to obtain a non-minimal state space (NMSS) [15]. This was motivated by observing that the position dynamics (1a) represent a linear second-order model which is known to have an exact ZOH discretisation of the form:

$$p_{k+1} = (1 + e^{f_m T}) p_k - e^{f_m T} p_{k-1} + b_1 u_k + b_2 u_{k-1} \tag{2}$$

which can also be represented by,

$$p_{k+1} = (2 - a) p_k - (1 - a) p_{k-1} + b_1 u_k + b_2 u_{k-1}. \tag{3}$$

Consider now the backward Euler approximations:

$$\dot{p}_k = \frac{p_k - p_{k-1}}{T_s} \quad \ddot{p}_{k+1} = \frac{\dot{p}_{k+1} - \dot{p}_k}{T_s} \tag{4}$$

(3) can be rearranged as:

$$\underbrace{p_{k+1} - 2p_k + p_{k-1}}_{T_s^2 \ddot{p}_{k+1}} = -a \underbrace{(p_k - p_{k-1})}_{T_s \dot{p}_k} + b_1 u_k + b_2 u_{k-1}. \tag{5}$$

After some algebraic manipulation combining (1a), (5) and (4), the following position acceleration model can be obtained:

$$\ddot{p}_{k+1} = f_m \dot{p}_k + k_1 u_k + k_2 u_{k-1} \tag{6}$$

where f_m, k_1, and k_2 are some equivalent coefficients. This position acceleration model essentially represents a forward Euler approximation using 2 previous inputs (u_k, u_{k-1}) instead of only 1 (as in the standard method), and plays a critical role in the discretisation of the pendulum dynamics (8).

Similarly, considering the backward Euler approximations:

$$\dot{\theta}_{ik} = \frac{\theta_{i_k} - \theta_{i_{k-1}}}{T_s} \quad \ddot{\theta}_{ik+1} = \frac{\dot{\theta}_{ik+1} - \dot{\theta}_{ik}}{T_s} \tag{7}$$

and giving a forward-Euler step in (1b) combined with the position acceleration model (6) results in the following angular acceleration model for the ith pendulum:

$$\begin{aligned} &\ddot{\theta}_{i_{k+1}} = a_i \dot{\theta}_{i_k} + b_i \sin\theta_{i_k} + c_i \cos\theta_{i_k} (f_m \dot{p}_k + k_1 u_k + k_2 u_{k-1}) \\ &\forall i = [1, 2] \end{aligned} \tag{8}$$

Combining (4)–(8), and considering the state $x_k = [p, \theta_1, \theta_2, \dot{p}, \dot{\theta}_1, \dot{\theta}_2, u_{k-1}]_k$, the simulation step is then given as

$$x_{k+1} = x_k + T_s f(x_k, u_k) \tag{9a}$$

$$f(x_k, u_k) = \left[f_{\text{up}}^{\text{T}}, f_{\text{down}}^{\text{T}}, (u_k - u_{k-1})/T_s \right]^{\text{T}} \tag{9b}$$

$$f_{\text{up}} = \begin{bmatrix} \dot{p}_k + T_s f_1 \\ \dot{\theta}_{1_k} + T_s f_2 \\ \dot{\theta}_{2_k} + T_s f_3 \end{bmatrix} \quad f_{\text{down}} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} \tag{9c}$$

$$f_1 = f_m \dot{p}_k + k_1 u_k + k_2 u_{k-1} \tag{9d}$$

$$f_2 = a_1 \dot{\theta}_{1_k} + b_1 \sin\theta_{1_k} + c_1 \cos\theta_{1_k} f_1 \tag{9e}$$

$$f_3 = a_2 \dot{\theta}_{2_k} + b_2 \sin\theta_{2_k} + c_2 \cos\theta_{2_k} f_1 \tag{9f}$$

where T_s is the sampling time. It is noted that the last term of function (9b) was only used to represent the propagation of the input $x_{7_{k+1}} = u_k = u_{k-1} + T_s(u_k - u_{k-1})/T_s$ and does not represent, in any way, a 'derivative of the input' $\dot{u}_k = (u_k - u_{k-1})/T_s$ which would lead to a completely different integration scheme if more intermediate steps were computed [1].

This discretisation differs slightly from the standard forward Euler method in the sense that the latter would compute a forward-Euler step on the higher derivative states (p, θ_1, θ_2). However, as stated previously, given only position and angle's measurement were available, the backward Euler approximations (4) and (7) were used instead to use the latest information of the system. Moreover, it would only take into account u_k for simulation purposes. Nonetheless, this modification was observed to produce much better predictions in offline analysis of the system identification process for both, position and angle dynamics, and in fact, increasing the number of previous input terms u_{k-j} was be able to improve them even further, possibly given that it accommodates some unmodelled higher-order motor dynamics which are known to be at least second-order in the angular velocity; third-order in the angular position; which can be accounted for using convolution/FIR models. However, the system was observed to get good performance whilst only including two previous input terms, u_k and u_{k-1}.

3 Non-linear model predictive control

In this section, a dual mode NMPC scheme based on the closed-loop paradigm [16] is proposed to cancel the open-loop unstable dynamics of the double inverted pendulum for numerical robustness of the optimisation. Furthermore, to enable real-time performance, the optimisations are performed within the RTI scheme which allows the constrained optimisation to be solved within the microsecond range [6]. Finally, a modification to the standard cost used for inverted pendulum control is proposed based on energy considerations along with a hybrid switching scheme which overall significantly improves the convergence of the algorithm, particularly for large disturbances, a situation where the assumptions for local-asymptotic closed-loop stability of the RTI scheme are lost. Simulations are presented along the section to illustrate the significance of the proposed approach.

3.1 Stable predictions and optimisation

In this paper, we are looking to optimise the system performance along a given prediction horizon N_p by minimising the cost function (10), defined as

$$J = \frac{1}{2}(Y_r - \hat{Y})^{\mathrm{T}} Q\,(Y_r - \hat{Y}) + \frac{1}{2}\hat{U}^{\mathrm{T}} R \hat{U} \quad \text{s.t.} \tag{10a}$$

$$\hat{x}_k = x_0 \tag{10b}$$

$$\hat{x}_{k+i|k} = f(\hat{x}_{k+i-1|k}, \hat{u}_{k+i-1|k}) \tag{10c}$$

$$\hat{y}_{k+i|k} = g(\hat{x}_{k+i|k}) \tag{10d}$$

$$U_{\min} \leq \hat{U} \leq U_{\max} \tag{10e}$$

$$Y_{\min} \leq \hat{Y} \leq Y_{\max} \tag{10f}$$

where $x_k \in \mathbb{R}^{n_x}$, $u_k \in \mathbb{R}^{n_u}$ and $y_k \in \mathbb{R}^{n_y}$ are the states, inputs and outputs of the system, respectively; the notation '$k+1|k$' reads 'predicted value at $k+1$ considered at sample time k', and will only be used in full when needed for clarity; $Q > 0 \in \mathbb{R}^{N_p n_y \times N_p n_y}$ and $R > 0 \in \mathbb{R}^{N_p n_u \times N_p n_u}$ are positive-definite matrices for penalising output-errors and inputs, respectively, typically selected as $Q = \text{blkdiag}([q_{k+1}, q_{k+2}, \ldots, q_{k+Np}])$ where q_{k+N_p} is typically referred to as the terminal weight, and $R = r_u I^{N_p n_u \times N_p n_u}$; $Y_r = [r_{k+1}^{\mathrm{T}}, r_{k+2}^{\mathrm{T}}, \ldots, r_{k+N_p}^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{N_p n_y}$, $\hat{Y} = [\hat{y}_{k+1}^{\mathrm{T}}, \hat{y}_{k+2}^{\mathrm{T}}, \ldots, \hat{y}_{k+N_p}^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{N_p n_y}$, $\hat{U} = [\hat{u}_k^{\mathrm{T}}, \hat{u}_{k+1}^{\mathrm{T}}, \ldots, \hat{u}_{k+N_p-1}^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{N_p n_u}$ are references, outputs and inputs column-vectors, respectively; (10b) is the initial condition; (10c) are the state dynamics; (10d) is the function that relates the output with the states; (10e) are the input constraints; and (10f) are the output constraints. For our particular system, the outputs are typically selected as in [1, 10, 12] as $y_{k+i} = [\dot{p}, \dot{\theta}_1 \dot{\theta}_2, p, \theta_1, \theta_2]_{k+i}$ ($n_y = 6$), and the references are selected as $r_{k+i} = [0, 0, 0, p_r, \theta_{1_r}, \theta_{2_r}]_{k+i}$.

Remark 1: Stability of the resulting closed-loop system can be typically ensured by having long horizons with zero-terminal constraints and/or proper terminal weights [6].

Cost function (10) for system (9a) represents a non-convex non-linear programming problem which is difficult to solve. Sequential quadratic programming (SQP) is a popular alternative where the cost is linearised at a given point to formulate a linearised quadratic program (QP) and find an optimal search direction, typically based on the Newton method, that eventually drives the solution to the local optimal. Notice in the case of predictive control, future state trajectories $\hat{x}_{k+i}$ required for the linearisation are only defined after a given input trajectory $\hat{u}_{k+i-1}$ has been applied through the state dynamics (10c) with the initial condition (10b). A workaround to this are shooting methods which use an 'initially guessed' **nominal input trajectory**, $\bar{U} = [\bar{u}_k^{\mathrm{T}}, \bar{u}_{k+1}^{\mathrm{T}}, \ldots, \bar{u}_{k+N_p-1}^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{N_p n_u}$ to generate **nominal state and output trajectories**, $\bar{X} = [\bar{x}_{k+1}^{\mathrm{T}}, \bar{x}_{k+2}^{\mathrm{T}}, \ldots, \bar{x}_{k+N_p}^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{N_p n_x}$ and $\bar{Y} = [\bar{y}_{k+1}^{\mathrm{T}}, \bar{y}_{k+2}^{\mathrm{T}}, \ldots, \bar{y}_{k+N_p}^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{N_p n_y}$, respectively, by simulating the system with $\bar{U}$ using initial condition (10b) and state dynamics (10c).

The standard NMPC single-shooting approach would linearise the system along this resulting trajectories with a first-order Taylor Series on the state dynamics, the output-state function, and the input. However, given the open-loop unstable dynamics of the inverted pendulum in its upright equilibrium, closed-loop dual-mode prediction models were motivated [16]. The linearised model at a given time step k is then given by,

$$\hat{x}_{k+1} = \bar{x}_{k+1} + \delta\hat{x}_{k+1} = \bar{x}_{k+1} + A_k \delta\hat{x}_k + B_k \delta\hat{u}_k \tag{11a}$$

$$\hat{y}_k = \bar{y}_k + \delta\hat{y}_k = \bar{y}_k + C_k \delta x_k \tag{11b}$$

$$\hat{u}_k = \bar{u}_k + \delta u_k = \bar{u}_k - K_k \delta x_k + \delta\hat{c}_k \tag{11c}$$

where K_k is a stabilising gain obtained from solving the time-varying discrete algebraic Ricatti equation backwards in time along the nominal state trajectory using the same Q and R weights as in [10], and:

$$A_k = \left.\frac{\partial f(\hat{x}_k, \hat{u}_k)}{\partial \hat{x}_k}\right|_{\substack{\hat{x}_k = \bar{x}_k \\ \hat{u}_k = \bar{u}_k}} B_k = \left.\frac{\partial f(\hat{x}_k, \hat{u}_k)}{\partial \hat{u}_k}\right|_{\substack{\hat{x}_k = \bar{x}_k \\ \hat{u}_k = \bar{u}_k}} \tag{12a}$$

$$C_k = \left.\frac{\partial g(\hat{x}_k)}{\partial \hat{x}_k}\right|_{\hat{x}_k = \bar{x}_k} \tag{12b}$$

Notice this dual-mode prediction model differs from common dual-mode schemes in the sense that standard methods would select a gain K, typically a constant one, which stabilises the system to the origin when the system is in the terminal region [17].

In contrast, our approach uses dual-mode closed-loop models to stabilise the predictions and achieve better numerical performance as in [13]. This is because we aim to deal with the situation when the system is not in the terminal region, and finding a stabilising gain that does the swing up whilst satisfying the constraints is, in general, not a trivial task for our system.

By substituting $\delta\hat{u}_k = -K_k\delta\hat{x}_k + \delta\hat{c}_k$ from (11c) in (11a), a stable state deviation model can be obtained as,

$$\Phi_k = A_k - B_k K_k \tag{13a}$$

$$\delta\hat{x}_{k+1} = \Phi_k\delta\hat{x}_k + B_k\delta\hat{c}_k \tag{13b}$$

Propagating model (13b) N_p steps forward starting from an initial state mismatch δx_0, leads to the following predictions matrices for all future inputs and outputs, $\hat{U}$ and $\hat{Y}$, are condensely represented by,

$$\hat{Y} = \bar{Y} + \delta\hat{Y} = \bar{Y} + G\delta x_0 + H\delta\hat{C} \tag{14a}$$

$$\hat{U} = \bar{U} + \delta\hat{U} = \bar{U} + D\delta x_0 + F\delta\hat{C} \tag{14b}$$

where $\delta x_0 = x_0 - \bar{x}_0$ is the initial condition mismatch which forms a part of the RTI scheme, $\delta\hat{C}$ are now the decision variables,

$$G = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_{N_p} \end{bmatrix} \quad H = \begin{bmatrix} h_{1,1} & 0 & \cdots & 0 \\ h_{2,1} & h_{2,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ h_{N_p,1} & h_{N_p,2} & \dots & h_{N_p,N_p} \end{bmatrix} \tag{15a}$$

$$D = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{N_p} \end{bmatrix} \quad F = \begin{bmatrix} I^{n_u} & 0 & \cdots & 0 \\ f_{2,1} & I^{n_u} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ f_{N_p,1} & f_{N_p,2} & \dots & I^{n_u} \end{bmatrix} \tag{15b}$$

where I^{n_u} is a $n_u \times n_u$ identity matrix, $G \in \mathbb{R}^{N_p n_y \times n_x}$, $H \in \mathbb{R}^{N_p n_y \times N_p n_u}$, $D \in \mathbb{R}^{N_p n_u \times n_x}$, $F \in \mathbb{R}^{N_p n_u \times N_p n_u}$, and

$$g_i = C_i \overset{\curvearrowleft}{\prod_{k=0}^{i-1}} \Phi_k \tag{16a}$$

$$h_{i,j} = \begin{cases} C_i B_{j-1}, & i = j \\ C_i \left[\overset{\curvearrowleft}{\prod_{k=j}^{i-1}} \Phi_k \right] B_{j-1}, & i > j \end{cases} \tag{16b}$$

$$d_i = \begin{cases} -K_{i-1}, & i = 1 \\ -K_{i-1} \overset{\curvearrowleft}{\prod_{k=0}^{i-2}} \Phi_k, & i > 1 \end{cases} \tag{16c}$$

$$f_{i,j} = \begin{cases} -K_{i-1} B_{j-1}, & i = j+1 \\ -K_{i-1} \left[\overset{\curvearrowleft}{\prod_{k=j}^{i-2}} \Phi_k \right] B_{j-1}, & i > j+1 \end{cases} \tag{16d}$$
$$\forall i = [1, N_p] \quad \forall j = [1, N_p]$$

where the i index refers to the ith matrix-row of matrices G,H,D and F, and its conceptually related to the outputs predicted i steps ahead; and the j index refers to the jth matrix-column of matrices H and F, and its conceptually related to the decision variables j steps into the future.

Substituting the stable linearised prediction models (14) in (10) and rearranging in terms of the decision variable $\delta\hat{C}$ to obtain the standard QP format gives;

$$J = \frac{1}{2}\delta\hat{C}^{\mathrm{T}} E \delta\hat{C} + \delta\hat{C}^{\mathrm{T}} f + const \quad s.t. \tag{17a}$$

$$E = H^{\mathrm{T}} Q H + F^{\mathrm{T}} R F \tag{17b}$$

$$f = -\left[H^{\mathrm{T}} Q (Y_r - \bar{Y} - G\delta x_0) - F^{\mathrm{T}} R (\bar{U} + D\delta x_0) \right] \tag{17c}$$

$$M\delta\hat{C} \leq \gamma \tag{17d}$$

$$M = \begin{bmatrix} F \\ -F \\ H \\ -H \end{bmatrix} \quad \gamma = \begin{bmatrix} U_{\max} - \bar{U} - D\delta x_0 \\ -(U_{\min} - \bar{U} - D\delta x_0) \\ Y_{\max} - \bar{Y} - G\delta x_0 \\ -(Y_{\min} - \bar{Y} - G\delta x_0) \end{bmatrix} \tag{17e}$$

with E known as the Hessian, f typically referred as the linear term, and M and γ are the constraint matrix and vector, respectively. Notice (10b)–(10d) are implicit in the linearisation of $\hat{U}$ and $\hat{Y}$. Moreover, note the not all the outputs may be required to be constrained which can be done by selecting (or computing) only the relevant rows of M and γ. In our particular system, only the position outputs will be constrained.

By derivating (17a) w.r.t. the decision variable $\delta\hat{C}$ and equating to zero $((\partial J/\partial\hat{C}) = 0)$, the well known unconstrained solution can be found to be $\delta\hat{C} = -E^{-1}f$. For constrained solutions, any QP solver can be used to compute the optimal deviation $\delta\hat{C}$ after having defined E, f, M, γ. For our experiments, an efficient version of the active-set based primal-dual Hildreth's QP found in [15] was used given its simplicity and its ability to be hot-started which is required for achieving fast implementation of the overall scheme. After solving the optimisation, the corrected input $\hat{U}$ can then be recovered by (14b). Only the first input is applied to the system and the process is repeated which is the well known 'receding horizon' strategy [11].

3.1.1 Stability and numerical robustness: Because this model produces the same predictions for a given $\delta\hat{U} = D\delta x_0 + F\delta\hat{C}$ than using the standard model without the stable predictions, and because $\delta\hat{U}$ can always be calculated exactly through the inversion of F which is always invertible, the solution for $\hat{U}$ is the same as the one given by the standard approach using unstable predictions, and therefore presents the same stability and convergence properties of the standard single-shooting approach. The benefit of it is that the predictions matrix H is now stable w.r.t the decision variable $\delta\hat{C}$ which leads to a numerically robust Hessian inversion required by the optimisation. This allows the prediction horizon to be increased as much as required without sacrificing numerical robustness. For reference, in our particular system, the condition number (c.n.) of the Hessian at the upward equilibrium when using the proposed approach was around the unit magnitude ($E_{c.n.} = 2.5 \times 10^0$); the c.n. without using the approach was ($E_{c.n.} = 5 \times 10^6$), 6 orders of magnitude larger which indeed shows severe numerical issues given it comes close to singular as the c.n. increases.

Finally, for the interest of the reader, the standard NMPC condensed single-shooting solution can be recovered by enforcing $K_{k+i} = \mathbb{O}, \quad \forall i = [0, N_p - 1]$ which would then use the unstable predictions.

3.2 Real time iterations

Ideally, the fully converged NMPC would relinearise cost function (10) until no deviation is required $\delta\hat{C} = \mathbb{O}$ [1]. However, this is not computationally tractable in practice given one must give a

solution at every time-step within the available time and avoid solving a problem that is only 'getting older' [9]. A very successful and popular approach to address this is to use the RTI scheme which exploits the fact that NMPC is required to successively solve optimisations which are closely related to each other. The method benefits from the fast contraction rate of Newton-type optimisations and achieves convergence of the solution 'on the fly', using the current predictions and measurements rather than through offline reference trajectories [1]. The overall RTI scheme is based on three well-defined strategies.

3.2.1 Initial value embedding: The input trajectory obtained in the previous sampling $\hat{U}_k = [\hat{u}_{k|k}^{\mathrm{T}}, \hat{u}_{k+1|k}^{\mathrm{T}}, \ldots, \hat{u}_{k+N_p-1|k}^{\mathrm{T}}]^{\mathrm{T}}$ is used in a shifted version to hot start the solution in the next sampling time, typically by duplicating the last value $\hat{U}_{k+1} = [\hat{u}_{k+1|k}^{\mathrm{T}}, \hat{u}_{k+2|k}^{\mathrm{T}}, \ldots, \hat{u}_{k+N_p-1|k}^{\mathrm{T}}, \hat{u}_{k+N_p-1|k}^{\mathrm{T}}]^{\mathrm{T}}$. Moreover, in the case of active-set based QP, the lagrange multipliers λ related to the constraints of the optimisation can also be used for hotstarting the QP in a shifted version.

3.2.2 Single SQP: Only a single linearisation of the QP is performed given the solution is hot started from the previous solution which is expected to be close. In the case where the previous solution was indeed close to the optimal solution and no significant disturbances have entered the system, this approach can be proved to have nominal local-asymptotic closed-loop stability [18]. In general, the solution is not given exactly but as an approximation that decreases the sub-optimality of cost J at each iteration. Moreover, one must be satisfied with finding a local minimum, and the solution will be subject to small approximation errors given only one re-linearisation is done.

3.2.3 Computation separation: To avoid the delay related to the computations required by the optimisation, we divide them into preparation and a feedback phase. A timing diagram that illustrates this is given in [1].

1. Preparation Phase: Uses a predicted nominal state $\hat{\bar{x}}_{k+1|k}$ obtained with the current input and states to compute the matrices E,M and vectors f,γ required by the optimisation assuming $\delta x_0 = 0$.
2. Feedback Phase: As soon as the state x_{k+1} becomes available, the deviation $\delta x_0 = x_{k+1} - \hat{\bar{x}}_{k+1|k}$ is used to complete the calculation of f and γ and the optimal correction $\delta\hat{U}$ to the current trajectory $\hat{U}$.

A slight modification was implemented in our test where the deviation δx_0 was only applied to the linear term f and the QP was iterated assuming $\delta x_0 = 0$ to find the active set λ before the state x_{k+1} was available. By doing so, the solution is now given by (18), which allows the preparation phase part ($\hat{U}_{pre-computed}$) to be pre-computed prior to the arrival of the measurement, and only the feedback correction $\hat{U} = \bar{U} - \hat{U}_{pre-computed} - K_x\delta x_0$ is required to be computed when the state becomes available, with $K_x = FE^{-1}(H^{\mathrm{T}}QG + F^{\mathrm{T}}RD) - D$

$$\hat{U} = \bar{U} + \overbrace{D\delta x_0}^{\text{Feedback Phase}} - FE^{-1}\Bigg[\underbrace{\overbrace{-(H^{\mathrm{T}}Q(Y_r - \bar{Y}) - R\bar{U})}^{\text{Unconstrained}} \overbrace{+M^{\mathrm{T}}\lambda}^{\text{Constrained}}}_{\text{Preparation Phase } (\hat{U}_{\text{pre-computed}})} + \underbrace{(H^{\mathrm{T}}QG + F^{\mathrm{T}}RD)\delta x_0}_{\text{Feedback Phase}}\Bigg]. \tag{18}$$

This modification completely removes the time-delay related to the iterations of the QP required to be done by the standard RTI in the feedback phase. By including this term δx_0 in the linear term f, the stability characteristics of the overall RTI scheme are preserved, however, given the term is ignored for the calculation of the constrained correction $E^{-1}M^{\mathrm{T}}\lambda$, the system may present small output constraint violations depending on how large is the deviation from the predicted state at a given time. However, this modification is justified considering that feasibility guarantees for output constraints in the presence of disturbances are, in general, difficult to achieve without using robust approaches or slack variables (soft-constriants) which are outside the scope of this paper. Nonetheless, the system presented excellent performance in constraints satisfaction as it will be seen in the experimental results presented in Section 6.

3.3 Improving RTI NMPC performance

A major issue with the RTI scheme is that the solution might give very poor performance whenever an abrupt change is made, e.g. when there is an abrupt change in the reference of the system [1], or a large fault or disturbance enters the system, which may lead to leaving the region of contraction of the Gauss–Newton method and in some cases may even lead to instability of the system [6]. In these cases, the previous solution will not be close to the optimal, and therefore, the method would need to quickly find a suitable correction from the previous solution. This issue may be addressed by adding suitable end weights and other regularity conditions [6]. However, to address this issue, this paper takes a different approach with two main modifications to the standard approach of NMPC of an inverted pendulum such as [1, 3, 7, 12], namely: an additional energy based cost; and a hybrid switching scheme.

3.3.1 Energy-based costs: Motivated by the fact that a common strategy for the swing-up of the pendulum are energy-based control laws [11], along with the fact that standard cost terms defined for inverted pendulum NMPC, e.g. [1, 12], do not actually capture the requirement of 'swinging-up' but rather a more restrictive cost requiring the optimisation to drive the angles to the desired reference without considering other upward equilibrium points, the outputs and references used for the cost function (10) were modified to include two extra terms related to the potential energy of both pendulums $E_{\theta_i} = \cos\theta_i$ as

$$y_k = \left[\dot{p}, \dot{\theta}_1, \dot{\theta}_2, p, \theta_1, \theta_2, \cos\theta_1, \cos\theta_2\right]_k \tag{19a}$$

$$r_k = \left[0, 0, 0, p_r, \theta_{1_r}, \theta_{2_r}, \cos\theta_{1_r}, \cos\theta_{2_r}\right]_k. \tag{19b}$$

Remark 2: With this modification, the optimisation now has $n_y = 8$ outputs.

Some of the relevant properties of this added term are:

1. *Boundedness:* The error $e_k = \cos\theta_{1_r} - \cos\theta_1$ is always bounded at $e_k = [-2, 0]$ for the upright position, and $e_k = [0, 2]$ for the downside position. This in general would make the additional term of the linear term (f) bounded.
2. *Singularity:* The derivative w.r.t. the added term ($C_i = -\sin\theta_i$) required by (14a) has a singularity at any $N\pi$ multiple given the sensitivity matrix is zero. Thus, if the system is at a steady condition, e.g. all other errors zero, the optimisation would have no sensitivity on it, therefore, not reacting or causing any movement. Although the system can be confined inside an incorrect singularity, if the system is started at any other sufficiently non-singular point, the optimisation will eventually drive the solution to the desired singularity.

By penalising the energy term much higher than the angles directly, the optimisation is more relaxed, essentially aiming to drive the potential energy of the pendulum to the desired state $E_{\theta_i} = \cos\theta_i \to \cos\theta_{i_r}$ whilst accepting swinging up in either direction. This is because if at a given time the system cannot swing the pendulums up in a given direction, the optimisation would naturally select the other direction which is not the case

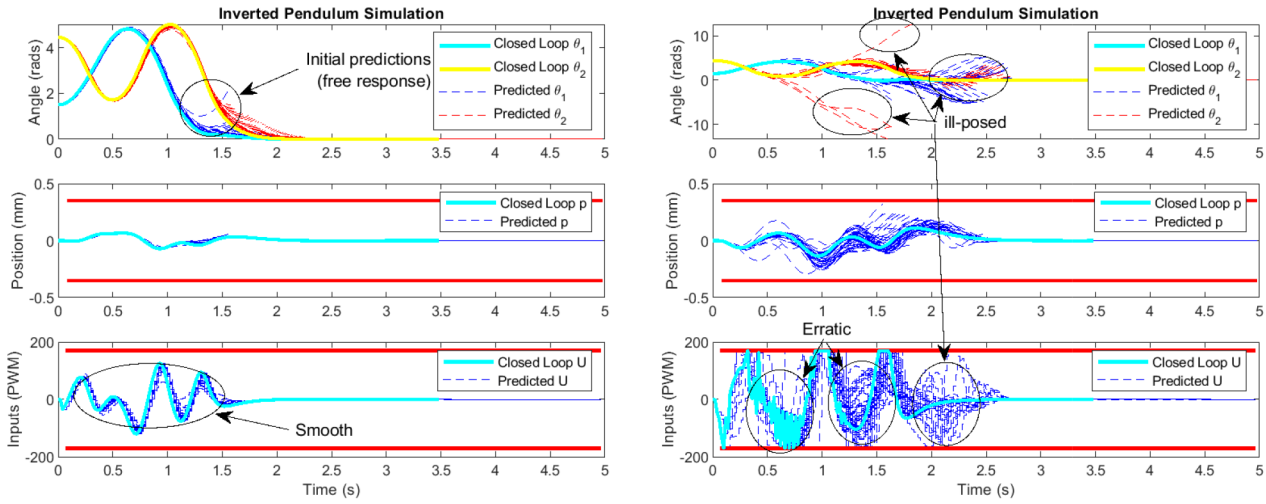


Fig. 2 *Example comparison of predicted (dashed-lines) and closed-loop (thick lines) responses with (2a) and without (2b) energy costs with parameteres defined in table 1;* $T_s = 0.02\,(\mathrm{s})$; $N_p = 75$; $x_0 = [0, 0, 0, 0, 1.51, 4.45, 0]^{\mathrm{T}}$; $\bar{U}_0 = \mathbb{O}$ *(free-response);* $Q_{with} = \mathrm{diag}([1, 0.1, 0.1, 100, 1, 1, 10, 10])$, $Q_{without} = \mathrm{diag}([1, 0.1, 0.1, 100, 10, 10, 0, 0])$, *and* $R = 0.001$

(a) With, ***(b)*** Without

Table 1 Identified parameters for the double inverted pendulum (4)

Motor	Coeffs.	Pend 1	Coeffs.	Pend 2	Coeffs.
f_m	−4.67	a_1	−0.129	a_2	−0.107
k_1	0.0174	b_1	38.4	b_2	49.6
k_2	0.0477	c_1	3.95	c_2	5.11

when the standard costs are used, and the solution was observed to be severely affected when the system reached this infeasibility condition. After a series of simulations, it was concluded that imposing higher penalties on this added terms instead of the angles directly, resulted in much better convergence properties than when using the standard cost, which in turn resulted in a larger region of contraction of the Gauss–Newton method. Moreover, notice the stability of the resulting scheme can still be guaranteed by imposing heavy terminal weights in the additional terms to emulate zero-terminal constraints, even though the sensitivity at that condition dissipates, which in turn leads to having the original problem once the system has reached the terminal region.

To visualise the benefits of this approach, a comparison simulation is given in Fig. 2 where the predicted and closed-loop trajectories are plotted, with and without focusing on the added energy term. For clarity, the predicted responses that presented the erratic behaviour are signalled. As it can be seen from (Fig. 2*b*), the optimisation penalising only the angles presented major erratic behaviour in the input at times $0.5 < t < 1$,and significant differences between predicted and closed-loop responses leading to ill-posed optimisation [16]. In contrast, the optimisation that focused effort on the added energy-cost (Fig. 2*a*) presented smooth and much better overall closed-loop performance.

Although this approach might not be immediately generalisable for other control systems and applications, it is very common to find trigonometric terms in robotics systems and mechatronic applications that arise from rotation matrices. This naturally brings the question of whether it is better to target a desired potential energy or an angle directly when dealing with multi-link robots. Indeed, it is well known that the understanding of the inverted pendulum dynamics helped with the development of many robotic applications nowadays, thus its generalisation to multi-link robotic problems, eg. triple inverted pendulum in series [10], could lead to a significant improvement in performance in a broader spectrum of applications, particularly when using the RTI scheme.

3.3.2 Hybrid switching scheme: As discussed previously, penalising the energy-related terms lead to smoother responses. However, because of aforementioned singularity problem, if only the energy terms are penalised instead of the angles directly, the optimisation would have no sensitivity to potential energy errors at the equilibrium, and would only be sensitive to angular velocity errors. Moreover, if a sufficiently small penalty was imposed on the angles, the optimisation could converge to upright positions were the angle errors were essentially ignored.

To avoid this problem whilst preserving the smoothness of the added energy terms during the swing up phase, a hybrid approach was used where the optimisation would switch between different weightings depending, not only on the region in which the angles where but also the time that they have been there.

The hybrid switching scheme is given by

$$q_{\theta_i} = \begin{cases} 1, & t_{\mathrm{lin}} < 2 \\ 10, & t_{\mathrm{lin}} \geq 2 \end{cases} \tag{20}$$

where q_{θ_i} is the weight of the ith pendulum angle error; and t_{lin} is the time that has elapsed since $\cos\theta_i > 0.9$, i.e. the time the system has been in the 'linear' zone.

Regarding the stability of this proposed hybrid scheme, it should be noted that both penalisation terms of q_{θ_i} were stable for our particular system, and the only reason for this change was to preserve the smoothness of the system during the swing up phase. As the change was implemented when the system was already in the terminal region, the cost of both selected weights dissipated to zero within the available horizon, which in turn made the change between both weights stable. Essentially, the selection of these different terms changes the frequency response of the system to a more 'rigid' or fast response for angle perturbations. Indeed, this approach could be used for fault-tolerant applications where the system momentarily has to undergo through a 'softer/smoother' set of actions to bring the system back to its target before regaining a more 'reactive' state.

4 State estimation – extended Kalman filter

As the only position and angle measurements were available in our system, a standard EKF was used for the purpose of state estimation. As this is a well-known method in the literature, the details of this are omitted and only the relevant equations and steps are provided.

The EKF uses the following prediction-correction type framework:

1. *Prediction Step:* The state, output and covariance at time k are estimated based on the previously estimated state and covariance as

$$\hat{x}_{k|k-1} = \hat{x}_{k-1|k-1} + T_s f(\hat{x}_{k-1|k-1}, u_{k-1}) \tag{21a}$$

$$P_{k|k-1} = A_{k-1} P_{k-1|k-1} A_{k-1}^{\mathrm{T}} + Q_{EKF} \tag{21b}$$

where $Q_{EKF} > 0$ is the process noise; and $P_{k|k-1}$ is the covariance matrix.

2. *Correction Step:* As soon as the outputs of the system become available, the correction step is then given by

$$\hat{y}_{k|k-1} = g(\hat{x}_{k|k-1}) \tag{22a}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_{EKF}\left[(y_k - \hat{y}_{k|k-1})\right] \tag{22b}$$

$$P_{k|k} = (I - K_{EKF} C_k) P_{k|k-1} \tag{22c}$$

$$K_{EKF} = P_{k|k-1} C_k^{\mathrm{T}} \left(C_k P_{k|k-1} C_k^{\mathrm{T}} + R_{EKF}\right)^{-1} \tag{22d}$$

where $R_{EKF} > 0$ is the output noise covariance matrix, and $C_k = \frac{\partial g(\hat{x}_{k|k-1})}{\partial \hat{x}_{k|k-1}}$.

Remark 3: As system (9a) is a NMSS, the input-related state (x_7) must be included in the measurement for observability. The proof of this is out of the scope of this paper.

5 Online system identification

In this section, an OSI scheme based on RLS with a forgetting factor combined with a delta-modelling approach [19] is presented. The latter was used for the purpose of learning/adapting the parameters of the discrete model (9a), particularly $f_m, k_1, k_2, a_i, b_i, c_i$. As this is a well-known method in the literature, the details of this are omitted and only the relevant equations and steps are provided.

The RLS algorithm with forgetting factor (λ) is given by

$$\tilde{z}_k = z_k - \Psi^{\mathrm{T}} \Theta_{k|k-1} \tag{23a}$$

$$K_{RLS} = P_{k|k-1} \Psi \left(\lambda + \Psi^{\mathrm{T}} P_{k|k-1} \Psi\right)^{-1} \tag{23b}$$

$$\Theta_{k|k} = \Theta_{k|k-1} - K_{RLS} \tilde{z}_k \tag{23c}$$

$$P_{k|k} = \lambda^{-1} \left(P_{k|k-1} - K_{RLS} \Psi^{\mathrm{T}} P_{k|k-1}\right) \tag{23d}$$

where Ψ is known as the regressors vector, $\Theta_{k|k}$ is the parameters vector, $P_{k|k}$ is a covariance matrix of appropriate dimensions, and λ is the forgetting factor, typically selected as $0.98 < \lambda < 1$.

For our system, the definition of Ψ, $\Theta_{k|k}$ and z_k for both, position and pendulum dynamics, are given by;

1. *Position dynamics:*

$$\Psi = [\dot{p}_{k-1}, u_{k-1}, u_{k-2}]^{\mathrm{T}} \tag{24a}$$

$$z_k = \ddot{p}_k = \frac{p_k - 2p_{k-1} + p_{k-2}}{T_s^2} \tag{24b}$$

$$\Theta_{k|k} = [f_m, k_1, k_2]^{\mathrm{T}}. \tag{24c}$$

2. *Pendulum dynamics:*

$$\Psi = [\dot{\theta}_{i_{k-1}}, \sin\theta_{i_{k-1}}, \cos\theta_{i_{k-1}} \ddot{p}_k]^{\mathrm{T}} \tag{25a}$$

$$z_k = \ddot{\theta}_{i_k} = \frac{\theta_k - 2\theta_{k-1} + \theta_{k-2}}{T_s^2} \tag{25b}$$

$$\Theta_{k|k} = [a_i, b_i, c_i]^{\mathrm{T}} \tag{25c}$$

which can be derived from the position and angle acceleration models (6) and (8), and also represent the use of the Delta – modelling approach [19], which is known to numerically perform better than ARX models for a system with fast sampling times. The coefficients were then extracted to be used in both NMPC and EKF frameworks presented previously.

On the other hand, several execution rules discussed in [20, 21] were implemented to shut-down the algorithm to protect it from periods of poor excitation which can lead to the rapid grow of covariance matrix $P_{k|k}$.

The implemented shut-down rules were:

1. The trace of the covariance $P_{k|k}$ was limited by

$$P_{k|k} = \frac{k_{\lim}}{tr(P_{k|k})} P_{k|k} \quad \text{if } tr(P_{k|k}) > k_{\lim} \tag{26}$$

to prevent it from becoming ill-conditioned, where for our particular system $k_{\lim} = 10$. Additionally, the limitation of the covariance trace allows for better control of the rate of convergence of the parameters.

2. The range of the parameters of both models were limited according to offline analysis, as well as based on the uncertainty of expected models coefficients. In particular, a threshold of $\pm 15\%$ was imposed for the coefficients b_i and c_i calculated from the expressions given in Section 2.1, as well as forcing the negative sign of the friction terms f_m and a_i.

If the RLS algorithm moved any of the coefficients outside the available range, they were simply saturated.

3. The RLS algorithms were only run when the system was detected to be moving, in particular, when the angular velocity was greater than a threshold, and kept running for a maximum of 2 s after this conditions were satisfied to be able to capture 'decaying' dynamics. Specifically, the thresholds used were:

$$\| \dot{p}_k \| > 0.4 \left(\frac{m}{s}\right) \quad \| \dot{\theta} i_k \| > 0.5 \left(\frac{\mathrm{rad}}{s}\right) \tag{27}$$

This ensured that the system was properly excited.

4. The model used for the NMPC was only updated if the uncertainty of the coefficients, captured in the covariance matrix $P_{k|k}$ was lower then a threshold. For this system, the uncertainty was simply considered as the trace of the covariance matrix, although a more accurate distribution of the uncertainty could be extracted by using the so-called Chi-squared (χ^2) distribution.

Although the rules for shut-down did improve the performance of the OSI as a potential adaptive controller, a proper excitation signal is required for better model estimation such as PRBS or frequency sweep (chirp), as without there is no guarantee that the model estimation would be accurate or even stable, thus no stability guarantee of the combined methodology could be provided, which is generally known. The overall performance of this algorithm can be seen in Fig. 3 and will be discussed in the results Section 6.1.

6 Experimental results

The test bench used for the experiments is depicted in Fig. 4. The cart is driven by a brushed 24 V DC motor via a toothed belt and a toothed-pulley of 0.05 (m) diameter. The DC motor is driven by a Cytron MD30C Motor Driver operated using the sign-magnitude drive with an 8-bit resolution PWM at a frequency of 20 kHz via a micro controller unit (MCU). Three incremental encoders are used to measure both pendulum angles and the DC motor rotation. The resolution of both pendulum encoders and the motor are 4000 and 2040 counts per revolution, respectively, which are processed by the MCU, leading to angle and position resolutions of $9 \times 10^{-2^\circ}$ and

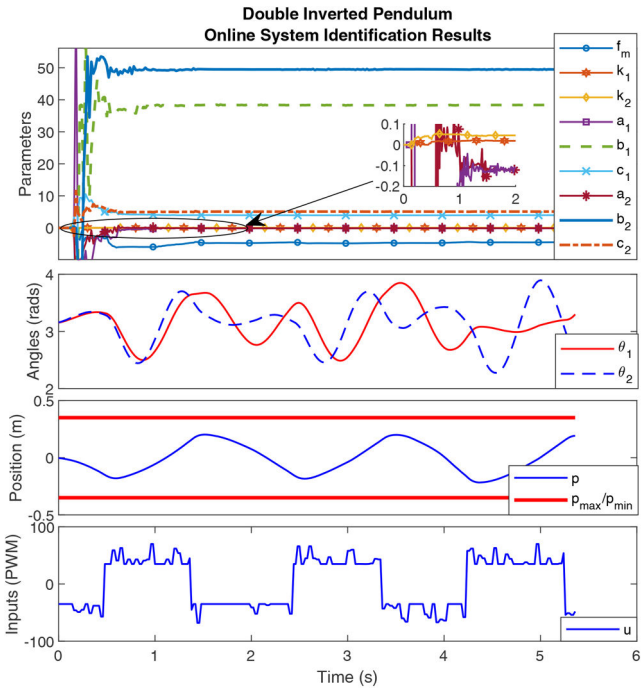


Fig. 3 *Online system identification example*

Fig. 4 *Test bench photograph*

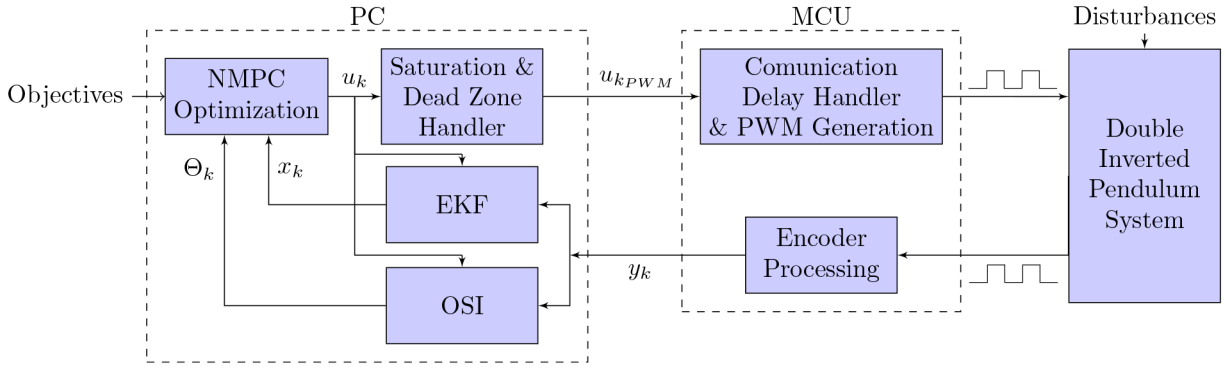


Fig. 5 *Control diagram*

7.7×10^{-5} (m), respectively. The sampling time of the system is handled by the MCU and kept constant at $T_s = 20$ (ms). Every sampling time, encoders data is streamed via (UART) serial communication to a PC where the calculations related to proposed NMPC approach, OSI and EKF are performed. After the control action is calculated via the RTI scheme, it is sent back to the MCU which generates the motor signals. Due to network communication delays, the motor signal was always implemented exactly 5 ms after the encoders data was streamed to have a constant behaviour at least. Fig. 5 shows an control diagram detailing the interaction between the different components.

6.1 Online system identification

To test the OSI algorithm presented in Section 5, the system was excited using a random input $30 < \| u \| < 60$, which reversed every time the system crossed a maximum limit of the position $\| x \| > 0.15$ (m) in the current direction. All the parameters were started from completely unknown values $\Theta_0 = \mathbb{0}$ with a forgetting factor of $\lambda = 0.995$ and initial covariance matrices as $P_0 = 1000 I_{3\times 3}$. The resulting performance of the overall OSI algorithm can be seen in Fig. 3. As it can be seen, the system presented very fast convergence rates, giving settling times for all the parameters of $\tau_s < 2$ (s), indicating that the models are indeed well defined. The resulting parameters after 1 min of excitation are

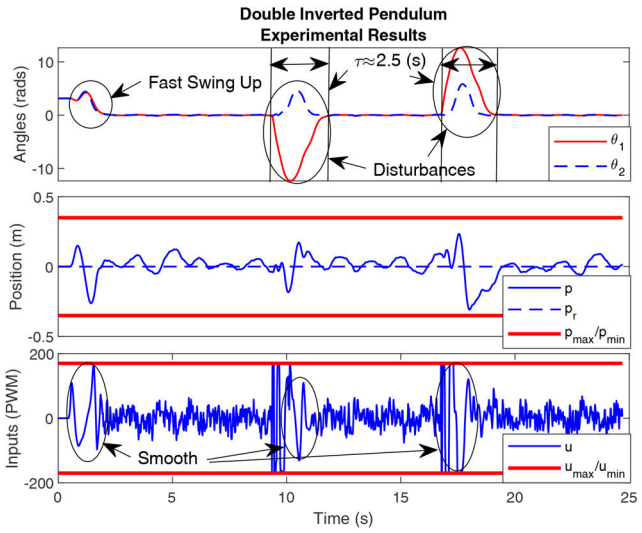


Fig. 6 *Example performance with initial condition steady at lower equilibrium and large disturbances at $t \simeq 9\,(s)$ and $t \simeq 17\,(s)$*

gathered in Table 1, and the input-output data is available in https://doi.org/10.24433/CO.8048147.v1 [14]. Notice the theoretical relationship $c_i = b_i/g$ stated in Section 2 is very close to the one observed in the resulting parameters. Finally, although the system was only tested for OSI, it could work using the available adaptation mechanism provided proper rules are used to avoid the periods of poor excitation, as discussed in [20].

6.2 Swing up, stabilisation and disturbance rejection

Regarding the optimisation setup, the cart has a maximum range for the position of $-0.35 < x < 0.35$ and the PWM input was constrained to $-200 < u_{\text{PWM}} < 200$ despite the actual maximum being 255 (8-bit) to avoid wearing of the DC motor which defined the constraints to be included in the optimisation. Furthermore, the DC motor presented a dead-zone non-linearity of $u_{\text{dz}} \simeq 30$ which was removed by implementing the conditional function (28), adjusting constraints to $-200 + u_{\text{dz}} < u < 200 - u_{\text{dz}}$ and using u in the relevant models to simulate and linearise the system.

$$u_{PWM} = \begin{cases} u + u_{dz}, & u > 0 \\ u - u_{dz}, & u < 0 \end{cases} \tag{28}$$

The prediction horizon for the NMPC was set at $N_p = 75\,(T_p = 1.5\,(\text{s}))$ leading to 600 outputs, 75 decision variables and 300 constraints to be optimised. The output and input weights were selected as $q_{k+i} = \text{diag}([1, 0.1, 0.1, 100, q_{\theta_1}, q_{\theta_2}, 10, 10])\,\forall i = [1, N_p - 1]$ and $R = 0.003I$ which were observed to give good balanced between fast swing up performance and input chattering due to noise at the steady state. A terminal weight $q_{k+N_p} = 10q_{k+i}$ was selected for the last values of the prediction horizon when $t_{lin} > 2\,(\text{s})$, emulating soft zero terminal constraints to improve the stability characteristics of the optimisation [6]. Moreover, a tailored C++ code available in https://doi.org/10.24433/CO.8048147.v1 [14] was developed using EIGEN library following suggestions of [4, 6], and was tested in a laptop running Ubuntu 18.04 with an Intel i7-5700 HQ @ 2.7 GHz giving computation times of $t_{\text{unc}} < 800\,\mu\text{s}$ for the unconstrained solution and $t_{\text{con}} < 2500\,\mu\text{s}$ for the constrained one when doing 10 iterations of an efficient version of Hildreth's QP found in [15]. Finally, the EKF weights were set at $Q_{EKF} = \text{diag}([0.1, 0.1, 0.1, 0.0001, 0.0001, 0.0001, 1])$ and $R_{EKF} = \text{diag}([0.0001, 0.0001, 0.0001, 1])$ based on the variance of the errors observed in an offline analysis of the system identification process.

The resulting performance of the overall scheme can be seen in Fig. 6 starting from the rest position at the lower equilibrium and introducing large disturbances at $t \simeq 9\,(\text{s})$ and $t \simeq 17\,(\text{s})$. As it can be seen, the system clearly exhibits much faster performance than [11] giving settling times of $\tau_s \simeq 1.4\,(\text{s})$ for the swing up manoeuvre and of $\tau \simeq 2.5\,(\text{s})$ after large disturbances. Moreover, the system presented smooth input shapes during the swing up phases as a result of the added energy costs and the hybrid switching scheme. Furthermore, notice the position constraint is clearly satisfied at $t \simeq 17\,(\text{s})$ after the disturbance was given, demonstrating good handling of the rapid active-set changes by the QP. Finally, in some cases the position presented small steady-state error and the well-known limit cycle, however, this can be removed using standard methods such as integral control or disturbance estimation methods which are not the focus of the paper, and therefore were omitted. For the interest of the reader, an overall video is provided in (https://youtu.be/7E-SXi3YKQohttps://youtu.be/7E-SXi3YKQo) where the results can be seen, and the input-output data is available in https://doi.org/10.24433/CO.8048147.v1 [14].

7 Conclusion

This paper presents a novel NMPC approach based on the RTI scheme for the swing-up and stabilisation of a parallel double inverted pendulum with experimental validation. The approach uses dual-mode closed loop predictions for the state deviation model to cancel the unstable open-loop dynamics of the double inverted pendulum which improve the numerical conditioning of the optimisation. For this particular system, the proposed approach was observed to have a condition number 6 orders of magnitude lower than the standard solution which can be critical for matrix inversion when using low precision computing. Moreover, two important modifications were introduced for the improvement of the RTI scheme in the presence of large disturbances, namely; additional energy-related costs and a hybrid switching scheme. The aforementioned modifications were observed to produce much smoother responses when compared to the standard single-shooting RTI NMPC. The approach was able to compute approximate constrained solutions in $t_c < 2500(\mu\text{s})$, and a C++ code implementing it can be found in https://doi.org/10.24433/CO.8048147.v1 [14]. Finally, the approach was combined with an OSI Scheme based on RLS to address parameter uncertainty.

An overall video of the resulting performance is provided in (https://youtu.be/7E-SXi3YKQohttps://youtu.be/7E-SXi3YKQo), and the data obtained throughout the tests is available in https://doi.org/10.24433/CO.8048147.v1 [14].

To the best of the authors' knowledge, this is the first contribution presenting numerical and experimental results for the swing-up and stabilisation of a parallel double inverted pendulum in the presence of large disturbances based on NMPC using the RTI method. Future work will include the extension to the multiple-shooting scheme along with further analysis of the closed-loop performance with adaptation mechanism active as well as offset-free methods to cancel possible input-output disturbances.

8 Acknowledgments

This work was funded by the CONACyT, Mexico.

9 References

[1] Gros, S., Zanon, M., Quirynen, R., *et al.*: 'From linear to nonlinear MPC: bridging the gap via the real-time iteration', *Int. J. Control*, 2016, **7179**, (November), pp. 1–19

[2] Seki, H., Ooyama, S., Ogawa, M.: 'Nonlinear model predictive control using sucessive linearization - application to chemical reactors', *Trans. Soc. Instrum. Control Eng.*, 2004, **E-3**, (1), pp. 66–72

[3] Houska, B., Ferreau, H.J., Diehl, M.: 'Autogenerating microsecond solvers for nonlinear MPC: a tutorial using ACADO integrators', *Optim. Control Appl. Methods*, 2015, **36**, pp. 685–704

[4] Vukov, M., Domahidi, A., Ferreau, H.J., *et al.*: 'Auto-generated algorithms for nonlinear model predictive control on long and on short horizons'. 52nd IEEE Conf. on Decision and Control, Florence, Italy, 2014

[5] Diehl, M., Bock, H.G., Schlöder, J.P.: 'A real-time iteration scheme for nonlinear optimization in optimal feedback control', *SIAM J. Control Optim.*, 2005, **43**, (5), pp. 1714–1736

[6] Houska, B., Ferreau, H.J., Diehl, M.: 'An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range', *Automatica*, 2011, **47**, (10), pp. 2279–2285

[7] Kalmari, J., Backman, J., Visala, A.: 'A toolkit for nonlinear model predictive control using gradient projection and code generation', *Control Eng. Pract.*, 2015, **39**, pp. 56–66
[8] Gros, S., Quirynen, R., Diehl, M.: 'Aircraft control based on fast non-linear MPC & multiple-shooting', *Proc. IEEE Conf. Decis. Control*, 2012, **1**, pp. 1142–1147
[9] Quirynen, R., Vukov, M., Diehl, M.: 'Multiple shooting in a microsecond'. Multiple Shooting and Time Domain Decomposition Methods, Cham, 2015, vol. 9, pp. 183–202
[10] Glück, T., Eder, A., Kugi, A.: 'Swing-up control of a triple pendulum on a cart with experimental validation', *Automatica*, 2013, **49**, (3), pp. 801–808
[11] Alamir, M., Murilo, A.: 'Swing-up and stabilization of a twin-pendulum under state and control constraints by a fast NMPC scheme', *Automatica*, 2008, **44**, (5), pp. 1319–1324
[12] Mills, a., Wills, A., Ninness, B.: 'Nonlinear model predictive control of an inverted pendulum'. American Control Conf., St. Louis, MO, USA, 2009
[13] Rossiter, J.A., Kouvaritakis, B., Rice, M.J.: 'A numerically robust state-space approach to stable-predictive control strategies', *Automatica*, 1998, **34**, (1), pp. 65–73
[14] Villarreal, O. J. G.: 'A Fast Dual Mode NMPC for Parallel Double Inverted Pendulum'. Code Ocean, September 2019. Available at https://doi.org/10.24433/CO.8048147.v1.
[15] Wang, L.: '*Model predictive control system design and implementation using matlab*' (Springer-Verlag London Limited, UK, 2009)
[16] Rossiter, J.A.: '*A first course in predictive control*' (CRC Press, Taylor & Francis, Boca Raton, 2018, 2nd edn.)
[17] He, D., Wang, L., Yu, S.: 'A new dual-mode NMPC scheme with terminal control laws of free-parameters'. Proc. of the 33rd Chinese Control Conf. CCC 2014, Nanjing, People's Republic of China, 2014, pp. 7667–7672
[18] Diehl, M., Findeisen, R., Allgöwer, F.*, et al.*: 'Nominal stability of real-time iteration scheme for nonlinear model predictive control', *IEE Proc., Control Theory Appl.*, 2005, **152**, (3), pp. 296–308
[19] Anderson, S.R., Kadirkamanathan, V.: 'Modelling and identification of non-linear deterministic systems in the delta-domain', *Automatica*, 2007, **43**, (11), pp. 1859–1868
[20] Campbell, S.F., Nguyen, N.T., Kaneshige, J.*, et al.*: 'Parameter estimation for a hybrid adaptive flight controller'. AIAA Infotech@ Aerospace Conf., Seattle, WA, no. April, 2009, pp. 1–27
[21] Gonzalez Villarreal, O.J., Rossiter, J.A., Shin, H.: 'Laguerre-based adaptive MPC for attitude stabilization of quad-rotor'. 2018 UKACC 12th Int. Conf. on Control, CONTROL 2018, Sheffield, UK, 2018

Model Predictive Control for Wave Energy Converters: A Moving Window Blocking Approach

Juan Guerrero-Fernández * **Oscar J. González-Villarreal** *
John Anthony Rossiter * **Bryn Jones** *

* *Department of Automatic Control and Systems Engineering, University of Sheffield, UK (e-mail: j.guerrero@sheffield.ac.uk; ojgonzalezvillarreal1@sheffield.ac.uk; j.a.rossiter@sheffield.ac.uk; b.l.jones@sheffield.ac.uk)*

Abstract: Ocean wave energy is one of the most concentrated sources of renewable energy. However, until now it has not reached the economic feasibility required to be commercialised. To improve the efficiency of wave energy converters several advanced control strategies have been proposed, including Model Predictive Control (MPC). Nevertheless, the computational burden of each optimisation problem is a drawback of conventional (Full-DoF) MPC, which typically limits its application for real-time control of systems. In this paper, a Moving Window Blocking (MWB) approach is proposed to speed-up the time required for each optimisation problem by reducing the number of decision variables. Numerical simulation of a single device point absorber wave energy converter controlled by this scheme confirms the potential of this approach.

Keywords: Wave energy converters, Model predictive controller, Moving window blocking.

1. INTRODUCTION

Ocean wave energy is one of the most concentrated renewable energy sources, and its resources are huge in a many of countries around the globe (Sheng, 2019). The estimated worldwide potential of ocean wave power is 32 000 TW h (Mørk et al., 2010), which is more than the worldwide electricity consumption of about 25 721 TW h (International Energy Agency, 2019).

The development and implementation of wave energy converters (WEC) may have several benefits, especially for those countries having abundant wave energy resources. Examples of the benefits range from individual benefits for the country such as increasing of their renewable energy matrix and guaranteeing energy supply diversity (Sheng, 2019), to global benefits by confronting the problems of climate change and the difficult challenge of reducing the dependency on conventional energy resources such as fossils or nuclear energy.

To date, wave energy technologies are technically immature for reliable and economical energy generation (Sheng, 2019). One of the biggest challenges is how to improve the efficiency of wave energy converters. To address this issue, several control strategies have been proposed to alter the dynamic behaviour of the device in order to maximise the extracted energy. *Model Predictive Control* (MPC) is a well-developed control strategy within Academia and Industry communities which takes into account constraints whilst optimising a given cost function (Faedo et al., 2017). Although MPC can have explicit offline solutions (H.J.Ferreau, H.G. Bock, 2008), this is not tractable for the WEC problem given the large amount of variation present in the wave excitation forces which are external disturbances to the optimisation. Thus, for this application, MPC requires an online solution where at each sampling time, solves an Optimal Control Problem (OCP) to produce an optimal control sequence, the first of which is applied to the plant as the control action (Li and Belmont, 2014). However, one of the drawbacks of MPC is the computational burden required to solve the OCP.

To reduce the computational burden of the optimisation, a popular approach is to use input-parameterisation techniques which allow to reduce the number of degrees of freedom of the optimisation. Several input-parameterisation have been proposed such as Laguerre Polynomials (Wang, 2004), as well as orthonormal parameterisations based on collocation points, typically referred as pseudospectral methods (Garcia-Violini and Ringwood, 2019).

In this paper, a *Moving Window Blocking* (MWB) MPC approach is proposed with the idea of reducing the computational time required to solve the OCP at each sampling time, and the resulting performance is compared with the Full-DoF MPC strategy and Generalised Predictive Control (GPC).

The remaining part of this paper is organised as follows: Section 2 presents the mathematical model for the WEC considered in this study. Full-DoF Model Predictive Control and, more specifically, a detailed description of the proposed Moving Window Blocking MPC approach is given in Section 3. The results of the simulations are presented and commented in Section 4. Finally, the conclusions and future work are set out in Section 5.

2. WEC MODELLING

For the development of the mathematical model of a wave energy converter (WEC), a heaving semi-submerged sphere is considered as in Figure 1. The hydrodynamic model is developed from first principles. Applying Newton's second law to the partially submerged sphere, the dynamics of the sphere are described by:

$$m\,\ddot{z}(t) = F_g - \iint_{S(t)} P(t)\,\mathbf{n}\,\mathrm{d}S \; + F_{PTO}(t) \tag{1}$$

Where m is the floater mass, z is the vertical displacement of the body relative to its hydrostatic equilibrium position, F_g is the force due to gravity, $F_{PTO}(t)$ is the force exerted by the Power Take-Off systems (PTO) (controller input $u(t)$), $P(t)$ is the pressure on an element $\mathrm{d}S$ on the buoy wetted surface, $\mathbf{n}$ is a vector normal to the surface element, $\mathrm{d}S$ and S is the submerged wetted surface.

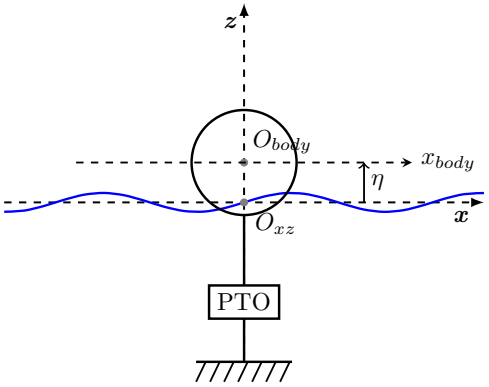


Fig. 1. A general wave energy converter with 1-DoF: heave

From (1), several models can be derived, depending on the complexity, computational time and accuracy desired. In this study, a linear hydrodynamic model is considered. For linear models, assuming the fluid is in-compressible, in-viscid and irrotational [1], (1) is typically solved using potential flow theory, in which the potential problem is linearised and computed around the position of equilibrium. Considering small displacements, and seabed as reference system, (1) is rewritten as follows:

$$m\,\ddot{z}(t) = F_{res}(t) + F_{rad}(t) + F_{exc}(t) + F_{PTO}(t), \tag{2}$$

where $F_{res}(t)$ is hydrostatic restoring force, $F_{rad}(t)$ is the radiation force, and $F_{exc}(t)$ the excitation force due to the incoming wave. The hydrostatic restoring force $F_{res}(t)$ represent the spring-like effect of the surrounding ocean water into the buoy, and is determined by k_h hydrostatic stiffness and $z(t)$ absorber position:

$$F_{res}(t) = -k_h\,z(t) \tag{3}$$

The excitation force $F_{exc}(t)$ describes the interactions between the incident waves and the body at its place of equilibrium, and is represented by the convolution of the excitation impulse response k_{exc} with the otherwise undisturbed free-surface elevation η at the centre of the body:

$$F_{exc}(t) = \int_{-\infty}^{t} k_{exc}(t-\tau)\,\eta(\tau)\,d\tau \tag{4}$$

[1] This is a standard assumption in the wave energy literature (Faedo et al., 2017).

Similarly, the radiation force $F_{rad}(t)$ is a damping/inertial force associated with waves radiated by the absorber oscillating in calm water scenario, and is expressed by the added mass μ_∞ and the convolution product between the radiation impulse response k_{rad} and the absorber velocity $\dot{z}(t)$:

$$F_{rad}(t) = -\mu_\infty\,\ddot{z}(t) - \int_{-\infty}^{t} k_{rad}(t-\tau)\,\dot{z}(\tau)\,d\tau \tag{5}$$

The convolution kernels k_{exc}, k_{rad} and the frequency-independent added mass μ_∞ are computed numerically using boundary element methods (BEMs). In this study the open source *NEMOH* (Penalba et al., 2017) was employed. Combining (3)-(5) with (2) gives the widely used equation (in WEC studies) Cummins' equation (Cummins, 1962):

$$\begin{aligned} m\,\ddot{z}(t) = &- k_h z(t) - \mu_\infty \ddot{z}(t) - \int_{-\infty}^{\infty} k_{rad}(t-\tau)\dot{z}(\tau)d\tau \\ &+ \int_{-\infty}^{\infty} k_{exc}(t-\tau)\eta(\tau)d\tau + F_{PTO}(t) \end{aligned} \tag{6}$$

At this point, a few statements can be made from (6). First, since the excitation force $F_{exc}(t)$ depends on the undisturbed free-surface elevation $\eta(t)$, it can be considered as an independent input to the system. Second, (6) is represented in state-space form for control strategy implementation and third, the direct computation of the convolution integral in (5) in time-domain simulation is computationally expensive and cumbersome (Roessling and Ringwood, 2015). To avoid the direct computation of the convolution integral at every time step, several methods to approximate the integral have been proposed (Yu and Falnes, 1995; Roessling and Ringwood, 2015; Pérez and Fossen, 2008). Approximating the convolution integral in (5) by a state-space system with the state vector $\underline{x_r}(t) \in \mathbb{R}^n$ is a common approach, where the input to the system is the velocity of the absorber ($v = \dot{z}$) and the approximation of the convolution integral term of the radiation force is the output:

$$\begin{aligned} \dot{\underline{x_r}}(t) &= A_r\,\underline{x_r}(t) \;+\; B_r\,\dot{z}(t) \\ \textstyle\int_{-\infty}^{\infty} k_{rad}(t-\tau)\dot{z}(\tau)\,d\tau &\approx C_r\,\underline{x_r}(t) \end{aligned} \tag{7}$$

This system is later included as a part of the overall model that describes the motion of the absorber. It is important to clarify that the system states in (7) have no physical meaning, but still contain information on the condition of the surrounding fluid (Cretel et al., 2011). In this study the state space matrices A_r, B_r, and C_r were computed by the open source toolbox *FOAMM* (Finite Order Approximation by Moment-Matching, based on the theoretical foundations presented in Faedo et al. (2018)) Defining the state and output vectors, $\underline{x_c} \in \mathbb{R}^{n+2}$ and $\underline{y_c} \in \mathbb{R}^2$, for the linear time-invariant state-space system:

$$\underline{x_c} = \begin{bmatrix} z \\ \dot{z} \\ \underline{x_r} \end{bmatrix} \qquad \underline{y_c} = \begin{bmatrix} z \\ \dot{z} \end{bmatrix} \tag{8}$$

the whole dynamics of the WEC is given by:

$$\begin{aligned} \dot{\underline{x_c}}(t) &= A_c\,\underline{x_c}(t) + B_c\,F_{pto}(t) + B_c\,F_{exc}(t) \\ \underline{y_c}(t) &= C_c\,\underline{x_c}(t) \end{aligned} \tag{9}$$

in which $A_c \in \mathbb{R}^{(n+2)\mathrm{x}(n+2)}$, $B_c \in \mathbb{R}^{(n+2)\mathrm{x}1}$, $C_c \in \mathbb{R}^{2\mathrm{x}(n+2)}$, are defined as:

$$A_c = \begin{bmatrix} 0 & 1 & \mathbf{0} \\ \frac{-k_h}{m+\mu_\infty} & 0 & \frac{-C_r}{m+\mu_\infty} \\ \mathbf{0} & B_r & A_r \end{bmatrix} \quad B_c = \begin{bmatrix} 0 \\ 1 \\ \mathbf{0} \end{bmatrix} \quad C_c = \begin{bmatrix} 1 & 0 & \mathbf{0} \\ 0 & 1 & \mathbf{0} \end{bmatrix}$$

where $\mathbf{0}$ denotes a zero matrix of required dimensions.

By discretising system (9), and replacing F_{pto} and F_{exc} for u and u_{exc}, respectively, to use standard nomenclature, results in a general discrete state-space of the form.

$$\underline{x}_{k+1} = A_d \underline{x}_k + B_d \underline{u}_k + B_d \underline{u}_{exc_k} \tag{10a}$$
$$\underline{y}_k = C_d \underline{x}_k \tag{10b}$$

For this study, a discretisation of a zero-order hold was considered using a sampling time of $T_s = 0.1s$. The resulting state space matrices are given in section 4.

3. MODEL PREDICTIVE CONTROL

3.1 General Objective

In this paper, Model Predictive Control was used as general optimal control methodology with the general purpose of maximising the mechanical energy E_{abs} absorbed by the PTO system over a time horizon, defined as:

$$E_{abs} = -\int_t^{t+T} u(\tau)\dot{z}(\tau)d\tau \tag{11}$$

Furthermore, real WEC systems will typically present position, input and input increments (slew rates) constraints related to physical limits which can be handled naturally by the MPC formulation. To benefit from the moving window blocking approach presented in subsection 3.4, this paper focuses particularly on the case where the WEC is within a "safe" operating region (operating within the position constraints, but without making contact with the end-stops). The device should be locked in a survival mode when exposed to extreme sea conditions (Sheng, 2019); this is reasonable given it is generally not possible to guarantee output feasibility (such as the buoy positions) for dynamics systems under significant disturbances. In simple terms, if a big enough wave is applied to the system, it might not even be possible to prevent it from reaching the limits, regardless of the input selection. An alternative might be to use soft-constraints for some output violations, however, this is out of the scope of this paper.

The discrete-time objective function is thus chosen as:

$$min \quad J_k = \sum_{i=1}^{N_p} u_{k+i-1}\dot{z}_{k+i} \tag{12a}$$
$$s.t. \quad u_{min} \leq u_{k+i-1} \leq u_{max} \tag{12b}$$
$$\Delta u_{min} \leq \Delta u_{k+i-1} \leq \Delta u_{max} \tag{12c}$$

where N_p is the prediction horizon. Note that this cost considers the force u and velocity $\dot{z}$ at different time steps ($k+i-1$ and $k+i$). This is chosen to ensure causality of the solution as discussed in Li and Belmont (2014).

3.2 Predictions

Following the methodology described in Cretel et al. (2010), the state space model (10) is augmented with the previous input u_{k-1} to use the input increment Δu_k as the decision variable resulting in:

$$x_{k+1} = Ax_k + B\Delta u_k + B_w u_{exc_k} \tag{13a}$$
$$y_k = Cx_k \tag{13b}$$

where the state is now $x_k = [\underline{x}_k^T u_{k-1}]^T \in \mathbb{R}^{n+3}$, the output is $y_k = [\underline{y}_k^T u_{k-1}]^T \in \mathbb{R}^3$, and

$$A = \begin{bmatrix} A_d & B_d \\ \mathbf{0} & 1 \end{bmatrix} \quad B = \begin{bmatrix} B_d \\ 1 \end{bmatrix} \quad B_w = \begin{bmatrix} B_d \\ 0 \end{bmatrix} \quad C = \begin{bmatrix} C_d & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}$$

This change will allow simple expressions for input and input rate constraints, as well as the computation of the product $(u_{k+i-1}\dot{z}_k)$ through an appropriate matrix Q as discussed in Cretel et al. (2010, 2011). By propagating the model (13a) N_p times forward, all future outputs $\hat{Y} = [y_{k+1}^T, y_{k+2}^T, \cdots, y_{k+N_p}^T]^T \in \mathbb{R}^{3N_p}$ are given by:

$$\hat{Y} = Gx_k + H\Delta\hat{U} + H_w\hat{U}_w \tag{14}$$

where $\Delta\hat{U} = [\Delta\hat{u}_k, \Delta\hat{u}_{k+1}, \cdots, \Delta\hat{u}_{k+N_p}]^T \in \mathbb{R}^{N_p}$ are the future input increments; $\hat{U}_w = [\hat{u}_{w_k}, \hat{u}_{w_{k+1}}, \cdots, \hat{u}_{w_{k+N_p}}]^T \in \mathbb{R}^{N_p}$ are the future wave excitation forces; and matrices $G \in \mathbb{R}^{3N_p \times (n+3)}$ and $H \in \mathbb{R}^{3N_p \times N_p}$ are defined as:

$$G = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^{N_p} \end{bmatrix}^T \quad H = \begin{bmatrix} CB & \mathbf{0} & \cdots & \mathbf{0} \\ CAB & CB & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{0} \\ CA^{N_p-1}B & \cdots & CAB & CB \end{bmatrix}$$

where $\mathbf{0}$ are zeros matrices with the same dimensions of CB, and H_w is defined as H using B_w instead.

3.3 Standard Optimisation

Having defined the prediction models, a standard quadratic cost function can be formulated as,

$$J = \frac{1}{2}\hat{Y}^T Q \hat{Y} \tag{15}$$

To compute the product $(u_{k-1}\dot{z}_k)$, the penalisation matrix $Q \in \mathbb{R}^{3N_p \times 3N_p}$ is selected as a block diagonal matrix with the inner matrices q_{k+i} defined as,

$$Q = \begin{bmatrix} q_{k+1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & q_{k+2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & q_{k+N_p} \end{bmatrix} \quad q_{k+i} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \forall i = [1, N_p] \tag{16}$$

It is interesting to note that although matrix Q is not positive definite, which is a common requirement for standard MPC problems, it does result in a positive definite Hessian as defined in (17) for the WEC model, which results in a convex optimisation problem.

By substituting the output predictions (14) in (15), rearranging in terms of the decision variables ($\Delta\hat{U}$), and including input and input rate constraints, the standard quadratic program (17) is obtained.

$$J = \frac{1}{2}\Delta\hat{U}^T E \Delta\hat{U} + \Delta\hat{U}^T f \quad s.t. \quad M\Delta\hat{U} \leq b \tag{17a}$$
$$E = H^T Q H \qquad f = H^T Q(Gx_k + H_w\hat{U}_w) \tag{17b}$$
$$M = \begin{bmatrix} I \\ -I \\ D \\ -D \end{bmatrix} \qquad b = \begin{bmatrix} \Delta u_{max}\mathbf{1} \\ -\Delta u_{min}\mathbf{1} \\ (u_{max} - u_{k-1})\mathbf{1} \\ (-u_{min} + u_{k-1})\mathbf{1} \end{bmatrix} \tag{17c}$$

where $E \in \mathbb{R}^{N_p \times N_p}$ is a matrix known as the Hessian; $f \in \mathbb{R}^{N_p}$ is a column-vector; $M \in \mathbb{R}^{4N_p \times N_p}$ is the constraint matrix; $b \in \mathbb{R}^{4N_p}$ is the constraint vector; $I \in \mathbb{R}^{N_p \times N_p}$ is an identity matrix; $D \in \mathbb{R}^{N_p \times N_p}$ is a lower triangular matrix; and $\mathbf{1} \in \mathbb{R}^{N_p}$ column-vector is a column vector of ones.

Having defined E, f, M, b, the optimisation can then be solved using any QP solver such as quadprog function of Matlab, QP OASES (H.J.Ferreau, H.G. Bock, 2008), etc. At each sampling time, only the first input is applied to the system and the process is repeated, which is the well known receding horizon control strategy.

3.4 Moving Window Blocking

In this paper, we used a blocking approach where the input is parameterised in blocks of size N_b having equal values, e.g. $u_k = u_{k+1} = \cdots = u_{k+N_b-1}$ for the first block, $u_{k+N_b} = u_{k+N_b+1} = \cdots = u_{k+2N_b-1}$ for the second block, etc., thus allowing the decision variables to be spread over the prediction horizon, as opposed to the standard Generalized Predictive Control (GPC) approach where the decision variable are "congested" at the beginning, and left constant after a "control horizon" (Rossiter, 2018). An example comparison of this is visualised in Fig. 2 for the WEC system defined in section (2), and is further discussed in the results section. This distinctive feature of the blocking approach is important for this application for two main reasons: firstly, the solution obtained from the original problem using full degrees of freedom applied to the WEC system is constantly saturated as seen in Fig. 2, thus can be accurately represented by blocks; and secondly, depending on the wave future values, it might be more important to have decisions available at the future, example when the wave reaches its crest and trough (maximum/minimum values).

The aforementioned blocking parameterisation can be achieved by defining a blocking matrix ($\mathbb{N}$) for the decision variables ($\Delta\hat{U}$) of the form:

$$\Delta\hat{U} = \mathbb{N}\Delta\hat{\mathbb{U}} \tag{18a}$$

$$\mathbb{N} = \begin{bmatrix} \mathbf{n} & \mathbf{0}_{N_b} & \cdots & \mathbf{0}_{N_b} \\ \mathbf{0}_{N_b} & \mathbf{n} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{0}_{\mathbf{n}} \\ \mathbf{0}_{N_b} & \cdots & \mathbf{0}_{N_b} & \mathbf{n} \end{bmatrix} \quad \mathbf{n} = \begin{bmatrix} 1 \\ \mathbf{0}_{N_b-1} \end{bmatrix} \tag{18b}$$

where $\hat{\mathbb{U}} \in \mathbb{R}^{N_u}$ are the blocked decision variables which have reduced dimensions of $N_u = \lceil \frac{N_p}{N_b} \rceil \mathbf{n} \in \mathbb{R}^{N_b}$, and $\mathbf{0}_v \in \mathbb{R}^v$ is a column-vector of v zeros. For simplicity, N_p should be selected as a multiple integer of the block size N_b, otherwise the last $\mathbf{n}$ in the diagonal might be different.

Moreover, as discussed in Cagienard et al. (2007), the application of standard blocking approaches has an inconsistent nature, and suffers from recursive feasibility problems given the decision in the previous time step cannot be replicated which is detrimental to the performance. To address this, the Moving Window Blocking (MWB) approach developed in Cagienard et al. (2007) proposed to shift the set of N_b admissible blocking matrices $\mathbb{N}_i$ along with the moving horizon resulting in an input parameterisation of the form.

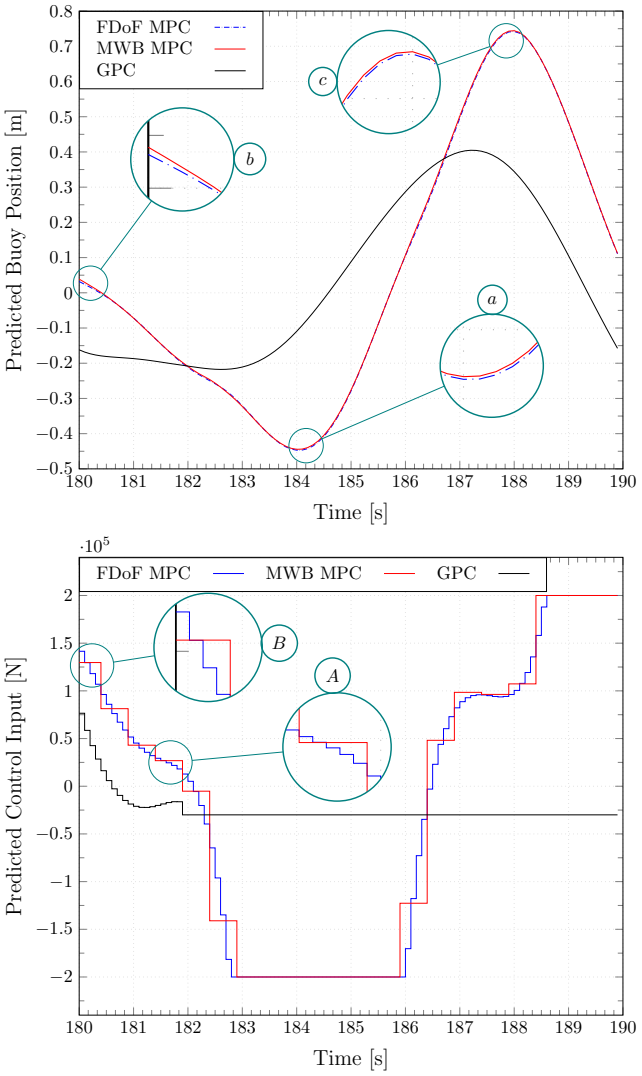


Fig. 2. Predicted Trajectories for Buoy Position z and Input u.

$$\Delta\hat{U} = \mathbb{N}_i\Delta\hat{\mathbb{U}} \qquad \forall i = [1, N_b] \tag{19a}$$

$$\mathbb{N}_i = \begin{bmatrix} \mathbf{n}_1 & \mathbf{0}_{N_b-i} & \cdots & \mathbf{0}_{N_b-i} \\ \mathbf{0}_{\mathbf{n}} & \mathbf{n} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{0}_{N_b} \\ \mathbf{0}_{N_b-2+i} & \cdots & \mathbf{0}_{N_b-2+i} & \mathbf{n}_f \end{bmatrix} \quad \mathbf{n}_1 = \begin{bmatrix} 1 \\ \mathbf{0}_{N_b-i} \end{bmatrix} \quad \mathbf{n}_f = \begin{bmatrix} 1 \\ \mathbf{0}_{N_b-2+i} \end{bmatrix} \tag{19b}$$

where $\mathbf{n}$ and $\mathbf{0}_v$ are defined as in (18). Notice the first and final block $(\mathbf{n}_1, \mathbf{n}_f)$ are shrinking and expanding, respectively. This parameterisation is then applied sequentially $i = 1 \rightarrow N_b$ until the first block reaches its limit, and resets to its original size ($i = 1$).

By substituting the MWB input parameterisation in the standard quadratic program (17), the application of the MWB approach then leads to formulating and solving N_b different quadratic programs sequentially and repeating infinitely $i = 1 \rightarrow N_b, 1 \rightarrow N_b, 1 \rightarrow \cdots$ as the horizon moves forward defined as:

$$J = \frac{1}{2}\Delta\hat{\mathbb{U}}^T E_{\mathbb{N}}^{[i]}\Delta\hat{\mathbb{U}} + \Delta\hat{\mathbb{U}}^T f_{\mathbb{N}}^{[i]} \quad s.t. \quad M_{\mathbb{N}}^{[i]}\Delta\hat{\mathbb{U}} \leq b \quad (20a)$$

$$E_{\mathbb{N}}^{[i]} = \mathbb{N}_i^T H^T Q H \mathbb{N}_i = \mathbb{N}_i^T E \mathbb{N}_i \quad (20b)$$

$$f_{\mathbb{N}}^{[i]} = \mathbb{N}_i^T H^T Q(Gx_k + H_w\hat{U}_w) = \mathbb{N}_i^T f \quad (20c)$$

$$M_{\mathbb{N}}^{[i]} = \begin{bmatrix} \mathbb{N}_i \\ -\mathbb{N}_i \\ D\mathbb{N}_i \\ -D\mathbb{N}_i \end{bmatrix} \quad b = \begin{bmatrix} \Delta u_{max}\mathbf{1} \\ -\Delta u_{min}\mathbf{1} \\ (u_{max} - u_{k-1})\mathbf{1} \\ (-u_{min} + u_{k-1})\mathbf{1} \end{bmatrix} \quad (20d)$$

where $E_{\mathbb{N}}^{[i]} \in \mathbb{R}^{N_u \times N_u}$ is the "compressed" Hessian, which can be pre-stored for faster computations. On the other hand, the "compressed" linear term $f_{\mathbb{N}}^{[i]} \in \mathbb{R}^{N_u}$ can also be pre-stored by separating the values in $f_{\mathbb{N}}^{[i]} = f_{1_{\mathbb{N}}}^{[i]} x_k + f_{2_{\mathbb{N}}}^{[i]} \hat{U}_w$ with $f_{1_{\mathbb{N}}}^{[i]} = \mathbb{N}_i^T H^T Q G$ and $f_{2_{\mathbb{N}}}^{[i]} = \mathbb{N}_i^T H^T Q H_w$. Moreoever, it is trivial to derive that when using the blocking matrix $\mathbb{N}_i$ as defined in (19), the constraint matrix have redundant zero rows $\forall i$, and can be reduced to,

$$M_{\mathbb{N}}^{[i]} = M_{\mathbb{N}} = \begin{bmatrix} I_{\mathbb{N}} \\ -I_{\mathbb{N}} \\ D_{\mathbb{N}} \\ -D_{\mathbb{N}} \end{bmatrix} \quad b_{\mathbb{N}} = \begin{bmatrix} \Delta u_{max}\mathbf{1}_{\mathbb{N}} \\ -\Delta u_{min}\mathbf{1}_{\mathbb{N}} \\ (u_{max} - u_{k-1})\mathbf{1}_{\mathbb{N}} \\ (-u_{min} + u_{k-1})\mathbf{1}_{\mathbb{N}} \end{bmatrix} \quad (21)$$

where $M_{\mathbb{N}} \in \mathbb{R}^{4N_u \times N_u}$ is the "reduced" constraint matrix, $b_{\mathbb{N}} \in \mathbb{R}^{4N_u}$ is the "reduced" constraint vector, $I_{\mathbb{N}} \in \mathbb{R}^{N_u \times N_u}$ is an identity matrix, $D_{\mathbb{N}} \in \mathbb{R}^{N_u \times N_u}$ is a lower triangular matrix, and $\mathbf{1}_{\mathbb{N}} \in \mathbb{R}^{N_u}$ is column-vector of ones, all of which have reduced dimensions $N_u = \lceil \frac{N_p}{N_b} \rceil$ when compared to the original constraint terms (17c), thus can lead to significant computational benefits as discussed in the results section. Once the optimisation is solved, the original decision vector can be recovered using (19).

4. RESULTS

In this section, we present the simulation results of the control of a point-absorber WEC using Full-DoF MPC, GPC and the proposed Moving Window Blocking (MWB) MPC approach. The WEC model considered is a heaving semi-submerged sphere reacting against a fixed reference (see Fig. 1), with a radius of 5 m and draft of 5 m, mass $m = 2.6831 \times 10^5$ kg placed in deep water. A sampling time of $T_s = 0.1s$ was used. The hydrodynamic coefficients were computed using the open source *NEMOH* (Penalba et al., 2017). The convolution integral in the radiation force (5) is approximated by a state-space model of order 6 (See (7)). Here the state-space matrices are computed using the toolbox *FOAMM*, which is based in the moment-matching method (Faedo et al., 2018). The resulting state space matrices for the discretised model of (10) are given by (22). The Matlab code and results presented in this paper are available through a Code Ocean compute capsule `https://doi.org/10.24433/CO.0481002.v1` (Guerrero-Fernandez and Gonzalez Villarreal, 2019).

To focus on the comparison of the control strategies, which is the main driver of this study, perfect knowledge of the future wave forces $\hat{U}_w$ and state x_k is considered during the simulation time. The wave elevation of the irregular sea wave was built using the JONSWAP (Joint North Sea Wave Project) spectrum discretised in frequency between 0.02 Hz to 0.80 Hz, corresponding to 1.25 s to 50 s periods respectively, with a frequency step of $\Delta f = 5.2 \times 10^{-3}$ Hz. Considering a significant wave height $H_0 = 2.0$ m and wave peak period $T_p = 10.0$ s. Fig. 3 shows the resulting excitation force on the buoy, with a force range from 8.7119×10^5 N to -8.5133×10^5 N.

Here, it is considered the Full Degrees Of Freedom (Full-DoF) MPC as the control strategy which delivers the maximum possible extracted energy (100% efficiency). For the optimisation setup, a prediction horizon of 10 s ($N_p = 100$) was used with a block size of $N_b = 5$ for the MWB approach which resulted in $N_u = 20$ decision variables. To perform a fair comparison, the GPC approach used the same amount of decision variables compressed at the beginning of the prediction horizon. Moreover, matrix B_d of (22) was re-scaled/normalized to avoid numeric conditioning problems of the optimisation. Finally, constraints on the input and input increment were considered as $||u_{k+i}|| \leq 200kN$ and $||\Delta u_{k+i}|| \leq 200kN \ \forall i = [0, N_p - 1]$, respectively.

Fig. 4 shows the energy extracted for the different controllers studied in this paper, and the final value of the energy extracted at the end of the 600 s simulation is shown in Table 1. The results show that the proposed MWB approach offers almost the same amount of energy compared to the maximum feasible (Full-DoF MPC), with an efficiency of 98.79 %. On the other side GPC is ranked third in the amount of energy extracted, with an efficiency of 92.84 %. Moreover, Zoom *A* in Fig. 4, shows the bidirectional reactive power flowing between the PTO and the absorber, condition required for the active control strategies to maximise the extracted energy (Pecher and Kofoed, 2017).

On an interesting note, it can be seen that Full-DoF unconstrained MPC with input saturation failed to extract energy altogether as seen in Table 1. It is for this precise reason that the whole approach was particularly formulated considering the input and input rate constraints as key parts, which can be captured efficiently using the MWB approach. An alternative is to add an extra quadratic penalization term on the input of the form ($J_\lambda = J + \lambda \sum u_{k+i-1}^2 \ \forall i = [1, N_p]$) to the cost function (12a) as discussed in Li and Belmont (2014), however, this causes disagreements between the terms, inevitable leading to suboptimalities. To perform a fair comparison, a brute-force search was performed to select the value of $\lambda = 1.12$ which achieved the highest energy absorption for the unconstrained (λ) penalised Full-DoF MPC solution with an efficiency of 88.71, thus still resulting in worse performance than both, GPC and MWB.

Table 1. Energy Extracted for 600 s simulation using Full-DoF MPC, MWB MPC, GPC and Full-DoF Unconstrained MPC (with and without additional λ penalization terms)

Method	Energy extracted [MJ]	Efficiency [%]
FDoF MPC	306.974	100
FDoF Unc. MPC	−328.738	LOSS
FDoF Unc. MPC (λ)	272.318	88.71
MWB MPC	303.274	98.79
GPC	285.000	92.84

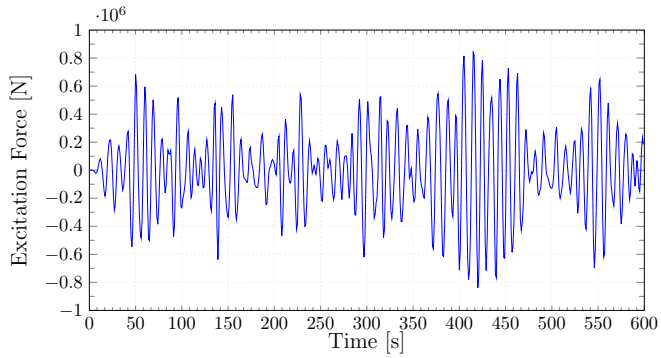


Fig. 3. Excitation force u_{exc_k} for a irregular sea condition built using the JONSWAP spectrum, with wave height $H_0 = 2.0\,\text{m}$ and wave peak period $T_p = 10.0\,\text{s}$

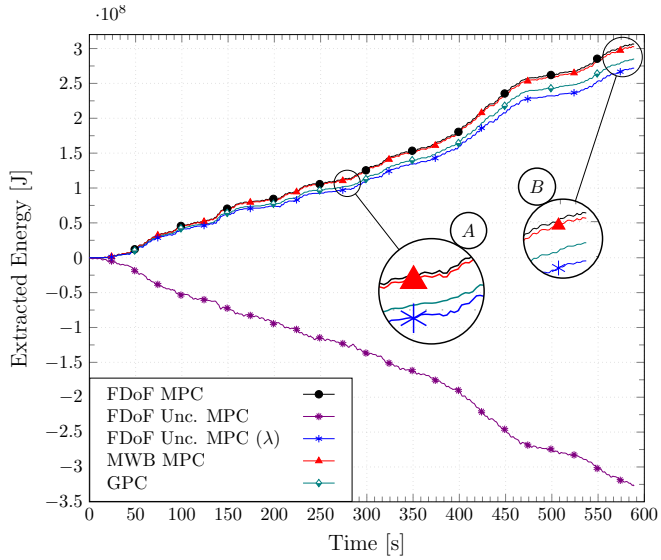


Fig. 4. Energy Extracted tendency for 600 s simulation using Full-DoF MPC, MWB MPC, GPC and Full-DoF Unconstrained MPC (with/without λ terms)

Fig. 2 shows the predicted trajectories of the buoy position and the control input u for the three MPC solutions, namely: the Full-DoF MPC, the MWB and the GPC. From the lower plot of Fig. 2, it can clearly be seen how the MWB approach embeds the blocked parameterisation, distributing the decision variables along the prediction horizon with the sequential shrinking approach (first block size of $4 \rightarrow i = 2$) visible in the zooms A and B, respectively. In contrast, in the GPC approach, all the decision variables are calculated for the beginning of the prediction horizon and kept constant after a certain time which leads to a significant difference in the predicted trajectory of the control action.

On the other hand, the predicted trajectories of the buoy position can be seen in the upper plot of Fig. 2 where the solutions for both, Full-DoF MPC and MWB MPC, are practically indistinguishable, with negligible differences visible in zooms a, b and c. This visual agreement is supported by the suboptimality given in Table 1. In contrast, this can not be said about the GPC approach, where one can see the significant differences in the predicted trajectories, most likely related to the differences in the available control action trajectories. In simple terms, the GPC approach is unable to replicate the position trajectory when using the same number of decision variables compressed in the beginning of the prediction horizon.

With regard to the computation times used to solve the optimal problem at each time step, Table 2 summarises relevant optimisation statistics of each method employed in this study when using the interior point method of Matlab R2018b "quadprog" function in a normal PC with an Intel i5-7500 @ 3.4 GHz CPU, and 8 GB @ 2.4 GHz DDR4 RAM. On average, the proposed MWB approach makes it possible to solve the optimal problem 12.6 times faster compared to the Full-DoF MPC. The reason for this gain in the computation time is due to the fact that, in this case, the number of decision variables and constraints are reduced by 5 times ($N_b = 5$), ie. from $N_p = 100$ to $N_u = 20$ decision variables, and from $4N_p = 400$ to $4N_u = 80$ constraints, which ultimately leads to faster and lower amount of iterations required by the QP to solve the problem. Similar comments of the timing statistics can be made for the GPC strategy, with the main drawback being a performance degradation (efficiency of 92.84 %). Finally, it can be seen that the MWB presented the smallest standard deviation for both average timing statistics, thus leading to an optimisation with more consistent/repeatable behaviour.

Table 2. Statistics of the Optimisation

Method	Avg. opt. time [ms]	Avg. num. of QP iterations	Avg. opt. time per iter. [ms]	Gain
MPC FDoF	19.78 ± 2.75	8.19 ± 0.79	2.42 ± 0.27	-
MWB MPC	1.58 ± 0.23	6.75 ± 0.98	2.37 ± 0.04	12.6
GPC	1.39 ± 0.51	7.37 ± 1.17	1.93 ± 0.10	14.2

5. CONCLUSION

The control strategies presented in this study are intended to maximise the energy production of a generic point-absorber wave energy converter subject to input and input rate constraints related to physical limits. The system benefits from the ability of Model Predictive Control to

$$A_d = \begin{bmatrix} 0.9905 & 0.0997 & -0.0003 & -0.0002 & 0.0003 & 0.0004 & -0.0005 & 0.0015 \\ -0.1896 & 0.9899 & -0.0048 & -0.0049 & 0.0057 & 0.0084 & -0.0096 & 0.0297 \\ -0.0253 & 0.2166 & 0.7789 & 0.2342 & -0.2682 & 0.1451 & -0.2394 & 0.1985 \\ -0.0021 & 0.0171 & -0.0373 & 1.0167 & -0.0214 & 0.0111 & -0.0190 & 0.0156 \\ -0.0361 & 0.3052 & -0.3113 & 0.3019 & 0.5081 & 0.6650 & -0.3376 & 0.2792 \\ 0.0013 & -0.0474 & 0.0476 & -0.0472 & -0.4112 & 0.8425 & 0.0486 & -0.0465 \\ -0.0217 & 0.1850 & -0.1887 & 0.1830 & -0.2292 & 0.1236 & 0.7820 & 0.3332 \\ 0.0015 & -0.0201 & 0.0203 & -0.0199 & 0.0232 & -0.0160 & -0.1424 & 0.9675 \end{bmatrix} \quad B_d = \begin{bmatrix} 0.0123 \\ 0.2465 \\ 0.0329 \\ 0.0028 \\ 0.0469 \\ -0.0017 \\ 0.0282 \\ -0.0019 \end{bmatrix} \cdot 10^{-6} \quad C_d = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}^T \tag{22}$$

include future information of both wave forces and physical constraints. Moreover, to reduce the computational burden, it uses Moving Window Blocking approach where the decision variables are parameterised through a set of input-blocking matrices which result in a sequence of Quadratic Programs of reduced size to be solved sequentially and repeating infinitely. This allows solutions up to 13.10 times faster with sub-optimalities as low as 98.8 % when compared to the Full Degrees of Freedom MPC optimal solution. Although both, the Full-DoF MPC and the proposed MWB MPC approach are computationally feasible for this particular single WEC device model, the proposed control strategy could be a key methodology for implementing Centralised Model Predictive Control for wave farms. The solution of the proposed approach was further compared with GPC, as well as with two version of Unconstrained Full-DoF MPC, one of which was shown to result in a complete loss of energy extraction.

Future work will include the assessment of the solution using real-time embedded hardware such as FPGAs, as well as faster QP solvers such as QP OASES. Moreover, the application will be extended to wave farms using a centralised optimisation framework, and compared with decentralised/distributed approaches as well as with other parameterisation such as collocation points based on pseudospectral methods. Finally, the mathematical models and MPC formulation will be extended to the nonlinear case, and will include further modeling such as actuator dynamics and future wave force predictions.

Ultimately, enhancing peer collaboration and transparency, the findings provided in this paper and the Matlab code used in the simulation are accessible through a Code Ocean compute capsule (`https://doi.org/10.24433/CO.0481002.v1`) (Guerrero-Fernandez and Gonzalez Villarreal, 2019).

ACKNOWLEDGEMENTS

The first author would like to acknowledge the support of MICITT (Ministerio de Ciencia, Tecnología y Telecomunicaciones) of Costa Rica, who funded this work through a scholarship under the contract MICITT-PINN-CON-2-1-4-17-1-027. The second author would like to acknowledge the support of CONACyT, Mexico.

REFERENCES

Cagienard, R., Grieder, P., Kerrigan, E.C., and Morari, M. (2007). Move blocking strategies in receding horizon control. *Journal of Process Control*, 17(6), 563–570.

Cretel, J., Lewis, A.W., Lightbody, G., and Thomas, G.P. (2010). An application of Model Predictive Control to a wave energy point absorber. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 1, 267–272.

Cretel, J.A., Lightbody, G., Thomas, G.P., and Lewis, A.W. (2011). Maximisation of energy capture by a wave-energy point absorber using model predictive control. In *Proceedings of the 18th World Congress The International Federation of Automatic Control*, 3714–3721. IFAC, Milano, Italy.

Cummins, W. (1962). The impulse response fitting and ship motions. Technical report, Institut fuer Schiffbau der Universitaet, Hamburg, Hamburg.

Faedo, N., Olaya, S., and Ringwood, J.V. (2017). Optimal Control, MPC and MPC-Like Algorithms for Wave Energy Systems: An Overview. *IFAC Journal of Systems and Control*.

Faedo, N., Peña-Sanchez, Y., and Ringwood, J.V. (2018). Finite-order hydrodynamic model determination for wave energy applications using moment-matching. *Ocean Engineering*, 163, 251–263.

Garcia-Violini, D. and Ringwood, J.V. (2019). Energy maximising robust control for spectral and pseudospectral methods with application to wave energy systems. *International Journal of Control*, 1–12.

Guerrero-Fernandez, J. and Gonzalez Villarreal, O.J. (2019). Model Predictive Control for Wave Energy Converters: A Moving Window Blocking Approach. Code Ocean. Available in https://doi.org/10.24433/CO.0481002.v1.

H.J.Ferreau, H.G. Bock, M.D. (2008). An online active set strategy to overcome the limitations of Explicit MPC. *International Journal of Robust and Nonlinear Control*, 18(October 2014), 816–830.

International Energy Agency (2019). Electricity Information: Overview. Technical report, International Energy Agency. URL `https://webstore.iea.org/`.

Li, G. and Belmont, M.R. (2014). Model predictive control of sea wave energy converters - Part I: A convex approach for the case of a single device. *Renewable Energy*, 69, 453–463.

Mørk, G., Barstow, S., Kabuth, A., and Pontes, M.T. (2010). Assessing the global wave energy potential. In *Proceedings of the International Conference on Offshore Mechanics and Arctic Engineering - OMAE*, volume 3, 447–454. ASME.

Pecher, A. and Kofoed, J.P. (2017). *Handbook of Ocean Wave Energy*. Springer.

Penalba, M., Cortajarena, J.A., and Ringwood, J. (2017). Validating a Wave-to-Wire Model for a Wave Energy Converter—Part II: The Electrical System. *Energies*, 10(7), 1002.

Pérez, T. and Fossen, T.I. (2008). Time-vs. frequency-domain Identification of parametric radiation force models for marine structures at zero speed. *Modeling, Identification and Control*, 29(1), 1–19.

Roessling, A. and Ringwood, J.V. (2015). Finite order approximations to radiation forces for wave energy applications. *Renewable Energies Offshore*, 359–366.

Rossiter, J. (2018). *A first course in predictive control*. CRC Press.

Sheng, W. (2019). Wave energy conversion and hydrodynamics modelling technologies: A review. *Renewable and Sustainable Energy Reviews*, 482–498.

Wang, L. (2004). Discrete model predictive controller design using Laguerre functions. *Journal of Process Control*, 14(2), 131–142.

Yu, Z. and Falnes, J. (1995). State-space modelling of a vertical cylinder in heave. *Applied Ocean Research*, 17(5), 265–275.

Towards Control of Autonomous Surface Vehicles in Rough Seas

Daniel R. McCullough * **Oscar J.G. Villarreal** *
Bryn L. Jones * **J.A. Rossiter** *

* *Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield United Kingdom.*

Abstract: This paper addresses the problem of controlling an Autonomous Surface Vehicle (ASV) in rough sea-states, with a view towards minimising wave-induced forces, whilst maintaining headway. This is a challenging control application since, and as is derived in the paper, the interaction between the vessel and the wave disturbance is nonlinear and coupled. This subsequently motivates the novel application of the Real Time Iteration Scheme (RTI) for Nonlinear Model Predictive Control (NMPC) of the ASV. Analysis of the resulting control signal provides an important insight into the role of the wave encounter frequency. Specifically, by actuating at twice the average wave encounter frequency, the nonlinear controller is able to reduce the wave forces, compared to an open-loop controller that achieves the same average velocity.

Keywords: Nonlinear and optimal marine system control, autonomous surface vehicle, nonlinear model predictive control, real time optimization.

1. INTRODUCTION

With the potential to replace manned vessels for dirty operations such as cleaning up oil spills, (Kim et al., 2012), dangerous ones, like those found in mine sweeping, or dull monotonous tasks like patrolling, (Oleynikova et al., 2010), the need for autonomous surface vehicles is increasing. This growth in use necessitates an increase in the ability of the ASV to handle more extreme ocean environments, such as rough seas, in a similar or superior manner as human pilots.

Traditional path following controllers may neglect ocean disturbances, (Lekkas and Fossen, 2014; Oh and Sun, 2010; Çimen and Banks, 2004), or consider only ocean drift forces, (Peymani and Fossen, 2013). Larger vessels, such as container ships can often assume ocean disturbances to be planar for most conditions in sea state 3 or below on the Douglas Scale. For larger vessels in the presence of waves, constraining roll is important for reducing sea-sickness and damage to cargo (Li et al., 2009, 2010).

However, smaller sea going vessels of the magnitude of tens of meters or smaller are greatly impacted by waves. Reinhart et al. (2010) use a priori optimized control path templates to find that tacking in littoral waves reduces bow diving. This behavior is used in a path planning algorithm which, when the angle between desired direction of travel and the main wave direction is smaller than a predefined threshold a secondary point is added to the path to increase the angle and to create this tacking behaviour. A PID controller is used to maintain the planned path without knowledge of the ocean environment which reduces the bow diving but does not eliminate it. With a set maximum pitch and roll constraint, Ono et al. (2014), calculates feasible safe velocity regions for use in path planning in rough seas. The model used a direct input, that is the input is the velocity of the system, allowing it to move from one safe velocity region to the next in one time step. On a boat this would not be possible, and the boat would have to move through unsafe velocity regions and potentially capsize or bow dive. Therefore, in this work we propose an optimal control strategy that is based upon a first-principles model of the ASV and wave interaction dynamics, with a view towards minimising wave induced forces whilst maintaining headway.

The rest of the paper is organized as follows; section 2 presents the derivation of a low-order state space model that describes the coupled dynamics between the ASV and a wave, section 3 introduces the control problem formulation, section 4 presents the results and discussion from the simulations, and section 5 concludes the paper, and discusses future work.

2. SYSTEM MODEL

2.1 ASV Dynamics

The ASV model is based upon a simplified description of the Halcyon ASV: more details of the 6 degrees of freedom (DOF) model can be found in (Heins et al., 2017). For the purpose of developing initial control strategies this paper examines the 1 DOF scenario, with the changes and simplifications from the full model noted below. The simplification is based upon the following assumptions.

- The model degrees of freedom are restricted to forwards (surge) motion only.
- There are no water-current or wind-induced forces.
- Actuation is restricted to the propeller input only (no steering).

- The wave induced forces arise from a single wave harmonic.
- The vessel is heading directly into the oncoming waves.

The equations of motion of the boat in the surge direction are as follows:

$$\dot{\chi}(t) = \nu(t), \tag{1a}$$

$$\dot{\nu}(t) = \frac{D(\nu(t)) + \tau_{\mathrm{p}}(\nu(t), \zeta(t)) + \tau_{\mathrm{w}}^{\nu}(\nu(t), \eta(t))}{M}, \tag{1b}$$

$$\dot{\zeta}(t) = \frac{-1}{\kappa}\zeta(t) + \frac{1}{\kappa}u(t). \tag{1c}$$

The system has five states, $x = [\chi, \nu, \zeta, \eta, \dot{\eta}]^T$, with $\chi(t)$ being the position in the boat reference frame at time t, $\nu(t)$ is the surge velocity of the boat, $\zeta(t)$ is the propeller speed, $u(t)$ is the propeller control input, and $\eta(t)$ and $\dot{\eta}$ are wave states defined in the following section. In the above equations $D(\nu(t))$ is the drag term, $\tau_{\mathrm{p}}(\nu(t), \zeta(t))$ is the propulsion from the propellers, κ is the propellor time constant, and $\tau_{\mathrm{w}}^{\nu}(\nu(t), \eta(t))$ is the wave force in the surge direction, derived in the next section. Table A.1 in the appendix list the parameters employed in this paper. The surge drag force equation is as follows:

$$D(\nu(t)) = -\frac{1}{2}\rho S C_f^*(\nu(t))\nu(t)^2, \tag{2}$$

where S is the wetted hull surface area and ρ is the density of water. The modified resistance curve, $C_f^*(\nu(t))$,is approximated by the following 6th order polynomial:

$$D(\nu(t)) = -\frac{1}{2}\rho S(p_1\nu(t)^6 + p_2\nu(t)^5 + p_3\nu(t)^4 + p_4\nu(t)^3 + p_5\nu(t)^2 + p_6\nu(t) + p_7)\nu(t)^2, \tag{3}$$

where p_x are constant coefficients defined in appendix A.2. The thrust from the dual propellers is modelled by:

$$\tau_{\mathrm{p}}(\nu(t), \zeta(t)) = 2K_\tau \rho d^4 \zeta(t)^2, \tag{4}$$

where d is the propeller diameter and where the thrust parameter K_τ is given by:

$$K_\tau(J) = K_\tau^{\{1\}}J^2 + K_\tau^{\{2\}}J + K_\tau^{\{3\}}, \tag{5}$$

where, $K_\tau^{\{i\}}$ are thrust polynomial constants defined in appendix A.3, and the advance ratio, J, is:

$$J = \frac{\nu(t)}{\zeta(t)d}. \tag{6}$$

2.2 Wave Environment and Forces

The force exerted on the boat by the wave is calculated using a Response Amplitude Operator (RAO) (Fossen, 2011). The full model uses look-up tables to find the values dependent on the conditions. In the case of the surge direction, the force RAO is appoximately an affine function of the surge velocity and wave frequency, as shown in Figure 1. The phase RAO is assumed to be constant for all boat velocities at a specific wave frequency. For the force RAO, the force is linearly dependent on the velocity, as well as linearly dependent on the wave frequency. Selecting a wave frequency, the dimensionalized force RAO can be approximated by the following equation:

$$\rho g|F^{\nu}(\nu(t))| \approx a\nu(t) + b, \tag{7}$$

where $a = 23.18$, $b = 10845$, for the wave frequency, ω, of 0.5 rad/s. Note, the force RAO magnitude, $|F^{\nu}(\nu(t))|$,

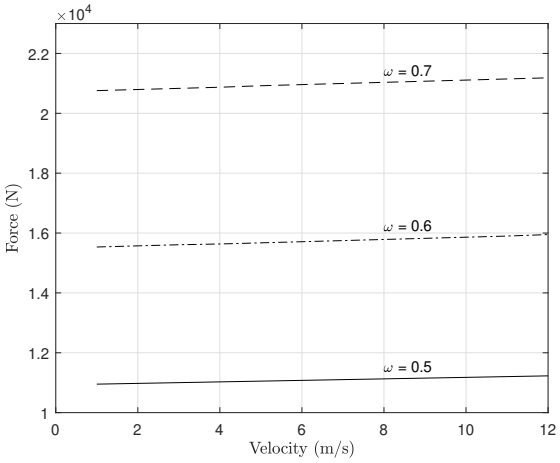


Fig. 1. Surge RAOs at various frequencies and velocities.The plot is dimensionlized with ρ and g

typically uses a subscript to indicate first order wave forces or second order drift forces. This paper only discusses first order wave forces so the subscript is excluded.

Wave Environment The force imparted on the boat from the wave is dependent upon the wave height and boat position. For a single harmonic in the surge direction the wave elevation is defined as:

$$\xi(\chi_{\mathrm{e}}, t) = a_h \cos(k\chi_{\mathrm{e}} - \omega t + \epsilon), \tag{8}$$

where a_h is the wave amplitude, ϵ is an arbitrary added phase, and $\chi_{\mathrm{e}} \in \mathbb{R}$ is the boat's position in an inertial reference frame. Assuming the boat's χ-axis coincides with the fixed reference frame χ_{e}-axis, χ_{e} can be described in the boat's body fixed frame by (Pérez and Blanke, 2002):

$$\chi_{\mathrm{e}} = \chi_0 + \int \nu(t)\, dt. \tag{9}$$

Inserting (9) into (8) and setting $\chi_0 = 0$ results in the wave elevation described in the boat's reference frame:

$$\xi(\chi, t) = a_h \cos\left(\int \nu(t)\, dt - \omega t - \epsilon\right). \tag{10}$$

With the deep water dispersion relation $k = \omega^2/g$ is assumed, the wave force term in (1b) is a function of the force RAO (7) and the wave elevation (10):

$$\tau_{\mathrm{w}}^{\nu}(\nu(t), \eta(t)) = -\rho g|F^{\nu}(\nu(t))|a_h \cos\left(\omega t + \frac{\omega^2}{g}\int \nu(t)\, dt + \phi_{\mathrm{RAO}} + \epsilon\right), \tag{11}$$

where g is the acceleration due to gravity, $|F^{\nu}(\nu(t))|$ is the force RAO, and ϕ_{RAO} is the phase RAO which is assumed constant at 1.502 radians. Note, typical notation for the wave frequency ω, wave amplitude a_h, and ϵ include a k subscript to indicate each wave component, however, to avoid confusion with the discrete time indices later, and since this paper only concerns a single wave component, the subscript has been dropped.

Next, the wave harmonic is decoupled from the height and force RAO to simplify use in state space form and is redefined as:

$$\eta(t) := \cos\left(\omega t + \frac{\omega^2}{g}\int \nu(t)\, dt + \phi_{\mathrm{RAO}} + \epsilon\right). \tag{12}$$

The dynamics of $\eta(t)$ are obtained by differentiating (12) with respect to time. The resulting expressions are somewhat involved, but can be simplified significantly by performing an order of magnitude analysis to retain only the leading-order terms under the following set of assumptions:

- $\omega \in [0.30, 0.75]$ rad/s. This is justified by observing that the vast majority of wave energy in a typical wave energy spectrum is concentrated in this band.
- $\nu(t) \in [0, 10]$ m/s. This is the typical operating range in surge velocity for the ASV studied.
- $\frac{\omega^2}{g}\dot{\nu}(t) \ll \omega + \frac{\omega^2}{g}\nu(t)$ so it is neglected.

With these assumptions the first derivative is:

$$\dot{\eta}(t) \approx -\left(\omega + \frac{\omega^2}{g}\nu(t)\right)\sin\left(\omega t + \frac{\omega^2}{g}\int \nu(t)dt + \phi_{\text{RAO}} + \epsilon\right). \tag{13}$$

The second derivative is:

$$\ddot{\eta}(t) \approx -\left(\omega + \frac{\omega^2}{g}\nu(t)\right)^2 \eta(t). \tag{14}$$

The term $\omega + \frac{\omega^2}{g}\nu(t)$ is the encounter frequency of the boat to a wave in a head sea.

2.3 Combined State Space Model

The combined surge and wave dynamics can be expressed in linear time varying form as shown in (Tomás-Rodríguez and Banks, 2010). Here, in (15), it is clear to see the coupling between the boat velocity and the wave state. The force of the wave imparted on the boat in the (2,4) term is dependent on both the velocity of the boat and the wave state, while in the (5,4) term, the square of the encounter frequency can be seen. Linearization of this system about a fixed velocity loses this coupling. This motivates the use of a nonlinear control technique.

3. CONTROLLER DESIGN

The following section presents the design of a Nonlinear Model Predictive Controller based on a condensed single-shooting approach to optimise the performance of vehicle according to a used defined cost function. The optimisation is implemented within the Real Time Iteration Scheme (RTI) (Diehl et al., 2005) which is a popular method to achieve real-time performance.

3.1 NMPC Controller

Considering a discrete-time representation of the general nonlinear system (15), the objective is to minimize a cost function of the form,

$$J = (Y_r - \hat{Y})^T Q (Y_r - \hat{Y}) + (U_r - \hat{U})^T R (U_r - \hat{U}) \tag{16a}$$
$$s.t.$$
$$x_k = x_0 \tag{16b}$$
$$x_{k+1} = f(x_k, u_k), \tag{16c}$$
$$y_k = g(x_k, u_k), \tag{16d}$$
$$U_{min} \leq \hat{U} \leq U_{max} \tag{16e}$$
$$Y_{min} \leq \hat{Y} \leq Y_{max} \tag{16f}$$

where $x_k \in \mathbb{R}^{n_x}$ are the states of the system at time k, $u_k \in \mathbb{R}^{n_u}$ are the inputs, and $y_k \in \mathbb{R}^{n_y}$ are the outputs. Moreover, $Q > 0 \in \mathbb{R}^{N_p n_y \times N_p n_y}$ and $R > 0 \in \mathbb{R}^{N_p n_u \times N_p n_u}$ are positive definite matrices for penalizing output and input errors, respectively; $Y_r, \hat{Y}, U_r, \hat{U}$ are column-vectors containing future output references, output predictions, input references and input predictions, respectively; and the optimisation subject to initial condition (16b), state dynamics (16c), state-output function (16d), and input and output constraints (16e) and (16f).

Cost function (16) is a Nonlinear Programming Problem (NLP), which is in general difficult to solve. Popular alternatives are Sequential Quadratic Programming (SQP) methods which form a linearized Convex Quadratic Program to find an optimal search direction which eventually drives the solution towards the local optimum. In predictive control, the linearization of the cost function is only defined after the future inputs and states trajectories are defined. To address this, single-shooting methods use an initially guessed nominal input trajectory $\bar{U} = [\bar{u}_k^T, \bar{u}_{k+1}^T, \cdots, \bar{u}_{k+N_p-1}^T]^T$ which can be used to obtain the nominal state and output trajectories, $\bar{X} = [\bar{x}_{k+1}^T, \bar{x}_{k+2}^T, \cdots, \bar{x}_{k+N_p}^T]^T$ and $\bar{Y} = [\bar{y}_{k+1}^Y, \bar{y}_{k+2}^T, \cdots, \bar{y}_{k+N_p}^T]^T$, respectively, by propagating the input through the state dynamics (16c) and obtaining the respective outputs through output function (16d).

By taking a first order Taylor approximation, with a slight abuse of notation, all future inputs and outputs can then be obtained starting from an initial condition mismatch δx_0 related to the Real-Time Iteration Scheme as,

$$\hat{U} = \bar{U} + \delta\hat{U} \tag{17a}$$
$$\hat{Y} = \bar{Y} + \delta\hat{Y} = \bar{Y} + G\delta x_0 + F\delta\hat{U} \tag{17b}$$

where matrices G and F are defined as

$$G = \begin{bmatrix} C_1 A_0 \\ C_2 A_1 A_0 \\ \vdots \\ C_{N_p} A_{N_p-1} \cdots A_1 A_0 \end{bmatrix}, \tag{18a}$$

$$\begin{bmatrix} \dot{\chi}(t) \\ \dot{\nu}(t) \\ \dot{\zeta}(t) \\ \dot{\eta}_k(t) \\ \ddot{\eta}_k(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & -\frac{D(\nu(t))}{M} & \frac{\tau_{\text{p}}(\nu(t),\zeta(t))}{M} & -\frac{\rho g |F^u(\nu(t))| a_h}{M} & 0 \\ 0 & 0 & -\frac{1}{\kappa} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\left(\omega + \frac{\omega^2}{g}\nu(t)\right)^2 & 0 \end{bmatrix} \begin{bmatrix} \chi(t) \\ \nu(t) \\ \zeta(t) \\ \eta_k(t) \\ \dot{\eta}_k(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{\kappa} \\ 0 \\ 0 \end{bmatrix} u(t) \tag{15}$$

$$F = \begin{bmatrix} C_1B_0 & \mathbb{O} & \cdots & \cdots \\ C_2A_1B_0 & C_2B_1 & \mathbb{O} & \cdots \\ C_3A_2A_1B_0 & C_3A_2B_1 & C_3B_2 & \cdots \\ \vdots & \vdots & \vdots & \ddots \\ C_{N_p}A_{N_p-1}\cdots A_1B_0 & C_{N_p}A_{N_p-2}\cdots A_2B_1 & \cdots & \cdots \end{bmatrix}. \tag{18b}$$

and,

$$A_k = \left.\frac{\partial f(\hat{x}_k, \hat{u}_k)}{\partial \hat{x}_k}\right|_{\substack{\hat{x}_k=\bar{x}_k \\ \hat{u}_k=\bar{u}_k}} B_k = \left.\frac{\partial f(\hat{x}_k, \hat{u}_k)}{\partial \hat{u}_k}\right|_{\substack{\hat{x}_k=\bar{x}_k \\ \hat{u}_k=\bar{u}_k}} \tag{19a}$$

Moreover, this paper focuses on regulating the average boat velocity and minimizing the wave forces defined as:

$$\tau_{\mathrm{w}} = \bar{\tau}_{\mathrm{w}} + \frac{\partial \tau_{\mathrm{w}}(\bar{\nu}_k, \bar{\eta}_k)}{\partial \nu_k}\delta\nu_k + \frac{\partial \tau_{\mathrm{w}}(\bar{\nu}_k, \bar{\eta}_k)}{\partial \eta_k}\delta\eta_k. \tag{20}$$

thus, resulting in the output matrix defined as:

$$C_i = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{\partial \tau_{\mathrm{w}}(\bar{u}_k, \bar{\eta}_k)}{\partial \nu_k} & 0 & \frac{\partial \tau_{\mathrm{w}}(u_k, \eta_k)}{\partial \eta_k} \end{bmatrix}. \tag{21}$$

The wave force in the surge direction does not vary by a large amount with a change in velocity which can be seen in Figure 1. Using an additional tuning parameter in the force RAO equation, (7) can be rewritten as:

$$|F_k^{\nu}(\nu_k)| = \alpha(a\nu_k) + b, \tag{22}$$

where α is the additional tuning weight which can be used to virtually increase the change in the wave force with respect to the boat velocity.

To obtain a desired average velocity, the rows of the linearized prediction model (17b) related to the velocity are averaged over the prediction horizon.

Optimization Substituting input and output linearised prediction models (17a) and (17b) in the original cost function (16), and rearranging the cost in terms of the decision variable $\delta\hat{U}$ (condensing approach) results in the standard QP form:

$$J = \frac{1}{2}\delta\hat{U}^T H \delta\hat{U} + \delta\hat{U}^T f + C \tag{23a}$$

s.t.

$$M\delta\hat{U} \leq \gamma \tag{23b}$$

$$H = F^T Q F + R \tag{23c}$$

$$f = -\left[F^T Q(Y_r - \bar{Y} - G\delta x_0) - R(\bar{U} - U_r)\right] \tag{23d}$$

$$M = \begin{bmatrix} I \\ -I \\ F \\ -F \end{bmatrix} \quad \gamma = \begin{bmatrix} U_{max} - \bar{U} \\ -(U_{min} - \bar{U}) \\ Y_{max} - \bar{Y} - G\delta x_0 \\ -(Y_{min} - \bar{Y} - G\delta x_0) \end{bmatrix} \tag{23e}$$

Having defined this, any QP solver of choice can be used to solve (23a). Once the optimal input deviation, $\delta\hat{U}$, is obtained, equation (17a) is used to recover the actual input. Only the first input is applied to the system, and the process is then repeated, which is the well known receding horizon strategy.

3.2 Real Time Iteration Scheme

The Real Time Iteration (RTI) scheme, is a strategy that enables real-time performance for Nonlinear Optimal Control. The following is a brief explanation of the procedure.

Initial Value Embedding (IVE) It uses the solution found in the previous step in a shifted version, typically duplicating the last input variable $u_{k+N_p|k+1} = u_{k+N_p-1|k}$, to obtain the nominal trajectory over which the formulation will linearise and optimise.

Single SQP To further reduce the computational burden and achieve predictable timings, only a single step of the SQP is performed. This is reasonable given that the solution is "hot-started" from the previous solution, which is expected to be close to the optimal solution, provided no significant unknown/unexpected disturbances have entered the system.

Computation Separation Separates the computations into preparation and feedback phases to avoid the computation delay related to the preparation of the QP. Diagrams showing the timings of these phases can be found in Gros et al. (2016).

(1) Preparation Phase: In between sampling times, the preparation phase uses a predicted nominal state for the next sampling time $\bar{x}_0 = \hat{x}_{k|k-1}$ as a starting point obtained from the last state $x_{k-1|k-1}$ and last input $u_{k-1|k-1}$ which allows the preparation of the QP main matrices $H, M, F, etc.$, and partially, vectors f and γ.
(2) Feedback Phase: Once the current state measurement becomes available the feedback phase calculates the state mismatch $\delta x_0 = x_0 - \bar{x}_0$, finishes the calculation of f and γ, and solves the QP. In some cases, it may be beneficial to run the QP prior to the state measurement assuming $\delta x_0 = 0$ to obtain an estimate of the Lagrange multipliers, λ, related to the inequalities constraints. In this strategy, the optimal solution obtained from the RTI can be shown to have the form as presented in Wang (2011):

$$\hat{U} = \bar{U} - H^{-1}\left[\underbrace{\overbrace{-(F^TQ(Y_r - \bar{Y}) - R\bar{U})}^{\text{Unconstrained}} \overbrace{+M^T\lambda}^{\text{Constrained}}}_{\text{Preparation Phase}} \underbrace{+F^TQG\delta x_0}_{\text{Feedback Phase}}\right] \tag{24}$$

4. SIMULATION RESULTS

The boat was simulated heading directly into oncoming waves with the propellers being the only actuation. The wave was a single harmonic with a wave height of 1 meter and a frequency of 0.5 rad/s. The NMPC had a prediction horizon of 200 steps ahead, with the sample period 0.08 seconds resulting in a prediction window of 16 seconds which captures just over one complete harmonic. The simulation was run for 30 seconds. The weights of the NMPC were $Q_u = 10$ for penalizing deviations of the average velocity from the desired average, $R_u = 1.4 \times 10^{-7}$ penalizing deviations from U_r, and the tuning weight in (22) is set as $\alpha = 100$. The following shows a comparison between a constant propeller input which produces an average 5 m/s velocity and the NMPC controller with a desired average velocity of 5 m/s.

Figure 2 shows the velocity profiles of the open loop controller and the NMPC controller compared to the

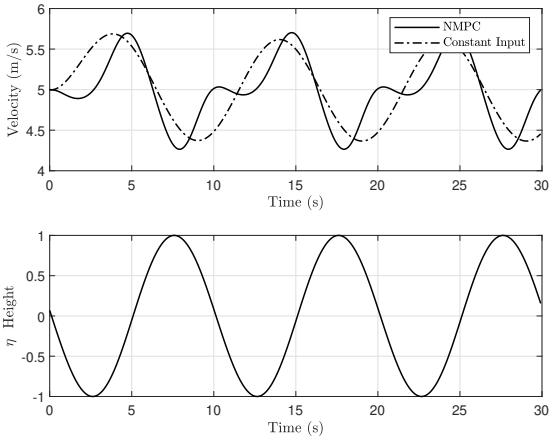


Fig. 2. The top plot shows the velocity profiles of the two controllers. The bottom plot shows the wave state η. The velocity profile shows a global minimum for the NMPC controlled boat when η is maximum and a local minimum when η is minimum.

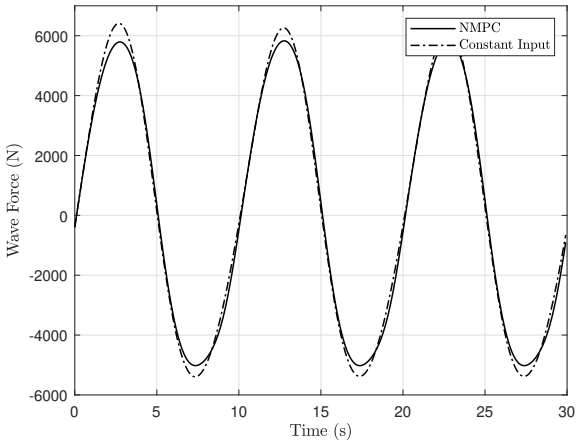


Fig. 3. Wave force comparison between the two controllers. Note: The base wave force is subtracted to more clearly show the difference in the two controllers.

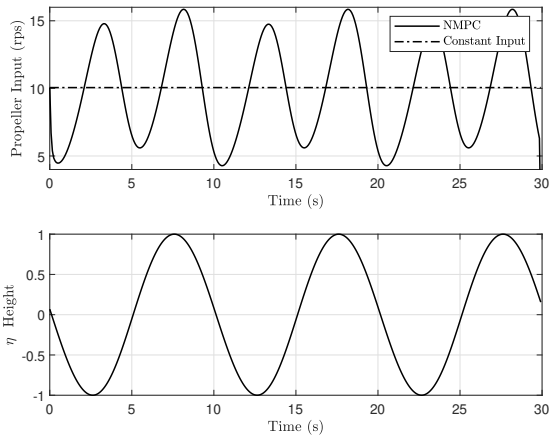


Fig. 4. The top plot shows the propeller input profiles of the two controllers. The bottom plot shows the wave state η.

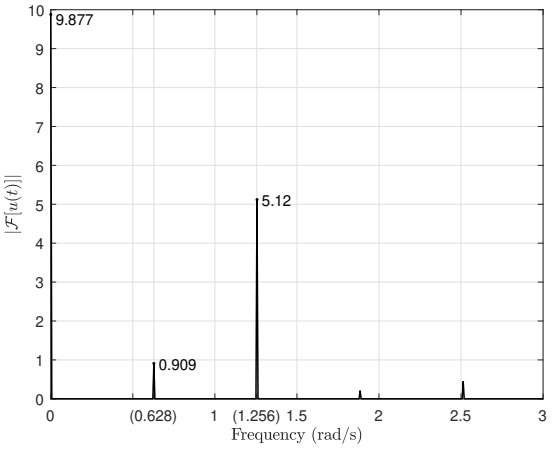


Fig. 5. An amplitude spectrum of the NMPC controller input

wave state, η. A clear difference in the velocity profiles can be seen. The open loop controller has an oscillating, single harmonic velocity resulting from changes only in the wave force, while the NMPC scenario has a more complex velocity profile. A global minimum of velocity for the NMPC controller occurs when the wave state is at its maximum, while a local minimum velocity for the NMPC controller occurs at the minimum of the wave state. Both the local and global maximum velocity occurs when the wave state is at zero. This behavior allows the boat to maintain the desired average velocity while reducing peak wave forces.

The resulting force on the boat can be seen in Figure 3. This figure shows the weighted wave force as calculated from (22). The base force, that which the boat would experience at 0 m/s, is subtracted from this figure to better show the difference in the two scenarios. Figure 4 shows the input for both controllers as it compares to the wave state. The NMPC input frequency appears to be twice that of the wave. This double harmonic is confirmed when looking at Figure 5. This figure shows the amplitude spectrum of the input signal to the propeller for NMPC. In the simulation, with an average velocity of 5 m/s the average encounter frequency of the boat is 0.628 rad/s, which has a small peak in the amplitude spectrum, while a much larger peak in seen at 1.256 rad/s or double the average encounter frequency. This can be explained by the fact that for each wave period, the minimal wave force occurs twice. NMPC exploits this by having the velocity profile shown in Figure 2, with peaks during the minimal wave force time.

5. CONCLUSION AND FUTURE WORK

This paper formulated and solved the problem of minimizing wave-induced forces upon an ASV heading into ocean waves. Because of the velocity-dependent encounter frequency, linearization of the dynamics removes the important coupling between the vessel and the wave, hence motivating the use of a NMPC which can benefit both, from the future prediction of the wave, as well as the ability

to handle nonlinear dynamics and constraints. Moreover, a key finding of this study was observed in the velocity and input profiles required to minimize wave forces which resulted in twice the average encounter frequency. Further studies will seek to use this coupling concept to explore other degrees of freedom such as pitch and roll as well as the additional input of steering, and use NMPC's ability to reduce forces and satisfy constraints to handle more complex sea states.

REFERENCES

Çimen, T. and Banks, S.P. (2004). Nonlinear optimal tracking control with application to super-tankers for autopilot design. *Automatica*, 40(11), 1845–1863.

Diehl, M., Bock, H.G., and Oder, J.P.S. (2005). A real-time iteration scheme for nonlinear optimization in optimal feedback control. 43(5), 1714–1736.

Fossen, T.I. (2011). *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons.

Gros, S., Zanon, M., Quirynen, R., and Bemporad, A. (2016). From linear to nonlinear MPC : bridging the gap via the real-time iteration. 7179. doi: 10.1080/00207179.2016.1222553.

Heins, P.H., Jones, B.L., and Taunton, D.J. (2017). Design and validation of an unmanned surface vehicle simulation model. *Applied Mathematical Modelling*, 48, 749–774.

Kim, Y.H., Lee, S.W., Yang, H.S., and Shell, D.A. (2012). Toward autonomous robotic containment booms: Visual servoing for robust inter-vehicle docking of surface vehicles. *Intelligent Service Robotics*, 5(1), 1–18. doi: 10.1007/s11370-011-0100-0.

Lekkas, A.M. and Fossen, T.I. (2014). Minimization of cross-track and along-track errors for path tracking of marine underactuated vehicles. In *2014 European Control Conference, ECC 2014*.

Li, Z., Sun, J., and Oh, S. (2009). Path following for marine surface vessels with rudder and roll constraints: An MPC approach. *Proceedings of the American Control Conference*, 3611–3616.

Li, Z., Sun, J., and Oh, S. (2010). Handling roll constraints for path following of marine surface vessels using coordinated rudder and propulsion control. *Proceedings of the 2010 American Control Conference*, 6010–6015.

Oh, S.R. and Sun, J. (2010). Path following of underactuated marine surface vessels using line of sight based model predictive control. *Ocean Engineering*, 37(2-3), 289–295.

Oleynikova, E., Lee, N.B., Barry, A.J., Holler, J., and Barrett, D. (2010). Perimeter patrol on autonomous surface vehicles using marine radar. *OCEANS'10 IEEE Sydney, OCEANSSYD 2010*, 1–5. doi: 10.1109/OCEANSSYD.2010.5603901.

Ono, M., Quadrelli, M., and Huntsberger, T.L. (2014). Safe maritime autonomous path planning in a high sea state. In *Proceedings of the American Control Conference*, 4727–4734.

Pérez, T. and Blanke, M. (2002). Simulation of Ship Motion in Seaway. 1–13.

Peymani, E. and Fossen, T.I. (2013). 2D path following for marine craft: A least-square approach. In *IFAC Proceedings Volumes (IFAC-PapersOnline)*, volume 9, 98–103. Toulouse, France.

Reinhart, R.F., Steil, J.J., Huntsberger, T.L., and Stoica, A. (2010). Tacking reduces bow-diving of high-speed Unmanned Sea Surface Vehicles. In *Proceedings - EST 2010 - 2010 International Conference on Emerging Security Technologies, ROBOSEC 2010 - Robots and Security, LAB-RS 2010 - Learning and Adaptive Behavior in Robotic Systems*, 177–182.

Tomás-Rodríguez, M. and Banks, S.P. (2010). *Linear, time-varying approximations to nonlinear dynamical systems: With applications in control and optimization*, volume 400. Springer Verlag. doi:10.1007/978-1-84996-101-1.

Wang, L. (2011). *Model Predictive Control System Design and Implementation Using MATLAB*. Springer. doi: 10.1007/978-1-4471-4399-4.

Appendix A. MODEL PARAMETERS

Table A.1. Halcyon Parameters

Name	Symbol	Value
Mass	M	11000 kg
Max Surge Velocity	$\nu^{\max}$	10 m/s
Propeller Diameter	d	0.622 m
Propeller Time Constant	κ	1.8 s
Wetted Surface Area	S	36.36 m^2
Water Density	ρ	1025 kg/m^3
Wave Amplitude	a_h	1 m
Wave Frequency	ω	0.5 rad/s
Acceleration due to Gravity	g	9.81 m/s^2
Phase RAO	ϕ_{RAO}	1.502 radians
Force RAO Slope	a	23.18 N/(m/s)
Force RAO Intercept	b	10845 N
RAO Tuning Weight	α	100

Table A.2. p Constant Coefficients

p_1	-9.22×10^{-7}	p_5	-5.52×10^{-3}
p_2	3.14×10^{-5}	p_6	2.49×10^{-3}
p_3	-4.00×10^{-4}	p_7	1.70×10^{-2}
p_4	2.31×10^{-3}		

Table A.3. $K_\tau^{\{i\}}$ Thrust Polynomial Constants

$K_\tau^{\{1\}}$	0.0041
$K_\tau^{\{2\}}$	-0.5002
$K_\tau^{\{3\}}$	0.6008

Dual Mode Stable Prediction Models for Numerically Robust Fast Nonlinear Model Predictive Control using Real Time Iterations

Oscar J. Gonzalez V., Anthony Rossiter

***Abstract*—This paper presents a novel Dual-Mode Nonlinear Model Predictive Control Scheme that uses stable prediction models for condensing based multiple-shooting frameworks that result in numerically robust optimisations. The proposed approach uses Time-Varying gains obtained from solving the Time-Varying Discrete Algebraic Ricatti Equation to stabilize the multiple-shooting trajectory, and proves the equivalency of the solution with the standard approach. Moreover, to achieve real-time performance, the approach uses the Real-Time Iteration Scheme, and an algorithm for its efficient implementation is provided. Simulations of an inverted pendulum, and its extension to the triple inverted pendulum, are presented along the paper to demonstrate the advantages and disadvantages, focusing on the numeric conditioning, the disturbance rejection properties, and the computational performance.**

***Index Terms*—Nonlinear Model Predictive Control (NMPC), Dual Mode, Real-Time Iterations (RTI), Multiple Shooting, Sequential Quadratic Programming (SQP), Nonlinear Optimization, Numerical Conditioning, Inverted Pendulum**

I. Introduction

In recent years, Nonlinear Model Predictive Control (NMPC) has gained a significant amount of attention as an advanced optimal control strategy [1], [2], [3]. Its popularity lies mainly in its ability of handling complex nonlinear dynamics and constraints. A key challenge for its implementation is the development of efficient solutions that allow fast/real-time performance [1], [2], [3]. One of the most successful approaches to tackle this is the Real-Time Iteration (RTI) Scheme, originally proposed in [4], which exploits the fact that NMPC is required to successively solve Optimal Control Problems (OCP) which are closely linked to each other [3]. Moreover, the efficiency of the resulting approach depends largely on how the algorithms are programmed, as well as the platforms in which they are deployed, e.g. using Field-Programmable Gate-Arrays (FPGA) [2]. To address this, several toolkits exist such as the ACADO toolkit [1], VIATOC and CasADi [5], to name a few, offering efficient autogeneration routines aimed at giving extremely fast performance and releasing the burden of programming NMPC routines manually. Furthermore, the underlying optimisations can be solved using simultaneous or sequential approaches leading to sparse or condensed OCPs [6], [7]. Work by [7] concluded that condensing based approaches, where states are eliminated from the decision variables, are faster for small to medium OCPs, whereas simultaneous/sparse approaches, where the states are kept as decision variables, give better overall performance for large-scale optimisations, and can deal successfully with unstable systems [6]. Finally, the optimisation can be done using a variety of methods such as collocation points [1], [8] and single/multiple shooting [2], [8].

On the other hand, the quality of the solutions is subject to the numeric accuracy used, and more importantly, the numeric conditioning of the formulated optimisation, particularly for condensing based approaches [9]. To address this issue, closed-loop dual-mode prediction models have been used extensively, although mostly for linear MPC, with fewer works found for NMPC. It is noted that dual-modes can be applied based on open-loop or closed-loop paradigm as discussed in [10], where the former, switches between two controllers depending on whether the state is inside the terminal region [11], [12], [13], and the latter imposes a stabilizing gain across all the prediction horizon and uses additional deviation variables for constraint handling [10], [14], [9], [15]. Dual-mode for NMPC based on the open-loop paradigm was originally proposed in [16] which guarantees stability for stable systems by using a single stabilizing gain K based on the Linear Quadratic Regulator (LQR) solution that stabilizes the state to the origin in the terminal region. Other works such as [17], [18], [12] also use this idea. Work by [19] used a PI controller instead for the terminal region. Work by [11] proposed an adaptive quasi-infinite NMPC which updated the LQR gain and terminal weights online based on the current steady-state target/reference and the model parameters obtained by the adaptation. Other approaches such as [20], [1], initialize the nonlinear optimisation with the LQR trajectory, and improve it from there. It is important to note that all of this approaches are better suited for stable systems as it is known that unstable systems are better handled by the closed-loop paradigm [10], [14] or, as mentioned earlier, by simultaneous approaches [6]. Dual-mode based on the closed-loop paradigm was originally proposed for state-space linear MPC in [9]. Similarly, work from [15] have used the closed-loop paradigm for NMPC with a single locally stabilizing gain K across the entire prediction to stabilize the system around a given steady state target/reference. Finally, work by [21], [22] used a time-varying controller calculated offline, to stabilize the system around a pre-defined periodic trajectory, which arguably could be refered to as "linear" MPC, as discussed in [3].

O. Gonzalez was with The University of Sheffield, Sheffield, UK e-mail: ojgonzalezvillarreal1@sheffield.ac.uk

A. Rossiter is with The University of Sheffield, Sheffield, UK e-mail: j.a.rossiter@sheffield.ac.uk

Manuscript received February 7, 2020; revised February 7, 2020.

A key issue that has not yet been addressed for condensing based NMPC is: *how can the system be stabilized around any trajectory that emerges from the nonlinear optimisation?*. Indeed, it is possible that the trajectory presents highly unstable dynamics before even getting close to the steady state target/reference (as it is the case of the triple pendulum), and non of the currently available toolkits offers a generic methodology or option for prestabilizing the system to allow condensing approaches to be used for unstable systems. Moreover, the importance of numeric conditioning of the optimisation for unstable systems, and consequently, how this affects or not the solution, is commonly overlooked or simply ignored. Finally, the ability to use the reduced numeric accuracy to obtain faster solutions is not commonly exploited.

This paper aims to address the aforementioned issues and proposes a generalisable method to tackle condensing based multiple-shooting NMPC frameworks for unstable systems. The proposed methodology uses the dual-mode approach based on the closed loop paradigm to obtain stable prediction models that improve the numerical properties of the optimisation. The approach uses time-varying gains K_k obtained from solving the Time-Varying Discrete Algebraic Ricatti Equation (DARE) to stabilize the multiple-shooting trajectory (as opposed to common approaches where a single linear terminal control law aims at stabilizing the state to the origin), and prove its equivalency with the standard multiple-shooting solution. Moreover, to achieve real-time performance, the method is combined with the RTI Scheme, and a general algorithm for its implementation is provided.

The paper is organized as follows: Section II starts by defining the general models and OCPs of interest. Sections II-A and II-B present a detailed derivation of the proposed approach including the definition of the dual-mode stable prediction models based on the multiple-shooting approach, and discuss the general form of the resulting optimal solution. Section II-C presents the key contribution of this paper, theorem 1; which establishes the equivalency between our proposed approach and the standard solution, resulting in the same stability and convergence properties, with several advantages and disadvantages discussed further in section II-D. Details related to the RTI Scheme are introduced in section II-E, and an algorithm split in two parts (preparation and feedback phases of RTI) is given in section II-F summarizing the whole methodology. Section III presents a first example of its application with simulations of an inverted pendulum focusing on numeric conditioning, disturbance rejection and computational performance. To provide a more complete example, section IV presents a simulation of a swing up and stabilization of the triple inverted pendulum system where the standard condensing based NMPC failed to solve the optimisation altogether, and discusses the numeric conditioning and disturbance rejection properties. Finally, section V emits conclusions, summarizes the contribution of this paper and presents future work.

For the interest of the reader, the Matlab and C++ files used to generate the results discussed in this paper can be found in [23].

II. Nonlinear Model Predictive Control

This paper focuses on discrete-time models of the form,

$$x_{k+1|k} = f(x_{k|k}, u_{k|k}) \tag{1}$$

where $x_k \in \mathbb{R}^{n_x}$, $u_k \in \mathbb{R}^{n_u}$ are states and inputs column-vectors, respectively. The notation "$k+1|k$" reads, "predicted value at time-step $k+1$ calculated at time step k", and will only be used for clarity when needed.

Remark 1. *If the system is in continuous-time, direct transcription methods can be used to obtain a discrete model, typically using integration methods such as Forward Euler or Explicit Runge-Kutta as in [3].*

We now seek to optimise the predicted performance of system (1) along a given prediction horizon N_p by minimizing cost function (2) defined as;

$$J = (X_r - \hat{X})^T Q (X_r - \hat{X}) + (U_r - \hat{U})^T R (U_r - \hat{U}) \tag{2a}$$
$$s.t. \quad x_k = x_0 \tag{2b}$$
$$\hat{x}_{k+i} = f(\hat{x}_{k+i-1}, \hat{u}_{k+i-1}) \quad \forall i = [1, N_p] \tag{2c}$$
$$U_{min} \leq \hat{U} \leq U_{max} \tag{2d}$$
$$X_{min} \leq \hat{X} \leq X_{max} \tag{2e}$$

where $Q > 0 \in \mathbb{R}^{N_p n_x \times N_p n_x}$ and $R > 0 \in \mathbb{R}^{N_p n_u \times N_p n_u}$ are positive definite matrices for penalizing state and input errors, respectively, typically selected as $Q = blkdiag([q_{k+1}, q_{k+2}, \dots, q_{k+N_p}])$ where q_{k+N_p} is typically referred to as the terminal weight, and $R = r_u I^{N_p n_u \times N_p n_u}$; $X_r = [x_{r_{k+1}}^T, x_{r_{k+2}}^T, \cdots, x_{r_{k+N_p}}^T]^T \in \mathbb{R}^{N_p n_x}$, $\hat{X} = [\hat{x}_{k+1}^T, \hat{x}_{k+2}^T, \cdots, \hat{x}_{k+N_p}^T]^T \in \mathbb{R}^{N_p n_x}$, $U_r = [u_{r_k}^T, u_{r_{k+1}}^T, \cdots, u_{r_{k+N_p-1}}]^T \in \mathbb{R}^{N_p n_u}$, $\hat{U} = [\hat{u}_k^T, \hat{u}_{k+1}^T, \cdots, \hat{u}_{k+N_p-1}]^T \in \mathbb{R}^{N_p n_u}$ are future state references, states, inputs references and input column-vectors, respectively; (2b) is the initial condition; (2c) are the state dynamics; and (2d) and (2e) are the inputs and state constraints, with $X_{max}/X_{min} \in \mathbb{R}^{N_p n_x}$ and $U_{max}/U_{min} \in \mathbb{R}^{N_p n_u}$.

Remark 2. *If appropriate, terminal constraints for the state $x_{min} \leq \hat{x}_{k+N_p} \leq x_{max}$ can be imposed for stability [2] by selecting appropriate vectors for X_{min}, X_{max}.*

To solve this optimisation we now look to apply Sequential Quadratic Programming (SQP) methods where the cost is linearized at a given trajectory, resulting in a linearized Convex Quadratic Program (QP) which can be used to find an optimal search direction, typically based in the Newton-method, that eventually converges to the local-optimal. Notice the linearisation of the trajectory is only defined after a given input/state pairs have been applied through dynamics (2c). A popular alternative are shooting methods which use an "initially guessed" **nominal input trajectory** $\bar{U} = [\bar{u}_k^T, \bar{u}_{k+1}^T, \cdots, \bar{u}_{k+N_p-1}^T]^T \in \mathbb{R}^{N_p n_u}$ and **nominal state trajectory** $\bar{X} = [\bar{x}_{k+1}^T, \bar{x}_{k+2}^T, \cdots, \bar{x}_{k+N_p}^T]^T \in \mathbb{R}^{N_p n_x}$ to linearize the OCP along the trajectory.

A. Dual Mode Stable Prediction Models

The standard multiple-shooting NMPC approach linearises the system along this trajectories using first order Taylor

Series for the state and the inputs, and imposes an additional continuity term (d_k of equation (3b)) for the propagation of the state. However, in this paper we look to address the issue that arises when the system presents unstable dynamics, and therefore present unstable predictions w.r.t. to the decision variables.

Dual-mode prediction models based on the closed loop paradigm [10] offer a viable solution to cancel the unstable dynamics of the system as originally developed and discussed in [9] for linear state-space GPC. However, as opposed to the linear case where a single linear gain K, typically obtained from LQR, can be used to pre-stabilize the system to the origin, a non-linear system may require time-varying, possibly nonlinear gains. Moreover, giving it may be difficult to find a generic stabilizing gain (linear or nonlinear) that satisfies constraints and stabilizes any system to the origin, this approach aims at using time-varying gains that aim at stabilizing the current guess of the optimal trajectory $(\bar{X}, \bar{U})$ instead.

To achieve this, the linearisation of the model is then given by,

$$\hat{x}_{k+1} - \bar{x}_{k+1} = \delta\hat{x}_{k+1} = A_k\delta\hat{x}_k + B_k\delta\hat{u}_k + \bar{d}_{k+1} \tag{3a}$$
$$\bar{d}_{k+1} = f(\bar{x}_k, \bar{u}_k) - \bar{x}_{k+1} \tag{3b}$$
$$\hat{u}_k - \bar{u}_k = \delta\hat{u}_k = -K_k\delta\hat{x}_k + \delta\hat{c}_k \tag{3c}$$

where,

$$A_k = \left.\frac{\partial f(\hat{x}_k, \hat{u}_k)}{\partial \hat{x}_k}\right|_{\substack{\hat{x}_k=\bar{x}_k \\ \hat{u}_k=\bar{u}_k}} \quad B_k = \left.\frac{\partial f(\hat{x}_k, \hat{u}_k)}{\partial \hat{u}_k}\right|_{\substack{\hat{x}_k=\bar{x}_k \\ \hat{u}_k=\bar{u}_k}} \tag{4}$$

and K_k is a stabilizing gain obtained from solving the Time-Varying Discrete Algebraic Ricatti Equations (DARE) (5) backwards in time along the nominal state/input trajectories, starting from $(\bar{x}_{k+N_p-1}, \bar{u}_{k+N_p-1}, P_{N_p} = q_{k+N_p})$ using q_{k+i} and r_u weights defined previously as in [24];

$$P_k = q_k + A_k^T P_{k+1} A_k + (B_k^T P_{k+1} A_k)^T K_k^T \tag{5a}$$
$$K_k^T = (r_u + B_k^T P_{k+1} B_k)^{-1} B_k^T P_{k+1} A_k \tag{5b}$$

Remark 3. *Note that this pre-stabilization scheme emerges from a secondary "inner" optimisation which has a different cost/objective than the original cost (2), ie. the objective of stabilizing the multiple shooting trajectory itself. However, it will be proved in theorem (1) that after combining both optimisation methods, the solution to the original problem is exactly same.*

By substituting $\delta\hat{u}_k = -K_k\delta\hat{x}_k + \delta\hat{c}_k$ from (3c) in (3a), a stable linearisation model can be obtained as,

$$\Phi_k = A_k - B_k K_k \tag{6a}$$
$$\delta\hat{x}_{k+1} = \Phi_k\delta\hat{x}_k + B_k\delta\hat{c}_k + \bar{d}_{k+1} \tag{6b}$$

After propagating model (6b) N_p steps forward starting from an initial state mismatch δx_0, all future inputs and state, $\hat{U}$ and $\hat{X}$, are condensely represented by;

$$\hat{X} = \bar{X} + \delta\hat{X} = \bar{X} + D + G\delta x_0 + H\delta\hat{C} \tag{7a}$$
$$\hat{U} = \bar{U} + \delta\hat{U} = \bar{U} + S + W\delta x_0 + F\delta\hat{C} \tag{7b}$$

where $\delta x_0 = x_0 - \bar{x}_0$ is an initial condition mismatch which forms part of the RTI Scheme, $\delta\hat{C}$ are now the inputs of the system, and,

$$D = \begin{bmatrix} \tilde{d}_1 \\ \tilde{d}_2 \\ \vdots \\ \tilde{d}_{N_p} \end{bmatrix} G = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_{N_p} \end{bmatrix} H = \begin{bmatrix} h_{1,1} & 0 & \cdots & 0 \\ h_{2,1} & h_{2,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ h_{N_p,1} & h_{N_p,2} & \cdots & h_{N_p,N_p} \end{bmatrix} \tag{8a}$$

$$S = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_{N_p} \end{bmatrix} W = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{N_p} \end{bmatrix} F = \begin{bmatrix} f_{1,1} & 0 & \cdots & 0 \\ f_{2,1} & f_{2,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ f_{N_p,1} & f_{N_p,2} & \cdots & f_{N_p,N_p} \end{bmatrix} \tag{8b}$$

where $D \in \mathbb{R}^{N_p n_x}$, $G \in \mathbb{R}^{N_p n_x \times n_x}$, $H \in \mathbb{R}^{N_p n_x \times N_p n_u}$, $S \in \mathbb{R}^{N_p n_u}$, $W \in \mathbb{R}^{N_p n_u \times n_x}$, $F \in \mathbb{R}^{N_p n_u \times N_p n_u}$, and with a slight abuse of notation by dropping the $k+i \to i$ indexes, the inner matrices are defined through the following recursions as,

$$\tilde{d}_i = \begin{cases} \bar{d}_i & i = 1 \\ \bar{d}_i + \Phi_{i-1}\tilde{d}_{i-1} & i > 1 \end{cases} \tag{9a}$$
$$g_i = \begin{cases} \Phi_{i-1} & i = 1 \\ \Phi_{i-1} g_{i-1} & i > 1 \end{cases} \tag{9b}$$
$$h_{i,j} = \begin{cases} B_{j-1} & i = j \\ \Phi_{i-1} h_{i-1,j} & i > j \end{cases} \tag{9c}$$
$$s_i = \begin{cases} \mathbb{O}^{n_u} & i = 1 \\ -K_{i-1}\tilde{d}_{i-1} & i > 1 \end{cases} \tag{9d}$$
$$w_i = \begin{cases} -K_{i-1} & i = 1 \\ -K_{i-1} g_{i-1} & i > 1 \end{cases} \tag{9e}$$
$$f_{i,j} = \begin{cases} I^{n_u \times n_u} & i = j \\ -K_{i-1} h_{i-1,j} & i > j \end{cases} \tag{9f}$$

We now look to use the condensing approach where, by substituting the stable linearized prediction models (7) in (2) and rearranging in terms of the decision variable $\delta\hat{C}$ results in the standard QP format (10a).

$$J = \frac{1}{2}\delta\hat{C}^T E\delta\hat{C} + \delta\hat{C}^T f + const \quad s.t. \tag{10a}$$
$$E = H^T Q H + F^T R F \tag{10b}$$
$$f = -\left[H^T Q(X_r - \bar{X} - D - G\delta x_0) - F^T R(\bar{U} + S + W\delta x_0 - U_r)\right] \tag{10c}$$
$$M\delta\hat{C} \leq \gamma \tag{10d}$$
$$M = \begin{bmatrix} F \\ -F \\ H \\ -H \end{bmatrix} \quad \gamma = \begin{bmatrix} U_{max} - \bar{U} - S - W\delta x_0 \\ -(U_{min} - \bar{U} - S - W\delta x_0) \\ X_{max} - \bar{X} - D - G\delta x_0 \\ -(X_{min} - \bar{X} - D - G\delta x_0) \end{bmatrix} \tag{10e}$$

with E known as the Hessian, f typically referred as the linear term, and M and γ are the constraint matrix and vector, respectively. In some cases, not all the states may be required to be constrained which can be done by selecting/computing only the appropriate rows of M.

B. The Optimal Solutions

By derivating (10a) w.r.t. the decision variable $\delta\hat{C}$ and equating to zero ($\frac{\partial J}{\partial \delta \hat{C}} = 0$), the well known unconstrained solution can be found to be,

$$\delta\hat{C}_{unc} = -E^{-1}f \tag{11}$$

For constrained solutions, any QP solver such as QPOases or Matlab function "quadprog" can be used to compute the optimal solution after having defined (E, f, M, γ), which is known to have a form of $\delta\hat{C}_{opt} = \delta\hat{C}_{unc} + \delta_\lambda\hat{C}$, ie. the unconstrained solution plus a deviation due to constraints (λ - Lagrange Multipliers) [25].

After solving the optimisation, an expansion step is applied by using the linearized models (7a) and (7b) to obtain both, the nominal state trajectory $\bar{X}^{[i+1]} = \hat{X}^{[i]}$ and the nominal input trajectory $\bar{U}^{[i+1]} = \hat{U}^{[i]}$, for the next iterations over which the SQP will re-linearize and optimize the QP. Only the first input is applied to the system and the process is repeated which is the well known "receding horizon" strategy.

C. The Equality of the Solutions

Theorem 1. *The Equality of the Solutions*

The solution with the proposed dual mode stable prediction models is exactly the same as the standard solution.

Proof. The standard solution, ie. the one that uses the predictions with $K_k = \mathbb{O}$, results in $F = I$, $S = W = \mathbb{O}$, and therefore $\delta\hat{C} = \delta\hat{U}$ which results in the following prediction matrices;

$$\hat{X} = \bar{X} + D_1 + G_1\delta x_0 + H_1\delta\hat{U} \tag{12a}$$
$$\hat{U} = \bar{U} + \delta\hat{U} \tag{12b}$$

with an unconstrained solution of the form,

$$\delta\hat{U} = \left(H_1^T Q H_1 + R\right)^{-1}\left(H_1^T Q(X_r - \bar{X} - D_1 - G_1\delta x_0) - R(\bar{U} - U_r)\right) = -E_1^{-1}f_1 \tag{13}$$

In contrast, our approach uses prediction models,

$$\hat{X} = \bar{X} + D_2 + G_2\delta x_0 + H_2\delta\hat{C} \tag{14a}$$
$$\hat{U} = \bar{U} + S + W\delta x_0 + F\delta\hat{C} \tag{14b}$$

and has an unconstrained solution of the form,

$$\delta\hat{U} = S + W\delta x_0 + F\left(H_2^T Q H_2 + F^T R F\right)^{-1}\left[H_2^T Q(X_r - \bar{X} - D_2 - G_2\delta x_0) - F^T R(\bar{U} + S + W\delta x_0 - U_r)\right] \tag{15}$$

Notice the D_1/D_2 – G_1/G_2 – H_1/H_2 notation has been used to distinguish the two state prediction models. However, because both models produce exactly the same predictions for a given δU, ie. $\hat{X} = \bar{X} + D_1 + G_1\delta x_0 + H_1\delta\hat{U} = \bar{X} + D_2 + G_2\delta x_0 + H_2\delta\hat{C}$ and $\hat{U} = \bar{U} + \delta\hat{U} = \bar{U} + S + W\delta x_0 + F\delta\hat{C}$, then the following hold;

$$D_2 = D_1 + H_1 S \tag{16a}$$
$$G_2 = G_1 + H_1 W\delta x_0 \tag{16b}$$
$$H_2 = H_1 F \tag{16c}$$

Substituting equation (16c) in (15) and rearranging it in terms of the Hessian E_1 of the standard solution (13) gives;

$$\delta\hat{U} = S + W\delta x_0 + F\left(F^T E_1 F\right)^{-1} F^T\left[H_1^T Q(X_r - \bar{X} - D_2 - G_2\delta x_0) - R(\bar{U} + S + W\delta x_0 - U_r)\right] \tag{17}$$

Given F is always invertible because of the identity matrix in the diagonal, and E_1 is always invertible because is positive definite, the terms related to the inverse of the inner product are given by;

$$F\left(F^T E_1 F\right)^{-1} F^T = F(F^{-1}E_1^{-1}F^{T^{-1}})F^T = E_1^{-1} \tag{18}$$

Substituting equations (16b), (16a) and (18) in (17) gives,

$$\delta\hat{U} = S + W\delta x_0 + E_1^{-1}\left[H_1^T Q(X_r - \bar{X} - D_1 - H_1 S - G_1\delta x_0 - H_1 W\delta x_0)\right) - R(\bar{U} + S + W\delta x_0 - U_r)\right] \tag{19}$$

Rearranging terms,

$$\begin{aligned}\delta\hat{U} &= S + W\delta x_0 - E_1^{-1}E_1(S + W\delta x_0) + E_1^{-1}\left[H_1^T Q(X_r - \bar{X} - D_1 - G_1\delta x_0) - R(\bar{U} - U_r)\right] \\ &= E_1^{-1}\left[H_1^T Q(X_r - \bar{X} - D_1 - G_1\delta x_0) - R(\bar{U} - U_r)\right] \\ &= -E_1^{-1}f_1\end{aligned} \tag{20}$$

Thus, the equality of the solutions (II-C) and (13) holds.

A similar but slightly longer proof holds for constrained solutions, given the same constraints are imposed which in turn would lead to the same constraint corrections to the unconstrained solution;

$$\hat{X} = \bar{X} + \delta\hat{X} + \delta_\lambda\hat{X} \tag{21a}$$
$$\hat{U} = \bar{U} + \delta\hat{U} + \delta_\lambda\hat{U} \tag{21b}$$

Moreover, notice this proof also holds for single-shooting scenarios where the system is linearized along the state trajectory $\bar{X}$ obtained with $\bar{U}$ resulting in $d_k = \mathbb{O}^{n_x}\forall k = [1, N_p]$, and consequently in $S = D = \mathbb{O}^{N_p n_x}$. □

D. Stability, Convergence and Numerical Robustness

Because the solution given by our proposed approach is exactly the same as the one given by the standard single/multiple-shooting solution as proved by theorem (1), the convergence and stability properties are exactly the same. However, the benefit of the proposed approach is that the prediction matrix H is now stable w.r.t. the decision variable $\delta\hat{C}$ which leads to a numerically robust Hessian inversion required by the optimisation. This allows longer prediction horizons for unstable systems without sacrificing numerical robustness of the solution, as well as possibly the use of less accurate inverse solutions and weaker numeric precision representations such as floats for computing purposes. Although the numeric advantages are particularly present when using condensed/sequential solutions, the methodology can also be used for simultaneous approaches.

The aforementioned benefits were particularly observed for the inverted pendulum systems used in the benchmark sections III and IV with significant condition numbering differences of the Hessian when the approach is not used, and in some cases, giving singular Hessian or failing the solve the optimisation altogether (as it is the case of the triple pendulum). The main disadvantage of the proposed approach is that it requires slightly longer computation times, particularly due to the computations related to the solution of Time-Varying DARE backwards in time to obtain K_k; the computations Φ_k, S, W and F; and the fully dense constraint matrix which prevents the use of the special case available in standard QPs where the constraints in the inputs are imposed through an identity matrix I rather than dense matrix F. Moreover, if an interior point method is used to solve the optimization, assuming the system was linearised over a feasible input trajectory $\bar{U}$, an initial feasible point for $\hat{U}$ is not necessarily achieved by $\delta\hat{C} = \mathbb{O}^{N_p n_u}$, as opposed to directly imposing $\delta\hat{U} = \mathbb{O}^{N_p n_u}$ in the standard method which would be feasible. An alternative is to initialize $\delta\hat{C} = F^{-1}(\mathbb{O}^{N_p n_u} - S - W\delta x_0)$, which is the required $\delta\hat{C}$ to obtain $\delta\hat{U} = \mathbb{O}^{N_p n_u}$ that is easily present in the standard solution.

E. Real Time Iterations

To achieve real-time performance of the optimisation, the Real-Time Iteration Scheme originally developed in [4] was used. The latter is briefly summarized in this section, and for more details, the reader is refered to [3] which gives an excellent tutorial like paper of this method.

The scheme consists of 3 strategies for the multiple-shooting approach:

1) Initial Value Embedding:
   It uses a shifted version of the solution for the nominal state and input trajectories obtained in the previous time step to hot-start the trajectories over which the SQP will linearise, typically duplicating the last input $\bar{u}_{k+N_p-1|k} = \hat{u}_{k+N_p-2|k-1}$, and shifting the state $\bar{X}_{k|k-1} = [\hat{x}^T_{k+1|k-1}, \dots, \hat{x}^T_{k+N_p-1|k-1}, \hat{x}^T_{k+N_p|k}]^T$, where $\hat{x}_{k+N_p|k} = f(\bar{x}_{k+Np-1|k-1}, \bar{u}_{k+N_p-2|k-1})$.
2) Single SQP Iteration:
   It performs only a single SQP iteration given the hot-started trajectory is expected to be close, provided no significant disturbances have entered the system. Assuming the latter and other conditions discussed in [3] are satisfied, the scheme can guarantee local asymptotic closed-loop stability.
3) Computation Separation:
   It separates the computations required for the optimisation into preparation and feedback phases to avoid the computation delays required by the optimisation:
   a) Preparation Phase: In between sampling times $k-1 \to k$, it uses the predicted state for the next sampling time $\bar{x}_0 = \hat{x}_{k|k-1}$ as an initial condition of (2) which enables the computation of all the matrices required by the optimisation (D, G, H, S, W, F, E, M), and partially the calculation of (f and γ) given the dependancy on δx_0.
   b) Feedback Phase: As soon as the state is available, it calculates $\delta x_0 = x_0 - \bar{x}_0$, completes the calculation of f and γ, and solves the QP.

F. Algorithm

To summarize the overall methodology, this section provides a generic algorithm for the overall implementation of the approach. The algorithm is divided into the preparation and feedback phases of the RTI Scheme, namely algorithms (1) and (2). Although it may seem slightly different calculations are used, note that the terms D and S are implicit in the update of $\bar{X}$ and $\bar{U}$ in lines 31 and 29, respectively, of algorithm (2); and similarly, the terms related to $G\delta x_0$ and $W\delta x_0$ are included in lines 3 and 4, respectively, of algorithm (1), all of which are used for the calculation of f and γ.

Algorithm 1: Feedback Phase

Data: $x_0, \bar{X}, \bar{U}, X_r, U_r, E, M, G, H, W, F, Q, R$

1 ————————- Update ————————-
2 Calculate $\delta x_0 = x_0 - \bar{x}_0$;
3 Update $\bar{X} = \bar{X} + G\delta x_0$;
4 Update $\bar{U} = \bar{U} + W\delta x_0$;
5 Calculate $f = -[H^T Q(X_r - \bar{X}) - F^T R(\bar{U} - U_r)]$;
6 Calculate $\gamma = \begin{bmatrix} U_{max} - \bar{U} \\ \bar{U} - U_{min} \\ X_{max} - \bar{X} \\ \bar{X} - X_{min} \end{bmatrix}$;
7 ————————- Optimize ————————-
8 $\delta\hat{C}$=QPSolver(E, f, M, γ);
9 ————————- Expansion Step ————————-
10 $\bar{X} = \bar{X} + H\delta\hat{C}$;
11 $\bar{U} = \bar{U} + F\delta\hat{C}$;
12 ————————- Result ————————-

Result: $u_k = \bar{U}(1), \bar{X}, \bar{U}$

III. Example 1: The Inverted Pendulum

To evaluate the performance of the proposed methodology, the inverted pendulum was used as a first example commonly used as a benchmark given it presents challenging underactuated, unstable and non-minimum phases nonlinear dynamics in the upward equilibrium subject to input and state constraints [26], [24], [27]. It is worth mentioning that although the numeric conditioning problem is naturally present in higher order systems (eg. 10-100 states), it can also be present in low-order systems such as the system at hand, and given condensing based approaches are naturally better suited for small to medium sized optimisations, this system was selected as a first example. Nonetheless, a higher order system such as the triple inverted pendulum (8 states) will be considered in section IV.

In this paper, a simplified model of the inverted pendulum available in [28] was used which is given by.

$$\ddot{p} = f_m \dot{p} + ku \tag{22a}$$

$$\ddot{\theta} = a\dot{\theta} + b\sin(\theta) + c\cos(\theta)(f_m v_k + ku_k) \tag{22b}$$

Algorithm 2: Preparation Phase

Data: $\bar{X}, \bar{U}, Q, R, N_p$
Obtain $\bar{x}_0$ from $\bar{X}$ obtained in the previous time ($\bar{x}_0 = \bar{x}_{k|k-1}$);
Shift $\bar{U}$ and $\bar{X}$;
———— Forward ————
for $k = 0$ **to** N_p **do**
  Store A_k, B_k, d_k
end
———— DARE Backwards ————
Initialize $P_{N_p} = q_{k+N_p}$;
for $k = N_p - 1$ **to** 0 **do**
  Compute $P_k = q_k + A_k^T P_{k+1} A_k + (B_k^T P_{k+1} A_k)^T K_k^T$;
  Store $K_k^T = (r + B_k^T P_{k+1} B_k)^{-1} B_k^T P_{k+1} A_k$;
  Store $\Phi_k = A_k - B_k K_k$
end
———— Main Matrices ————
Initialize $F = I$;
for $k = 1$ **to** N_p **do**
  if $k == 1$ **then**
    Store $\tilde{d}_k = d_k$;
    Store $g_k = A_{k-1}$;
    Store $h_{k,1} = B_{k-1}$;
    Store $w_k = -K_{k-1}$;
  else
    Store $\tilde{d}_k = d_k + A_{k-1}\tilde{d}_{k-1}$;
    Store $g_k = A_{k-1} g_{k-1}$;
    Store $h_{k,1\to k} = [A_{k-1} h_{k,1\to k-1}, B_{k-1}]$;
    Store $s_k = -K_{k-1} w_{k-1}$;
    Store $w_k = -K_{k-1} g_{k-1}$;
    Store $f_{k,1\to k-1} = -K_{k-1} h_{k,1\to k-1}$;
    Store $\bar{u}_{k-1} = \bar{u}_{k-1} + s_k$
  end
  Store $\bar{x}_k = \bar{x}_k + \tilde{d}_k$;
end
Form $M = [F^T, -F^T, H^T, -H^T]^T$;
Calculate $E = H^T Q H + F^T R F$;
———— Result ————
Result: $\bar{X}, \bar{U}, G, H, W, F, E, M$

Considering the state $x_k = [v, \omega, p, \theta]^T$ with $v = \dot{p}$ and $\omega = \dot{\theta}$ and using a forward-euler integration method the simplified model is given by;

$$x_{k+1} = x_k + T_s f(x_k, u_k) \tag{23a}$$

$$f(x_k, u_k) = \begin{bmatrix} f_m v_k + k u_k \\ a\omega_k + b\sin(\theta_k) + c\cos(\theta_k)(f_m v_k + k u_k) \\ v_k \\ \omega_k \end{bmatrix} \tag{23b}$$

where $T_s = 0.02\ (s)$ is the sampling time; p is the position; v is the velocity; θ is the pendulums' angle; ω is the pendulums' angular velocity; and u_k is the input of the system. Furthermore, the coefficients were defined as $f_m = -4.67$; $k = 0.065$; $a = -0.129$; $b = 38.4$; and $c = 3.95$. Finally, constraints in the input and position were imposed as $-170 < u < 170$ and $-0.35 \leq p \leq 0.35$, respectively.

A. Numerical Performance Evaluation

To evaluate the performance of our methodology, the optimisation was done for different prediction horizons using weights $q_{k+i} = diag([0.1, 0.1, 10, 10])\ \forall i = [1, N_p - 1]$, $r_u = 0.001$ to penalize the state and input errors. A terminal weight of $q_{k+N_p} = 10 q_{k+i}$ was imposed in the last state of the horizon x_{k+N_p} to improve stability properties of the optimisation. All the simulations started at the lower equilibrium in steady state $x_r = x_0 = [0, 0, 0, \pi]^T$, and a reference change to the upward equilibrium ($x_r = [0, 0, 0, 0]$) was given by introducing it at the end of the prediction horizon to achieve better performance of the RTI Scheme as discussed in [3]. Notice the required input to stabilize the inverted pendulum in the upper equilibrium is zero, thus $U_r = \mathbb{0}^{N_p n_u}$.

To analyze the numerical robustness of the optimisation, the condition number $(c.n)$ of the Hessian E was calculated and compared between both, the standard and the proposed novel solution using different numeric precision (floats and doubles), and the maximum $c.n_{max}$ of each solution was gathered in table I for all prediction horizons.

To visualize this differences, an example performance of the optimisation is given in figure 1 for the solution with prediction horizon $N_p = 75$ where the $c.n.$ is plotted for both solutions along with the resulting trajectories. It can be seen that the standard solution gives a condition number of up to $c.n. = 1.39e+06$, and presents a difference between both solutions of nearly 6 orders of magnitude larger, which is fairly significant considering the relatively short prediction horizon used. Looking further at table I, the condition number of the standard solution increased as the prediction horizon increased giving differences of up to 13 orders of magnitude for $N_p = 150$, and the Hessian becoming singular for $N_p = 200$ when using double precision and for $N_p > 75$ when using float precision. This ultimately prevents the standard methodologies from using floating precision which leads to faster computation times as detailed in table II. In contrast, the proposed solution maintained steady at $c.n_{max} \approx 3.58\ \forall N_p$ independent of the numeric precision. It should be mentioned that common prediction horizons for the inverted pendulum are relatively long (2 to 4 seconds [3], [27]), which is approximately the time required to swing up and stabilize the system. However, other systems such as the ball-plate apparatus of [29] can present numeric conditioning problems in as low as 1 second, for which the proposed approach offers a viable solution.

Precision	Double		Float	
N_p	DM	STD	DM	STD
75	3.58	1.39e+06	3.58	2.28e+06
100	3.58	4.52e+08	3.58	(Singular)
125	3.58	1.47e+11	3.58	(Singular)
150	3.58	5.02e+13	3.58	(Singular)
200	3.58	(Singular)	3.58	(Singular)

Table I: Maximum Condition Numbers Comparison for Different Prediction Horizons and Numeric Precision. STD and DM refer to the standard and dual mode solution, respectively.

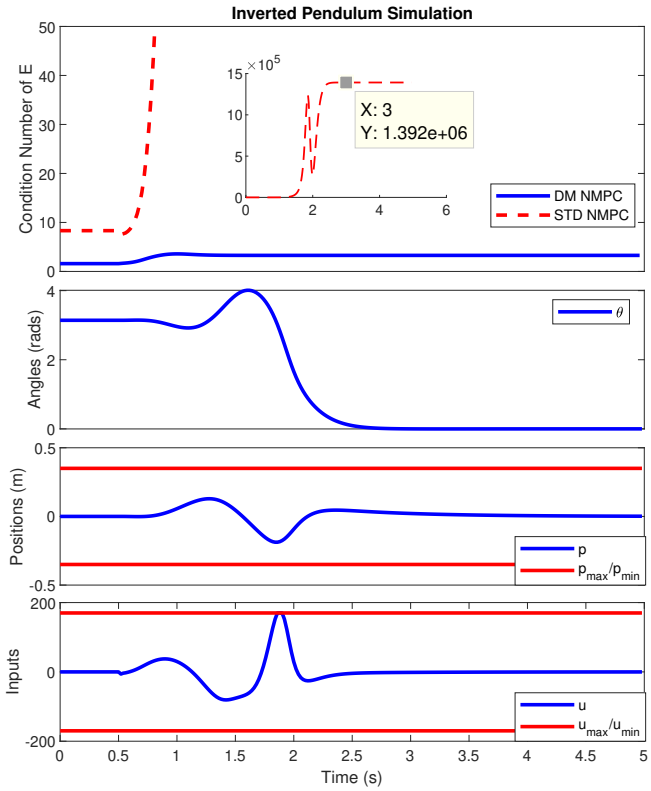


Figure 1: Example numerical conditioning using double precision with initial condition $x_0 = [0, 0, 0, \pi]^T$, $N_p = 75$. STD and DM refer to the standard and dual mode solution, respectively.

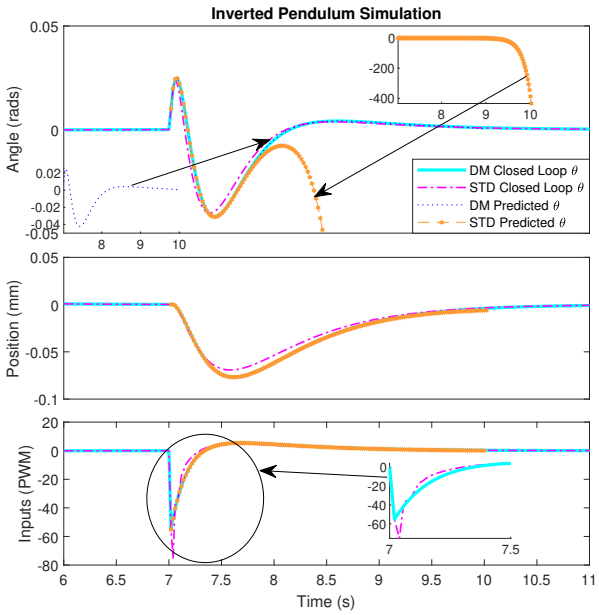


Figure 2: Disturbance Response - Prediction and Closed Loop Comparison for $N_p = 150$ when using Matlab "inv(A)" function. Disturbance of $x_k = x_k + [0, 0.5, 0, 0]^T$ was injected at $t = 7\ (s)$. STD and DM refer to the standard and dual mode solution, respectively.

Remark 4. *It should be noted that the numerical conditioning problem can also be changed by selecting a different number of shooting points as well as the number of intermediate steps of the discretization and linearisation process [3]. However, this doesn't tackle the source of the problem, nor does it provide a general methodology to address it using an arbitrary/desired number of elements to be selected by the user.*

For the interest of the reader, a Matlab code reproducing the results of figure 1 and table I is given at [23].

B. Disturbance Rejection Comparison

Another interesting result was obtained when comparing the responses against disturbance rejection which were observed to present small differences despite the equality of the solutions proven by theorem (1). This was particularly present when using long horizons and, more importantly, when using the weak inverse function "inv(A)" of Matlab to obtain the unconstrained solution, which is known to be less accurate than solving a linear system using $A\backslash b$. To test this, a disturbance of $x_k = x_k + [0, 0.5, 0, 0]^T$ was injected at $t = 7\ (s)$ (continuation from figure 1 - system in upper equlibirum) for which the unconstrained solution satisfies. Figure 2 shows an example of this where the predicted and closed-loop responses are plotted after the disturbance is injected. Only the initial predicted trajectories were plotted to avoid saturation. As it can be seen, the predicted trajectories of the angle using the standard solution (orange dashed curve with markers - visible in the upper right corner) diverged significantly from the closed loop, which in essence resulted in an ill-posed optimization [10] and caused the closed loop solution (magenta dash-dotted curves) to differ as it can be seen from all 3 responses (angles, position and inputs). In contrast, the predictions of the angle using proposed dual mode approach (blue dotted curve - visible in the lower left corner) are indistinguishable from the closed-loop response (cian - solid line curve). Interestingly, the closed-loop responses were identical before the introduction of the disturbance, which suggest that this problem is clearly related to the numeric conditioning of matrix G whose norm grows as big as $||G|| \geq 2.49e + 08$, thus affecting the linear term f significantly when $||\delta x_0|| >> 0$.

It is noted that this anomaly ONLY happened when using the weak inverse function, and it was not present when using the command $A\backslash b$ to obtain the unconstrained solution resulting in the exact same solutions as expected from theorem (1). Nonetheless, it offered an important insight into another potential advantage of the overall methodology.

C. Computation Times

As discussed earlier, the methodology has the disadvantage that further computations are required when compared to the standard approach. To evaluate this, a C++ code based on the EIGEN library was developed following suggestions from

[2], [7] by using the recursive calculations, pre-storage and avoiding unnecessary zero-computations. Moreover, to assess the computation times for the constrained case, 10 iterations of a reduced version of general Primal-Dual Interior Point Methods available in [30], [31], [32] were computed, including the efficient solution of the system (24) with $\delta\bar{C}, \bar{\lambda}$ guesses

$$\begin{bmatrix} E & -M^T \\ \Lambda M^T & C \end{bmatrix} \begin{bmatrix} p_{\delta C} \\ p_\lambda \end{bmatrix} = \begin{bmatrix} -f - E\delta\bar{C} + M^T\bar{\lambda} \\ \mu - C\bar{\lambda} \end{bmatrix} \quad (24)$$

the selection of the initial guess, and the selection of the appropriate step size α that satisfies $M\delta(\bar{C}^{[i]} + \alpha p_{\delta C}) - \gamma \geq \epsilon$ and $\bar{\lambda}^{[i]} + \alpha p_\lambda \geq \epsilon$, where $C = diag(M\delta\bar{C}^{[i]} - \gamma)$ and $\Lambda = diag(\bar{\lambda})$. For more information, the reader is referred to [31]. The resulting code is available in [23].

Remark 5. *For proper operation of the interior point, the constraints must be put in the form $M\delta\hat{C} \geq \gamma$, ie. both must be negated.*

The developed code was run 1000 times for the conditions previously discussed, and the average computation times were stored. The code was compiled using -O3, -mavx and -mfma C-flags to specify the optimisation level, auto-vectorization (avx), and fused multiply-add (fma) operations, respectively, and was tested in a Laptop running Ubuntu 18.04 with an i7-5700 HQ @ 2.7 GHz Intel Processor, and a DDR3 RAM @ 1.6 GHz. A detailed comparison between the average computation times in microseconds is given in table II for prediction horizon $N_p = 75$ using both, the standard and the proposed solution, as well as using different numeric precision (doubles/floats) which allow faster computations. For clarity of where the algorithm introduces the additional computation times, they were decomposed in several intermediate steps, namely; Forward: the time required to generate and store $d_k, A_k, B_k \ \forall \ k = [0, N_p]$; DARE: Computation of DARE backwards $K_k \ \forall k = [0, N_p - 1]$ and computing $\Phi_k \ \forall \ k = [0, N_p - 1]$; Matrices: Computation of all main matrices and vectors $(D, G, H, S, W, F, E, f, M, \gamma)$; Inversion: Solution of $\delta C = -E^{-1}f$, required for the unconstrained solution; QP Steps: Computation of 10 iterations of the Interior-Point QP solution; Unconstrained: Summation of all the steps required for the unconstrained solution; and Constrained: Average constrained computation time.

As it can be seen, the method introduces additional computations mainly in DARE and the computation of all the matrices, particularly because of the computation of S, W, F, f, γ. Nonetheless, the proposed solution remains remarkably close to the standard solution, with only 13% to 20% additional computation time for the unconstrained cases of both precisions; and 3.3% to 3.9% for the constrained cases of both precisions, which is reasonable given the large advantage of up to 3.89 million times better numeric conditioning. Moreover, because the floating point solution using the proposed dual-mode approach is now non-singular, it can be used to solve the optimisation up to $1.5 - 2$ times faster.

IV. Example 2: The Triple Inverted Pendulum

To further illustrate the benefits of the proposed methodology and provide a more complete example that further shows its generalisation capabilities for higher order systems, this section presents its application to a triple inverted pendulum which is a considerable more complex nonlinear system than the single inverted pendulum. Indeed, due to its highly unstable dynamics, the standard condensing based multiple shooting NMPC approach was unable to solve this problem altogether, independently of the prediction horizon used. Thus, this provides an example of a problem that previously was unable to be solved using the latter which further stresses out the importance of the contribution.

Precision	Double		Float	
Step	*DM*	*STD*	*DM*	*STD*
Forward	5	5	3	3
DARE	5	0	4	0
Matrices	306	270	204	168
Inversion	35	35	25	25
10 IP-QP Steps	2190	2149	1044	1037
Unconstrained	351	310	236	196
Constrained	2541	2459	1280	1233

Table II: Comparison of Average Computation Times in microseconds (μs) for prediction horizon $N_p = 75$ and double/float numeric precision. STD and DM refer to the standard and dual mode solution, respectively.

In this paper, the equations of motion for a point-mass triple pendulum provided in [33] were used, combined with the cart acceleration differential equation (22a) with the assumption that the pendulums will have no effect on the cart. This assumption is standard in many approaches present in the literature as the pendulums' effects can be canceled using subordinate/inner acceleration/velocity controllers for the cart as described in [26], [24].

Thus, the equations are given by:

$$\ddot{p} = f_m\dot{p} + ku \quad (25a)$$

$$M(\theta)\ddot{\theta} = -N(\theta)\dot{\theta}^2 - R\dot{\theta} - P(\theta) - f(\theta)(f_m\dot{p} + ku) \quad (25b)$$

where $M(\theta)$, $N(\theta)$, R, $P(\theta)$ and $f(\theta)$ are defined as in [33], and the specific parameters used for our simulation are given in table III. Assuming the state $x_k = [v, \omega_1, \omega_2, \omega_3, p, \theta_1, \theta_2, \theta_3]^T$ with $v = \dot{p}$ and $\omega_i = \dot{\theta}_i$, the system was simulated and linearised using $N_s = 2$ steps of forward euler method as described in [3] with a sampling time of $T_s = 0.02(s)$. The inner step was required to improve the accuracy and stability of the integration method as the system is known to present highly chaotic behavior [33].

Remark 6. *Given the complexity of the system, Matlab's Symbolic Toolbox was used to obtain the expressions of the linearisation terms.*

m_1	0.3	L_1	0.3	R_1	0.1	g	9.81
m_2	0.27	L_2	0.27	R_2	0.1	f_m	-4.67
m_3	0.243	L_3	0.243	R_3	0.1	k	0.065

Table III: Triple Pendulum Parameters

Regarding the optimisation setup, a prediction horizon of $T_p = 2 \ (s)(N_p = 100)$ was selected, and the penalization weights were selected as $q_{k+i} = diag([0.1, 0.2, 0.3, 0.4, 10, 20, 30, 40]) \ \forall i = [1, N_p - 1]$ with

the terminal weight selected as $q_{k+N_p} = 100q_{k+i}$, and the input penalisation term as $r_u = 0.001$. As in the previous example, all the simulations started from the lower equilibrium in steady state ($x_r = x_0 = [0,0,0,0,0,0,0,0]^T$), and a reference of $x_r = [0,0,0,0,0,-\pi,\pi,-\pi]^T$ was introduced at the end of the prediction horizon. Moreover, to relax the optimisation (as it is indeed a much more difficult problem), the position was constrained to $-0.5 \leq p \leq 0.5$ whilst keeping the same input constraints as for the single inverted pendulum of section III, ie. ($-170 \leq u \leq 170$).

To further improve the performance of the underlying SQP method, an additional exponentially decaying penalisation term defined as $\delta(r_u)_{k+i} = 1000r_u(\alpha)^i \ \forall i = [1, N_p]$ with $\alpha = (0.01)^{\frac{1}{N_p}}$ was imposed on the input deviation $\delta\hat{U}$ which modified the original cost function (10a) to:

$$J_{\delta R} = J + \left(\delta\hat{U}\right)^T \delta R \left(\delta\hat{U}\right) \tag{26}$$

where $\delta R = diag([\delta(r_u)_{k+i}]) \in \mathbb{R}^{N_p n_u \times N_p n_u}$, which modified the Hessian E and linear term f to:

$$E_{\delta R} = H^T Q H + F^T(R + \delta R)F \tag{27a}$$

$$f_{\delta R} = -\left[H^T Q(X_r - \bar{X} - D - G\delta x_0) - F^T R(\bar{U} - U_r) - F^T(R + \delta R)(S + W\delta x_0)\right] \tag{27b}$$

Although the performance can also be improved by using proper step-size of the Newton-method, this additional term was motivated by observing that the prediction errors due to linearisation grow as they move forward through the horizon. Therefore, by preventing large deviations at the beginning of the horizon, the prediction errors in future time-steps are reduced which consequently improves the contraction rate of the underlying Newton-method. On the other hand, it can be proved that the solution with this added penalisation term only affects the rate of convergence towards the solution, but does not change the solution itself. Finally, it is trivial to show that theorem 1 still holds with this modification.

Figure 3 shows a $T = 10(s)$ simulation of a swing-up and stabilization of the triple inverted pendulum problem with a disturbance of $x_k = x_k + [0,0,0,0.1,0,0,0,0]^T$ introduced at $t = 5\ (s)$ for which the unconstrained solution satisfies. Of particular interest is the figure on the lower-right corner where the steady state condition number $(c.n.)_{ss} \approx 252$ can be seen which, for this system, is naturally much higher than that of the single inverted pendulum presented in figure 1. Indeed, the latter undergoes critic points during the swing up reaching a maximum of $(c.n.)_{max} \approx 811$, approximately 3.2 times higher, which once again shows the complexity of the system at hand. Nonetheless, the method preserves the expected properties of low conditioning number which protect the solution from numerical instability, and the resulting controller is observed to perform well against disturbances.

Remark 7. *It is worth noting that linearising the system at the upward equilibrium without the proposed approach had a condition number of the "would-be" optimisation of* $(c.n.) = 3.07 \times 10^{23}$. *Thus, considering the solution can undergo the aforementioned critic points, it is not surprising the standard method was unable to be applied.*

V. Conclusion

This paper presents a novel Dual-Mode NMPC methodology that uses stable prediction models to obtain numerically robust solutions for condensing based multiple shooting NMPC frameworks which are particularly well suited for unstable systems. A proof of the equivalency of the solution with the standard single/multiple-shooting solution is given, which consequently results in the exact same stability and convergence properties. The method uses a stabilizing gain obtained from solving the Time-Varying DARE backwards in time along the shooting trajectory. The proposed approach differs from all previously proposed Dual-Mode NMPC schemes in the sense that it aims at stabilizing the trajectory using time-varying gains, rather than stabilizing the states to the origin as the standard methodologies, typically using a single gain obtained from LQR. Although the methodology was derived particularly for a multiple-shooting sequential (condensing-based) solution, it can be applied for multiple-shooting scenarios using simultaneous approaches. To achieve real-time performance, the NMPC was deployed using Real-Time Iterations, and an overall algorithm is presented along the paper summarizing the proposed approach. Simulations of an inverted pendulum, and its extension - the triple inverted pendulum, are presented focusing on the numerical conditioning, disturbance rejection and computation time differences compared with the standard solution, demonstrating the advantages and weaknesses of the methodology. It is noted that the triple inverted pendulum case was unable to be applied with the standard method, which provides further evidence of the importance of this contribution.

Having observed the benefits of the proposed approach which clearly result in significant improvements whilst offering a general procedure to tackle unstable systems for NMPC, future work will aim to merge the proposed approach with the ACADO toolkit which currently offers no option for stable-predictive models in their auto-generation routines, despite the generic extension steps required. Moreover, to reduce the computation time of the optimisation, efficient parameterised solutions based on Laguerre polynomials [34] or Blocking approaches [29] will be explored.

To the best of the authors knowledge, this is the first paper offering a generalisable dual mode closed-loop paradigm for multiple shooting condensing-based NMPC schemes, and proving the equality of the latter with the standard solution. The MATLAB and C++ files used to obtain the results in this paper can be found in [23].

Acknowledgment

The first author would like to thank CONACyT, Mexico for funding this research.

References

[1] B. Houska, H. J. Ferreau, and M. Diehl, "Autogenerating microsecond solvers for nonlinear MPC: A tutorial using ACADO integrators," *Optimal Control Applications and Methods*, vol. 36, pp. 685–704, 2015.

[2] B. Houska, H. J. Ferreau, and M. Diehl, "An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range," *Automatica*, vol. 47, no. 10, pp. 2279–2285, 2011.

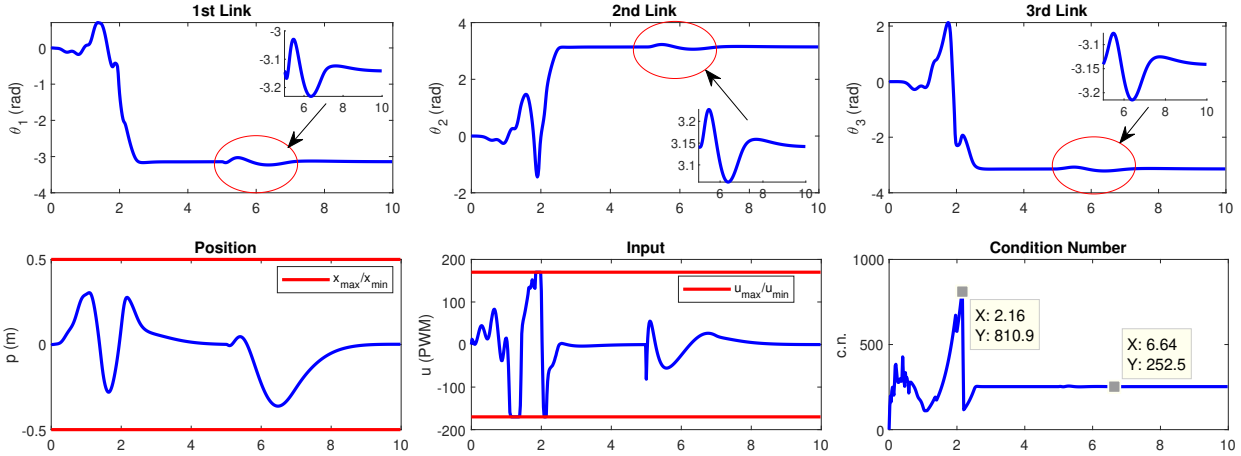


Figure 3: Triple inverted pendulum swing up and stabilisation simulation with disturbance of $x_k = x_k + [0, 0, 0, 0.1, 0, 0, 0, 0]^T$ injected at $t = 5\ (s)$. The maximum and steady state conditioning numbers are shown in the lower-right figure for reference.

[3] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear MPC: bridging the gap via the real-time iteration," *International Journal of Control*, vol. 7179, no. November, pp. 1–19, 2016.

[4] M. Diehl, H. G. Bock, and J. P. Schlöder, "A Real-Time Iteration Scheme for Nonlinear Optimization in Optimal Feedback Control," *SIAM Journal on Control and Optimization*, vol. 43, no. 5, pp. 1714–1736, 2005.

[5] J. Kalmari, J. Backman, and A. Visala, "A toolkit for nonlinear model predictive control using gradient projection and code generation," *Control Engineering Practice*, vol. 39, pp. 56–66, 2015.

[6] S. Gros, R. Quirynen, and M. Diehl, "Aircraft control based on fast nonlinear MPC & multiple-shooting," *Proceedings of the IEEE Conference on Decision and Control*, no. 1, pp. 1142–1147, 2012.

[7] M. Vukov, A. Domahidi, H. J. Ferreau, M. Morari, and M. Diehl, "Auto-generated algorithms for nonlinear model predictive control on long and on short horizons," *52nd IEEE Conference on Decision and Control*, pp. 5113–5118, 2013.

[8] R. Quirynen, M. Vukov, and M. Diehl, "Multiple Shooting in a Microsecond," in *Multiple Shooting and Time Domain Decomposition Methods*, vol. 9, pp. 183–202, Springer, Cham, 2015.

[9] J. A. Rossiter, B. Kouvaritakis, and M. J. Rice, "A numerically robust state-space approach to stable-predictive control strategies," *Automatica*, vol. 34, no. 1, pp. 65–73, 1998.

[10] J. A. Rossiter, *A first course in predictive control*. Boca Raton: CRC Press, Taylor & Francis, 2nd ed. ed., 2018.

[11] R. Huang, S. C. Patwardhan, and L. T. Biegler, *Adaptive quasi-infinite horizon NMPC of a continuous fermenter*, vol. 9. IFAC, 2010.

[12] W. E. A. Ren, "Input-to-state stabilizing sub-optimal NMPC with an Application to DC-DC converters," *International Journal of Robust and Nonlinear Control*, vol. 18, no. October 2014, pp. 890–904, 2008.

[13] A. Arpornwichanop and P. Kittisupakorn, "Dual mode NMPC for regulating the concentration of exothermic reactor under parametric uncertainties," *Journal of Chemical Engineering of Japan*, vol. 37, no. 6, pp. 698–710, 2004.

[14] B. Khan, *Efficient Parameterise solutions of predictive control*. Ph.d. thesis, The University of Sheffield, 2013.

[15] Q. Weiwei, H. Bing, L. Beixuan, Y. Renping, and L. Gang, "An improved dual-mode robust nonlinear MPC with one-step set optimization," *Chinese Control Conference, CCC*, vol. 2015-Septe, pp. 4004–4009, 2015.

[16] H. Chen and F. Allgöwer, "A computationally attractive nonlinear predictive control scheme with guaranteed stability for stable systems," *Journal of Process Control*, vol. 8, no. 5-6, pp. 475–485, 1998.

[17] L. P. Gutiérrez González, D. Odloak, O. Sotomayor, and H. Alvarez, "A dual mode MPC scheme for nonlinear processes," *IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 17, no. 1 PART 1, pp. 12159–12164, 2008.

[18] M. Bacic, M. Cannon, and B. Kouvaritakis, "Feedback linearization constrained NMPC for bilinear systems," *International Conference on Control and Automation*, no. June, pp. 261–265, 2003.

[19] K. J. Ramírez, L. M. Gómez, and H. Alvarez, "Control predictivo no lineal basado en modelo por modo dual con estabilidad garantizada," *Ingeniería Y Competitividad*, vol. 16, no. 1, pp. 23–34, 1969.

[20] M. Neunert, C. De Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli, "Fast nonlinear Model Predictive Control for unified trajectory optimization and tracking," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2016-June, no. ICRA, pp. 1398–1404, 2016.

[21] M. Diehl, L. Magni, and G. De Nicolao, "Efficient NMPC of unstable periodic systems using approximate infinite horizon closed loop costing," *Annual Reviews in Control*, vol. 28, no. 1, pp. 37–45, 2004.

[22] M. Diehl, L. Magni, and G. De Nicolao, "Online NMPC of a looping kite using approximate infinite horizon closed loop costing," *IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 36, no. 18, pp. 519–524, 2003.

[23] O. J. G. Villarreal, "Dual Mode Stable Prediction Models for Numerically Robust NMPC using Real-Time Iterations." Code Ocean, Nov 2019. Available in https://doi.org/10.24433/CO.9069068.v1.

[24] T. Glück, A. Eder, and A. Kugi, "Swing-up control of a triple pendulum on a cart with experimental validation," *Automatica*, vol. 49, no. 3, pp. 801–808, 2013.

[25] L. Wang, *Model Predictive Control System Design and Implementation Using Matlab*. Springer, 2009.

[26] M. Alamir and A. Murilo, "Swing-up and stabilization of a Twin-Pendulum under state and control constraints by a fast NMPC scheme," *Automatica*, vol. 44, no. 5, pp. 1319–1324, 2008.

[27] a. Mills, A. Wills, and B. Ninness, "Nonlinear model predictive control of an inverted pendulum," *American Control Conference*, pp. 2335–2340, 2009.

[28] M. Alamir, "Fast NMPC: A reality-steered paradigm: Key properties of fast NMPC algorithms," *2014 European Control Conference, ECC 2014*, no. 4, pp. 2472–2477, 2014.

[29] R. Cagienard, P. Grieder, E. C. Kerrigan, and M. Morari, "Move blocking strategies in receding horizon control," *Journal of Process Control*, vol. 17, no. 6, pp. 563–570, 2007.

[30] A. Forsgren and P. E. Gill, "Primal-dual interior methods for nonconvex nonlinear programming," *SIAM Journal on Optimization*, vol. 8, no. 4, pp. 1132–1152, 1998.

[31] M. H. Wright, "The interior-point revolution in optimization: History, recent developments, and lasting consequences," *Bulletin of the American Mathematical Society*, vol. 42, no. 1, pp. 39–56, 2005.

[32] J. Li, J. Lv, and J. Jian, "A globally and superlinearly convergent primal-dual interior point method for general constrained optimization," *Numerical Mathematics*, vol. 8, no. 3, pp. 313–335, 2015.

[33] I. Rivas-Cambero and J. M. Sausedo-Solorio, "Dynamics of the shift in

resonance frequency in a triple pendulum," *Meccanica*, vol. 47, no. 4, pp. 835–844, 2012.
[34] J. A. Rossiter, L. Wang, and G. Valencia-Palomo, "Efficient algorithms for trading of feasibility and performance in predictive control," *International Journal of Control*, vol. 83, no. 4, pp. 789–797, 2010.

Oscar Julian Gonzalez Villarreal received his B. Eng in Mechatronics Engineering from ITESM, Campus Monterrey, Mexico in 2014, and his M. Sc. in Autonomous Vehicles Dynamics and Control from Cranfield University, UK in 2016. He is currently undergoing PhD studies in The University of Sheffield under the supervision of Dr. A. Rossiter with main focus on Efficient NMPC Solutions for Fault-Tolerant Flight Control Systems.

John Anthony Rossiter is a Reader at the University of Sheffield, having previously worked at Loughborough University and received his PhD from Oxford University in 1990. His main research interest is predictive control where he has published widely, including a popular text book. He has devoted significant effort to understanding and developing effective practices for learning and teaching, for which he has received some awards and currently serves as chair of the IFAC TC 9.4 and IEEE TC on control education. Biography text here.

This is the last page.