Design and Certification of Industrial Predictive Controllers

Ontwerp en certificering van industriële voorspellende regelaars

Abhishek Dutta

Promotoren: prof. dr. ir. R. De Keyser, dr. ir. C.-M. Ionescu
Proefschrift ingediend tot het behalen van de graad van
Doctor in de Ingenieurswetenschappen: Werktuigkunde-Elektrotechniek

Vakgroep Elektrische Energie, Systemen en Automatisering
Voorzitter: prof. dr. ir. J. Melkebeek
Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2014 - 2015

UNIVERSITEIT
GENT

ISBN 978-90-8578-719-8
NUR 992
Wettelijk depot: D/2014/10.500/65

Design and Certification of Industrial Predictive Controllers

Dissertation submitted to obtain the academic degree of Doctor of Electromechanical Engineering

ir. Abhishek Dutta

Ghent University
Junior Member, Wolfson College Cambridge

Promoters: prof. dr. ir. Robin De Keyser, dr. ir. Clara Ionescu
Ghent University

Advisors: prof. dr. ir. Jan Maciejowski, dr. ir. Edward Hartley
University of Cambridge

Members of the examination board:

prof. dr. ir. Patrick De Baets (chairman)	Ghent University
prof. dr. ir. Dirk Aeyels (secretary)	Ghent University
prof. dr. ir. Hendrik De Bie	Ghent University
prof. dr. ir. Robin De Keyser	Ghent University
dr. ir. Clara Ionescu	Ghent University
prof. dr. ir. Jan Swevers	KU Leuven
prof. dr. ir. Mircea Lazar	TU Eindhoven
prof. dr. ir. Jan Maciejowski	University of Cambridge

Foreword

This thesis is the product of my interaction with a large number of people, with whom I have had the pleasure to discuss a wide range of topics in control, mathematics and engineering.

I would first like to thank my promoters, Prof. Robin De Keyser and Dr. Clara Ionescu, for listening to my sometimes very scattered thoughts and helping me channel them in the right direction. Throughout my studies they have taken an active interest in my work and given ample opportunity to collaborate and present at the highest levels.

I would also like to thank my advisers, Prof. Jan Maciejowski and Dr. Edward Hartley, for believing in my radical scientific ideas and bringing them to valid conclusions.

I take this opportunity to thank the agency for Innovation by Science and Technology (IWT-Vlaanderen) for providing me with generous financial support during the course of my research within the framework of Learning Control for Production Machines (LeCoPro) project (Grant No. 80032). My interaction with the members of the LeCoPro project has also been highly beneficial and especially Dr. Bruno Depraetere, who saw me through a challenging period of time.

The Control Lab in Ghent has been my home for many years and the Control Lab at Cambridge my home for a significant duration and everybody who has passed through has contributed to my understanding of control systems in some way or other. My thanks to Dr. Bart Wyns for his moral support, Bamdev for the useful discussions, Maryna and Erland for their help with translation, Stefana and Mirjana for being my two favourite students turned friends and Yu for being a kind lab mate.

Finally, I would like to thank my father, mother and grandmother for keeping up with someone like me who is cursed with a thinking mind.

Ghent, March 2014
Abhishek Dutta

Table of Contents

List of Figures

Nederlandse samenvatting –Summary in Dutch–

De meeste fysische systemen hebben begrenzingen op de grootte van hun ingangs- en uitgangssignalen door de aanwezigheid van fysische en/of economische beperkingen en/of om veiligheidsredenen. Bij traditionele regelaars wordt in de ontwerpfase geen rekening gehouden met deze begrenzingen wat er toe kan leiden dat deze worden overschreden. Deze overschrijding kan instabiel gedrag en/of ernstig rendementsverlies veroorzaken afhankelijk van de toepassing. Model-gebaseerde voorspellende regelsystemen (MPC-Model based Predictive Control) voorzien een systematische behandeling van alle vormen van begrenzingen wat heeft geleid tot een enorme impact op de praktijk van de industriële regeltechniek. MPC is een vorm van regelen waarbij de actuele processtuuringang wordt verkregen door bij elke bemonstering een eindige horizon open kring begrensd optimaal controleprobleem op te lossen, gebruik makend van de huidige toestand (dwz voorgaande in- en uitgangswaarden) van de installatie als initiële toestand. De optimalisatie levert een optimale controle sequentie en de eerste controlewaarde in deze sequentie wordt op de installatie toegepast. Bij de volgende bemonstering vindt een nieuwe optimalisatie plaats gebaseerd op de nieuwe metingen; dit is het principe van regeling met terugwijkende horizon (receding horizon principle).

Drie decennia zijn verstreken sinds enkele fundamentele publicaties van industriële en academische onderzoekers aanleiding gaven tot een grote toename van onderzoek en commerciële en industriële activiteiten op het vlak van MPC. De verbetering van de efficiëntie van het on-line optimalisatie-gedeelte heeft geleid tot de verspreiding van MPC in mechanische en mechatronische systemen, naast de eerdere toepassingen in procesbesturing en petrochemische installaties. Gezien de noodzaak voor het ontwerpen van veilige regelaars, vooral voor productiemachines, is het essentiëel om de stabiliteit van het gesloten-kring feedback regelsysteem te garanderen. De enorme vooruitgang geboekt door de academische gemeenschap in het waarborgen van de stabiliteit door middel van een MPC aanpak in het toestandsdomien kon niet altijd rechtstreeks toegepast worden in een industriële omgeving. De reden is dat de meeste van de industriële implementaties vermijden om stabiliserende terminale voorwaarden te gebruiken (dwz terminale penalisaties, terminale begrenzingen, terminale regelaars) en in het ingangs-uitgangs domein werken. Een groot deel van dit proefschrift gaat over het ontwerp en de certificering van de haalbaarheid en stabiliteit van de ingangs-uitgangs MPC

regelaars voor industriële toepassingen zonder terminale voorwaarden.

De meeste fysische systemen zijn niet-lineair en een bepaalde klasse van deze systemen zijn bovendien gedistribueerd dwz systemen van erg grote schaal die zijn samengesteld uit verschillende subsystemen met interactie. De aanwezigheid van praktische begrenzingen in deze gevallen in combinatie met de noodzaak van het voorspellen van toekomstige gecontroleerde scenario's, maakt MPC tot de logische keuze voor niet-lineaire gedistribueerde systemen. De uitwerking van de niet-lineaire MPC (NMPC) vereist een iteratief mechanisme en de gedistribueerde NMPC (DNMPC) vereist een andere iteratieve procedure tussen de verschillende NMPCs die de subsystemen controleren. De rest van de bijdrage van dit proefschrift ligt in het bewijzen van convergentie van NMPC iteraties en het garanderen van een verbetering van de totale kost met elke iteratie in het geval van een DNMPC zonder terminale voorwaarden. Uiteraard hebben al deze real-life systemen een tijdsafhankelijke dynamica. De ontwikkeling van adhoc leertechnieken om met deze veranderingen om te gaan is een supplementaire technologische bijdrage van deze thesis.

Het tweede hoofdstuk brengt de drie belangrijkste formuleringen van voorspellende regeling die al meer dan twee decennia bestaan samen; er wordt gepoogd om een algemene equivalentie tussen deze technieken te bekomen. De onderzoeksgemeenschap actief in adaptieve regeling heeft MPC geformuleerd op basis van ingangs-uitgangs modellen en filtertechnieken. De Extended Prediction Self-Adaptive Control (EPSAC) wordt beschouwd als de basis strategie uit deze groep van online MPC regelaars. Het eerste equivalent van EPSAC is gebouwd voor de MPC-regelaars die gebaseerd zijn op Diophantine vergelijkingen die leiden tot een gesloten-vorm oplossing van EPSAC in het onbegrensde geval. Het tweede equivalent wordt afgeleid voor de groep van MPC regelaars gebaseerd op toestandsmodellen, die gebruikt worden voor het afleiden van een expliciete oplossing voor de begrensde EPSAC. Dit hoofdstuk legt niet alleen de grondslag voor de stabiliteitsanalyse van EPSAC via de afgeleide equivalente formuleringen maar het helpt ingenieurs ook om deze drie MPC technieken te gebruiken en te analyseren.

Bijna alle MPC onderzoekers hebben stabiliteit van toestandsruimte MPC aangetoond door terminale voorwaarden toe te voegen. In tegenstelling tot de industriële werkwijze vermindert dit de haalbare regio waarin het systeem kan functioneren. Het derde hoofdstuk is gewijd aan de haalbaarheid en stabiliteitsanalyse van de ingangs-uitgangs MPC zonder kunstmatige terminale voorwaarden, waarbij het aantal wijzigingen in de toekomstige stuursequentie minder is dan de horizon waarover de proces-dynamica wordt voorspeld. De haalbaarheid van de oplossing in aanwezigheid van begrenzingen kan ten allen tijde worden gegarandeerd en voor alle verstoringen, indien de oorspronkelijke toestand zich in een set bevindt die onveranderlijk is, dwz. dat de toekomstige toestandstrajectories binnen deze set blijven. De haalbaarheid met onbeperktheid in de tijd zorgt er zo voor dat trajectories altijd begrensd blijven en dit leidt rechtstreeks tot het begrip van praktische

stabiliteit. Dit wordt op twee manieren bekomen: (i)'persistente haalbaarheid' wat gebaseerd is op het bestaan van een haalbare oplossing en (ii)'recursieve haalbaarheid' wat de berekening vereist van de expliciete oplossing van het MPC probleem. Deze hulpmiddelen kunnen worden gebruikt door industriële regeltechniekers om het werkveld te identificeren waar hun implementaties gegarandeerd veilig zijn en tevens als richtlijnen voor het ontwerp van veilige MPC regelaars.

In hoofdstuk vier wordt adaptieve penalisatie gebruikt als een mechanisme om asymptotische stabiliteit van de nominale regelkring zonder terminale voorwaarden te bekomen, maar alleen voor repetitieve systemen met enkel begrenzingen op de procesingang. Dit wordt bereikt door de gewichten van de toekomstige stuuracties aan te passen zodanig dat de ingangsbegrenzingen inactief worden. Dan kan de karakteristieke veelterm voor de aangepaste lineaire MPC afgeleid worden wat de overeenkomstige stabiliteitsmarges oplevert. Verder wordt het dynamisch vernauwen van de grenzen over de voorspellings horizon ook gebruikt ter verbetering van de robuuste haalbaarheid. De techniek werd met veelbelovende resultaten uitgetest op de regeling van een mechanisch massa-veer-demper systeem met sensor en actuator op verschillende locaties (non-collocated control).

In de praktijk zijn dynamische systemen met begrenzingen niet-lineair vanwege de onderliggende fysica of variatie van de parameters. De optimalisering in de NMPC formuleringen zoals de niet-lineaire EPSAC (NEPSAC) maakt gebruik van een iteratieve techniek. Hoofdstuk vijf begint met het produceren van een formeel bewijs dat NEPSAC convergeert naar een lokaal optimale oplossing en een tuning recept om deze eigenschap te garanderen. In een industriële omgeving overheersen twee soorten niet-lineariteiten. De eerste soort komt voor wanneer het systeem kan worden beschreven als een stuksgewijs affiene (PWA) vorm. De ontwikkelingen inzake persistente en recursieve haalbaarheid en stabiliteit zonder terminale voorwaarden werden uitgebreid voor PWA dynamiek. Ten tweede kan een niet-lineair systeem een schakelende dynamiek hebben. Voor dergelijke systemen, is een NMPC architectuur op twee niveau's ontworpen waarbij het hogere niveau het schakelen tussen de NMPCs op de lagere niveaus coördineert. Deze regelstrategie werd uitgetest op een proefstand voor het in- en uitschakelen van mechanische koppelingen. De gepresenteerde NMPC technieken zijn snel, eenvoudig en maken geen gebruik van terminale voorwaarden; ze kunnen dus gemakkelijk worden ingezet door industriële gebruikers.

In de praktijk bestaan veel grootschalige systemen uit interagerende, begrensde, niet-lineaire subsystemen, die elk hun eigen gesloten-kring regeling hebben. Dit komt omdat een volledig gecentraliseerde regelaar te duur zou uitvallen door de grote hoeveelheid berekeningen en communicatie. Gedistribueerde NMPCs zijn een logische keuze omdat ze via de voorspellingen rekening kunnen houden met de begrenzingen en met de naburige interacties. In hoofdstuk zes worden NEPSAC controllers zonder terminale voorwaarden gebruikt om een DNMPC opstelling uit te bouwen waarbij elke NEPSAC de totale kost optimaliseert met betrekking tot

zichzelf en eenmalig communiceert aan de aangrenzende buurman. Een formeel bewijs wordt ontwikkeld om aan te tonen dat een dergelijke strategie gegarandeerd tot verbetering leidt van de totale kost. Een hydrostatische aandrijflijn wordt gebruikt als testsysteem om aan te tonen dat de voorgestelde DNMPC aanpak toepasbaar is op snelle, interactieve, niet-lineaire, begrensde industriële systemen.

Terminale voorwaarden zijn in essentie moeilijk te berekenen, ze kunnen de prestaties verminderen en worden niet gebruikt in de industrie. De belangrijkste bijdrage van dit proefschrift is dan ook een systematische ontwikkeling en analyse van MPC zonder terminale voorwaarden voor lineaire, niet-lineaire en gedistribueerde systemen. Dit wordt ondersteund door nieuwe theoretische instrumenten voor het aantonen van de haalbaarheid, de stabiliteit en de convergentie van de D/N/MPC regelaars en door testresultaten op industriële benchmark-systemen.

English summary

Most physical systems have limitations in the size of their inputs and outputs due to the presence of physical, economic and safety constraints. Traditional controllers do not consider these constraints during the design phase which may lead to their violation. This violation, depending on the industrial application may cause unstable behaviour and/or severe loss in efficiency. Model Predictive Control (MPC) provided a systematic means of handling all forms of constraints leading to tremendous impact on industrial control practice. MPC is a form of control in which the current control action is obtained by solving, at each sampling instant, a finite horizon open-loop constrained optimal control problem, using the current state (i.e. past inputs, past outputs) of the plant as the initial state. The optimization yields an optimal control sequence and the first control value in this sequence is applied to the plant. At the next sampling instant a new optimization is performed based on the new measurements; this marks the idea of receding horizon control.

Three decades have passed since milestone publications by several industrial and academic researchers spawned a flurry of research and commercial, industrial activities on MPC. The improvement in efficiency of the on-line optimization part of MPC led to its adoption in mechanical and mechatronic systems from process control and petrochemical applications. Given the need for designing safe controllers, especially for production machines, ensuring stability of the closed-loop MPC is quintessential. However, the massive strides made by the academic community in guaranteeing stability through state-space MPC have not always been directly applicable in an industrial setting. The reason being that most of the industrial implementations avoided using stabilizing terminal conditions (i.e. terminal penalty, terminal constraint, terminal control) and worked in the input-output domain. A major part of this thesis is concerned with design and certification of feasibility, stability of input-output MPC controllers for industrial applications without terminal conditions.

Most physical systems in practice are nonlinear and a class of these are distributed i.e. large scale systems composed of interacting subsystems. The presence of practical constraints in these cases combined with the need for forecasting future controlled scenarios makes MPC a natural choice for nonlinear and distributed systems. The nonlinear MPC (NMPC) realization requires an iterative mechanism and the distributed NMPC (DNMPC) requires another iterative procedure between the several NMPCs controlling the subsystems. The rest of the contribution of this

thesis remains in proving convergence of NMPC iterations and guaranteeing an improvement in the overall cost with every iteration in the case of DNMPC, both without terminal conditions. Needless to say, all these real-life systems have time-varying dynamics. A minor technological contribution has been the development of adhoc learning techniques to cope with these changes.

The second chapter brings together the three main families of predictive control formulations which have existed for more than two decades; an attempt is made to establish a generalized equivalence between these techniques. The adaptive control community formulated MPC based on input-output models and filtering techniques. Extended Prediction Self-Adaptive Control (EPSAC) is considered as the baseline representative strategy from this family of online MPC controllers. The first equivalence of EPSAC is constructed to the MPC controllers based on Diophantine equations leading to a closed-form solution of EPSAC in the unconstrained case. The second equivalence of EPSAC is given towards the family of state-space based MPC controllers which is used to derive an explicit solution for constrained EPSAC. This chapter not only prepares the groundwork for stability analysis of EPSAC through the derived equivalent formulations but also helps an engineer to use and analyze these three MPC techniques.

Almost all the MPC researchers have shown stability of state-space MPC by adding terminal conditions, which is contrary to the industrial practice and reduces the feasible region where the plant can operate. The third chapter is devoted to feasibility, stability analysis of input-output MPC without imposing artificial terminal conditions and where fewer changes are made to future controls than the horizon over which the process dynamics are predicted. Constraint satisfaction (i.e. feasibility) can be guaranteed for all time and for all disturbances if the initial state is inside a set which is invariant i.e. future state evolutions remain within this set. Infinite time feasibility thus ensures that trajectories remain bounded always, directly leading to the notion of practical stability. This is achieved in two ways : (i) 'persistent feasibility' that relies on the existence of a feasible solution and (ii) 'recursive feasibility' that requires computation of the explicit (particular/optimal) solution to the MPC problem. These tools can be used by the industrial control practitioners to identify the operating region where their implementations are certified safe and also as guidelines for designing safe MPC controllers.

In chapter four, penalty adaptation is used as a mechanism to deliver a more rigorous asymptotic stability derivation of the nominal closed loop MPC without terminal conditions, but only for input constrained repetitive systems. This is achieved by adjusting the weights of the future control moves in a way that makes the constraints inactive. Hence closed form polynomial expression for the adapted linear MPC can be derived which allows to specify the corresponding stability margins. Further, dynamic tightening of constraints through the future horizon is used as a means of improving robust feasibility. The technique is tested for the control of non collocated (actuator, sensor act at different locations) systems with

promising results.

In practice, constrained dynamical systems are nonlinear due to the underlying physics, space or parameter variance. The optimization associated with the NMPC formulations such as the nonlinear EPSAC (NEPSAC) uses an iterative technique. Chapter five starts by giving a formal proof of convergence of NEPSAC to locally optimal solution and a tuning prescription to guarantee this property. In an industrial setting, two types of non-linearity are predominant. Firstly, when the system can be expressed in a piecewise affine (PWA) form. The developments on persistent and recursive feasibility, stability without terminal conditions have been extended for PWA dynamics. Secondly, a nonlinear system can have switching dynamics. A two-level NMPC architecture has been designed for such systems, in which the higher level coordinates the switching between the low level NMPCs and is tested for the smooth engagement control of clutches. The presented NMPC techniques are fast, simple and do not use terminal conditions, thus readily adoptable by the industry.

Many large scale systems in practice are composed of interacting constrained nonlinear subsystems, each running their own closed-loop controls. This is because one fully centralized controller is expensive in terms of both computation and communication. Distributed NMPCs are a natural choice as they can account for the constraints and the neighbouring interactions through predictions. In chapter six, NEPSAC controllers without terminal conditions are used to construct the DNMPC setup where each NEPSAC optimizes a global cost with respect to itself and communicates once to its adjacent neighbour. A formal proof is developed to show that such a policy leads to guaranteed improvement in the global cost. A hydrostatic drivetrain is used as a benchmark to show that the proposed DNMPC approach is applicable to fast, interacting, nonlinear, constrained industrial plants.

In essence terminal conditions are difficult to compute (requires involvement of the designer which is not so simple), may compromise performance and are not used in the industry. The main contribution of the thesis is a systematic development and analysis of MPC without terminal conditions for linear, nonlinear and distributed systems (the industrial engineer in this case does not need to know what is under the hood of the certifier). This has been supported by new theoretical tools for proving feasibility, stability and convergence of the D/N/MPC controllers and by test results on industrial benchmark systems.

1 Introduction

1.1 Introduction

Three decades have passed since the seminal papers on Model Heuristic Predictive Control [3], Dynamic Matrix Control (DMC) [4], Extended-horizon Self-Adaptive Control (EPSAC) [5] and Generalized Predictive Control (GPC) [6] appeared. These publications, along with earlier papers reporting similar ideas generated an unprecedented level of excitement within the process control community and ushered in the era of Model Predictive Control (MPC). Three decades later, it is regarded by many as one of the most important developments in control engineering, mainly because it is the only systematic way of handling hard constraints through online optimization [7].

MPC is a form of control in which the current control action is obtained by solving, at each sampling instant, a finite horizon open-loop constrained optimal control problem, using the current state (i.e. past inputs, past outputs) of the plant as the initial state. The optimization yields an optimal control sequence and the first control value in this sequence is applied to the plant. At the next sampling time a new optimization is performed based on the new measurements; this is the idea of receding horizon control. MPC has had a tremendous impact on industrial control practice. Nowadays it is found in the control rooms of almost every petrochemical plant [8]. Equally important is the impact MPC made on control research. Through the effort of many researchers, MPC now rests on a firm theoretical foundation [9], coupled with development in fast optimization algorithms that are orders-of-

magnitude faster [10]. These developments enabled the implementation of MPC for applications involving mechanical and electronic systems [11].

1.2 Motivation

All physical systems have complex dynamics and the requirements on the performance of the controller are usually quite demanding. Additionally, their inputs and outputs are limited in size due to the presence of safety, physical limits, etc. Furthermore, an application might also require a certain level of performance, which can be translated to a synergetic combination of economic cost function and constraints.

Omitting these constraints in the controller design and analysis phase may lead to a control action that results in the violation of these constraints. Depending on the criticality of the constraint in the associated industrial application, this violation might result in unstable behaviour causing system failure, possibly even leading to loss of human life. Similarly, it is possible that a controller which does not take into account changing environmental conditions, would drive the system into an unsafe region.

Given this need for designing safe controllers, especially for production systems, this thesis concentrates on designing receding horizon constrained optimal controllers with practical adaptation mechanisms and guarantee that the constraints will not be violated.

Next to the developments in process industry and adaptive control community stemming from the input-output/transfer-function based MPC, the research literature on MPC began to adopt a state-space formulation and rigorous feasibility and stability proofs followed by means of adding terminal weights, terminal constraints and terminal control law [9]. However, a theoretical framework is not very useful unless it can be implemented for practical systems. On the other hand, the transfer function formulation was more intuitive and used matured system identification tools. Next, the ad hoc terminal conditions reduced the feasibility of the problem, apart from being difficult to compute. Thus, the industrial implementations largely continued to use input-output formulations of MPC and avoided the use of the mentioned terminal stabilizing ingredients, thus appearing to defy using mathematical analysis.

In this thesis, practical stability, i.e. boundedness of trajectories, is proven for these industrial input-output MPC formulations without terminal conditions, by deducing the largest ‘safe’ region from which the closed-loop system can never exit, thus remaining bounded for infinite time. This region could be smaller than the pre-specified region defined by the safety and performance constraints. The reason for this is that the specified constraints do not necessarily take into account the actual physics of the process. For example, the speed limit for cars does not

always take into account the conditions of the road and if one encounters a very sharp bend in the road then this limit might not be safe.

Next, another means of certifying asymptotic stability for input constrained systems by penalty adaptation is introduced for vibrating systems. Many real-life systems have switched nonlinear dynamics in nature, e.g. a car moving over a terrain with changing slopes. Many others can be viewed as a composition of interacting subsystems like harvester machines, wind mills etc. Further in this thesis, practical solutions for controlling such nonlinear and distributed systems with MPC under changing environmental conditions is demonstrated on real test setups. Formal proofs of convergence and guaranteed improvement in cost are also given for the nonlinear and distributed MPCs.

1.3 Organization and Main contributions

This dissertation is organized as follows:

Chapter 2: Equivalence in Predictive Control Formulations

This chapter introduces a framework that brings together a number of ideas from the last thirty years in MPC and attempts to place them in a more general, modern context. The adaptive control community proposed an input-output MPC based on filtering techniques. De Keyser's EPSAC would be taken as a representative strategy from this family of online MPC controllers. The first equivalence of EPSAC stems from Clarke's GPC through the use of Diophantine equations and a corresponding 3-DOF closed-loop expression is derived. The second equivalence is constructed towards the family of state-space MPC controllers and more specifically to Bemporad et al.'s explicit solution of the optimal control problem.

Though parts of this chapter have been developed independently, they have parallels in the literature. The contribution is thus minor and limited to (i) generalizing the Diophantine based closed-form solution and (ii) deriving the special cases of the state-space representation of EPSAC.

The results presented in this chapter have been published in:
A. Dutta, R. De Keyser, C.-M. Ionescu, J. Stoev, G. Pinte, and W. Symens, "Robust predictive control design for optimal wet-clutch engagement," in *American Control Conference (ACC)*, pp. 4576–4581, IEEE, 2012

A. Dutta, C.-M. Ionescu, R. De Keyser, B. Wyns, J. Stoev, G. Pinte, and W. Symens, "Robust and two-level (nonlinear) predictive control of switched dynamical systems with unknown references for optimal wet-clutch engagement," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 228(4), pp. 233–244, 2014

A. Dutta, R. De Keyser, Y. Zhong, B. Wyns, G. Pinte, and J. Stoev, "Robust

predictive control of a wet-clutch using evolutionary algorithm optimized engagement profile," in *System Theory, Control, and Computing (ICSTCC), 2011 15th International Conference on*, pp. 1–6, IEEE, 2011

Chapter 3: Certified Predictive Control without Terminal Conditions

Set invariance is a fundamental concept in the analysis of controllers for constrained systems. The reason for this is that constraint satisfaction can be guaranteed for all time and for all disturbances if and only if the initial state is contained inside a robust control invariant set. Two ways of guaranteeing infinite time feasibility are identified: (i)'persistent feasibility' which relies only on a feasible solution to the optimization associated with the MPC and (ii)'recursive feasibility' which relies on explicit computation of a particular/optimal solution but is less conservative. Both these techniques directly lead to practical stability of the closed-loop and is demonstrated on longitudinal flight control.

The main contribution of this chapter is in adapting the existing techniques in set invariance to characterize the feasibility region in the case where (i) control horizon is shorter than prediction horizon and (ii) no terminal conditions are used. Building on this, a priori and a posteriori techniques in certifying feasibility/stability of the MPC problem have been developed.

The results presented in this chapter have been published in:
A. Dutta, E. Hartley, J. Maciejowski, and R. De Keyser, "Certification of a class of industrial predictive controllers without terminal conditions," in *Decision and Control, 53rd IEEE Conference on*, IEEE, 2014.

Chapter 4: Penalty Adaptive Predictive Control

Penalty adaptive MPC (PAMPC) is introduced as an alternative means of certifying asymptotic stability of the closed-loop. This is achieved by a mechanism of adjusting the weight of the control moves such that the constraints are rendered inactive, thereby circumventing the nonlinearity introduced by the constraints. Thus closed-form polynomial expressions for the adapted linear controller can be derived which allows us to give stability margins in the input-constrained case for nominal repetitive systems. Further, dynamic tightening of constraints through the control, prediction horizon is introduced as a means to enhance robust feasibility. Test results are given over a non-collocated mass-spring-damper setup.

The novelty here is the presentation of a tool i.e. penalty adaptation which may enable to derive the nominal closed-form solution for the adapted MPC problem for input-constrained repetitive systems, with an extension to the disturbed case.

The results presented in this chapter have been published in:
A. Dutta, M. Loccufier, C. M. Ionescu, and R. De Keyser, "Penalty adaptive model

predictive control (PAMPC) of constrained, underdamped, non-collocated mechatronic systems," in *Control Applications (CCA), 2013 IEEE International Conference on*, pp. 1006–1011, IEEE, 2013

A. Dutta, M. Loccufier, C. M. Ionescu, and R. De Keyser, "Robust penalty adaptive model predictive control (PAMPC) of constrained, underdamped, non-collocated systems," *Journal of Vibration and Control*, 2014.

Chapter 5: Switched Nonlinear Predictive Control

In practice, non-linearity is ubiquitous; De Keyser's iterative NEPSAC algorithm is adopted for the control of nonlinear systems and a formal proof of convergence is developed. Two types of switching nonlinearity are identified. In the first type, the nonlinear system can be expressed in a piecewise affine (PWA) fashion. Such a system controlled by MPC without terminal conditions can be certified by both persistent and recursive feasibility tests. This is illustrated on an example of a car navigating a terrain with time-varying slope. In the second type, the switching can be between linear and non-linear system which have different state representations. For such systems, a new architecture called two-level NMPC (2L-NMPC) in which reference adaptation is used to transit from one phase to another is introduced. The 2L-NMPC is demonstrated experimentally for wet-clutch control.

The major contributions of the chapter for nonlinear systems are: (i) a full convergence proof of nonlinear MPC i.e. NEPSAC, (ii) extension of the new feasibility tools developed in chapter 3 to PWA nonlinear systems, (iii) a two-level NMPC architecture for switching nonlinearity.

The results presented in this chapter have been published in:
A. Dutta, B. Depraetere, C. Ionescu, G. Pinte, J. Swevers, and R. De Keyser, "Comparison of two-level nmpc and ilc strategies for wet-clutch control," *Control Engineering Practice*, vol. 22, pp. 114–124, 2014

A. Dutta, C. Ionescu, B. Wyns, R. De Keyser, J. Stoev, G. Pinte, and W. Symens, "Switched nonlinear predictive control with adaptive references for engagement of wet clutches," in *Nonlinear Model Predictive Control, 4th IFAC conference on*, vol. 4, pp. 460–465, 2012

A. Dutta, C.-M. Ionescu, B. Wyns, R. De Keyser, J. Stoev, G. Pinte, and W. Symens, "Switched predictive control design for optimal wet-clutch engagement," in *IFAC Workshop on Engine and Powertrain Control, Simulation and Modeling (ECOSM-2012)*, pp. 319–324, 2012.

Chapter 6: Distributed Nonlinear Predictive Control

A pragmatic distributed nonlinear MPC without terminal conditions is developed for global mechatronic systems composed of interacting sub-systems and a monotonic decrease in the global cost (possibly non-convex) is guaranteed. Further-

more, in order to tackle time-varying process dynamics, a learning algorithm is developed, thereby improving the performance of the global control. A first principles modeling of the hydrostatic drivetrain has been developed and used for the design of distributed control of the coupled nonlinear benchmark system. Experimental validation on the benchmark suggests that the proposed control methodology is successful in practice.

The novelties of this chapter to distributed control are: (i) guaranteeing improvement in global cost with every iteration of the distributed NEPSAC algorithm, (ii) inclusion of a distributed learning mechanism for time-varying dynamics.

The results presented in this chapter have been published in:
A. Dutta, C. M. Ionescu, and R. De Keyser, "A pragmatic approach to distributed nonlinear model predictive control: Application to a hydrostatic drivetrain," *Optimal Control Applications and Methods*, 2014

A. Dutta, R. De Keyser, and I. Nopens, "Robust nonlinear extended prediction self-adaptive control (NEPSAC) of continuous bioreactors," in *Control & Automation (MED), 2012 20th Mediterranean Conference on*, pp. 658–664, 2012.

Chapter 7: Concluding Remarks and Perspectives

This chapter summarizes the contributions made by this thesis and outlines directions for future research.

Appendix A: MPC Certification Algorithm

A novel algorithm to certify a MPC controller without terminal conditions together with the computation of the associated sets is presented.

Appendix B: Model-based and Model-free Learning Control Strategies

The 2L-NMPC technology and the obtained results are compared to other model-based and model-free learning strategies for wet clutch control.

The main contribution is a detailed evaluation of model-based and model-free learning control strategies.

The results presented here have been published in:
A. Dutta, Y. Zhong, B. Depraetere, K. Van Vaerenbergh, C. Ionescu, B. Wyns, G. Pinte, A. Nowe, J. Swevers, and R. De Keyser, "Model-based and model-free learning strategies for wet clutch control," *Mechatronics*, 2014

2

Equivalence in Predictive Control Formulations

MPC is not that different from the optimal control problems studied in the 50s and 60s. However, in the classical approaches, the aim was to derive an explicit form of an optimal feedback law offline. This required a solution to the Hamilton-Jacobi-Bellman (HJB) equation, which is generally not solvable except a few cases like the celebrated Linear Quadratic Regulator (LQR) problem [24]. Because of this, most interesting optimal control problems, practically speaking, remained unsolved. The MPC approach bypasses the need to solve the HJB equation by performing an open-loop optimal control calculation online based on the current state of the process as feedback.

2.1 Introduction

In the 60s and early 70s, the idea of MPC continued to show up sporadically in the literature. Propoi [25] proposed to use linear programming to control linear systems with hard constraints. Richalet and coworkers [3] introduced a technique called Model Heuristic Predictive Control employing a finite impulse response model and at the same time Charlie Cutler was generating a lot of interest through DMC based on truncated step response model [4]. Both these methodologies saw large scale industrial implementations.

Independent from these developments in the process industry, the adaptive con-

trol community saw a rise of its own version of MPC. EPSAC [5] and GPC [6,26] amongst others naturally employed a transfer function model, like much of the work in adaptive control, and stochastic aspects played a key role from the very beginning. Because of the finite horizon, stability was not guaranteed. At the commencement of 90s, stability was successfully addressed in a series of papers [27], [28]; these papers established stability of linear, unconstrained, input-output systems by imposing terminal equality constraints on inputs and outputs over a finite interval; equivalent to imposing terminal constraint employed by [29]. Because the system is linear, cost quadratic, terminal constraint is linear equality, and control and output constraints are absent, an analytical solution can be computed. However, in the more relevant situation when output and/or input constraints are enforced, guaranteeing infinite time feasibility is as important as stability. This chapter develops the analytical tools necessary to be able to give such guarantees without using too restrictive terminal conditions and without constraining the first state prediction to be in a precomputed invariant set.

Next to these, the state-space based MPC matured through the efforts of many researchers [30], [31] and now rests on a firm theoretical foundation, with though restrictive but rigorous stability conditions [9]. The restriction comes from the addition of stabilizing terminal conditions due to which the online problem has more chances of becoming infeasible (may be due to small perturbations that put the process outside the nominal region of attraction) and in process control infeasibility is very costly. In most practical problems, measurements of state variables are not directly available and one typically employs a state estimator for output feedback [32]. An equivalent offline solution to the constrained optimization problem was developed by Bemporad et al. [33] where the control law was shown to be piecewise linear and continuous, and could be applied online by solving a point location problem and then using a lookup table to obtain an affine local state feedback law. This explicit characterization of the solution will be used as one of the building blocks in our analysis through multi-parametric quadratic programming [33].

2.2 Input-Output MPC

Some of the advantages of transfer functions over state-space, amongst others, are in the compact representation of time delays, availability of matured system identification techniques and intuitiveness. In the multivariable case too, obtaining a transfer function matrix is simpler but it must be noted that a state-space transformation in this case cay be more appropriate for representation and control. In this section we briefly describe the EPSAC-MPC formulation which is based on input-output modelling and filtering techniques. Due to its simplicity of implementation, this algorithm has been used extensively in industrial applications. This will be

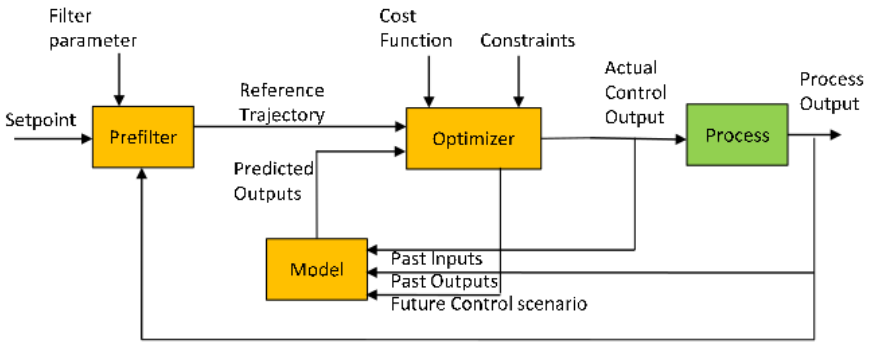


Figure 2.1: A schematic representation of the principles of model-based predictive control.

used as the baseline linear MPC strategy for real-time implementations. The aim of this chapter is then to analyze the EPSAC formulation by means of Diophantine equations (that analytically separates the response based on past and future inputs, outputs) and subsequently present an equivalent state-space representation. These equivalent realizations form the basis of the stability analysis of EPSAC in the subsequent chapters.

The process is modeled in discrete time with $y(t), \hat{y}(t), n(t)$ as process output, model output, disturbance respectively with $t \in \mathbb{Z}$ denoting time and q^{-1} the backward shift operator (i.e. acts on an element of time series to produce the previous element and can be raised to arbitrary integer powers k: $q^{-k}y(t) = y(t-k)$) as [34]:

$$y(t) = \hat{y}(t) + n(t) = \frac{q^{-d}B(q^{-1})}{A(q^{-1})}u(t) + \frac{C(q^{-1})}{D(q^{-1})}e(t) \tag{2.1}$$

where B/A represents the model dynamics with $d \geq 0$ samples delay and C/D is chosen to form the disturbance filter, with $e(t)$ as white noise with zero mean. Let the system polynomials be defined as (without loss of generality):

$$\begin{aligned} A &= 1 + a_1 q^{-1} + \ldots + a_{n_a} q^{-n_a} \\ B &= b_1 q^{-1} + \ldots + b_{n_b} q^{-n_b} \\ C &= 1 + c_1 q^{-1} + \ldots + c_{n_c} q^{-n_c} \\ D &= 1 + d_1 q^{-1} + \ldots + d_{n_d} q^{-n_d} \end{aligned} \tag{2.2}$$

The fundamental step is based on the prediction using the basic process model given by:

$$y(t+k|t) = \hat{y}(t+k|t) + n(t+k|t) \tag{2.3}$$

where $y(t+k|t)$ is the prediction of process output k steps in future computed at time t, based on prior measurements and postulated values of inputs. Prediction of model output $\hat{y}(t+k|t)$ and of colored noise process $n(t+k|t)$ can be obtained by the recursion of process model and filtering techniques respectively. The future response can then be expressed as:

$$y(t+k|t) = y_{base}(t+k|t) + y_{optimize}(t+k|t) \tag{2.4}$$

The two contributing factors have the following origins:

- $y_{base}(t+k|t)$ is the cumulative effect of past control inputs, base future control sequence $u_{base}(t+k|t)$ which is chosen a-priori and predicted disturbances.
- $y_{optimize}(t+k|t)$ is the discrete time convolution of the future control actions $\{\delta u(t|t), \ldots \delta u(t+N_u-1|t)\}$ with impulse, step response coefficients of the system, where $\delta u(t+k|t) = u(t+k|t) - u_{base}(t+k|t)$.

The design parameter N_u is the control horizon. The optimal control is then obtained by minimizing the following cost function with respect to $u(.|.)$:

$$\begin{aligned} V &= \sum_{k=N_1}^{N_2} [r(t+k|t) - y(t+k|t)]^2 + \gamma \\ subj.\ to \quad & \Delta u(t+k|t) = 0 \quad \forall k \in [N_u, N_2) \\ & u(t+k|t) \in \mathbb{U}, \ \ \forall k \in [0, N_2), \ \ y(t+k|t) \in \mathbb{Y}, \ \ \forall k \in [N_1, N_2] \end{aligned} \tag{2.5}$$

where $r(t+k|t)$ is the desired reference trajectory and control increment $\Delta u(t+k|t) = u(t+k|t) - u(t+k-1|t)$. The prediction horizon is the interval from N_1 to N_2. The second cost term γ can take any of the following formulations:

$$\gamma = \forall\{0, \lambda \sum_{k=0}^{N_u-1} [u(t+k|t)]^2, \lambda \sum_{k=0}^{N_u-1} [\delta u(t+k|t)]^2, \lambda \sum_{k=0}^{N_u-1} [\Delta u(t+k|t)]^2\} \tag{2.6}$$

with λ being the control penalty. The cost function in the multivariable case leading to distributed control is discussed in chapter 6.

The various input and output constraints can all be expressed in terms of say, $\delta U \triangleq [\delta u(t), \ldots, \delta u(t+N_u-1)]^T$, resulting in the matrices $\bar{M}, \bar{N}$ with $\bar{M}.\delta U \leq \bar{N}$. For example, if the constraint on the vector of future inputs U is $|U| \leq \mathbb{U}$, then it can be expressed as $|U_{base} + \delta U| \leq \mathbb{U}$, where U_{base} is the vector of future base control. Similarly, if the constraint on the vector of predicted outputs $\hat{Y}$ is given by $|\hat{Y}| \leq \mathbb{Y}$, then it can be expressed as $|\bar{Y} + G \cdot \delta U| \leq \mathbb{Y}$, where $G, \bar{Y}$ are the impulse/step response matrix, vector of future base response respectively. From these relations, it is straightforward to deduce the $\bar{M}, \bar{N}$ matrices.

Note that, in the case of output constraints, $\bar{N}$ is composed of the constraint value itself minus components which include base response and disturbance prediction. As a result, it leads to dynamic constraint tightening (as the disturbance estimate is computed online) and thus improved feasibility. The convex polytopic constraints lead to a convex quadratic program (QP) optimization which can be solved by an active sets based method, leading to an explicit solution:

$$\gamma_{act} = -(M_{act}.\bar{H}.M_{act}^T)^{-1}(N_{act} + M_{act}.\bar{H}^{-1}.J) \tag{2.7}$$

$$\delta U = -\bar{H}^{-1}(J + M_{act}^T.\gamma_{act}) \tag{2.8}$$

where $\bar{H}, J$ are the respective Hessian, gradient of the cost V in (2.5). The solution itself is obtained in a straightforward way by taking partial derivatives of the Lagrangian (cost+constraints) and is detailed in 2.4.2. The algorithm iteratively computes the Lagrange multipliers γ_{act} and keeps the ones which are positive, thus forming the active constraint sets M_{act}, N_{act}. The existence of $\bar{H}^{-1}$ guarantees the convergence of γ_{act} and hence optimality. In essence, i.e. the inequality constrained QP is converted to a sequence of equality constrained QPs and the purpose of the active set QP solver is to find the correct set of active constraints to minimize the cost function without violating the constraints [35]. The sketch of the iterative procedure follows:

1. Compute a feasible solution, for instance by solving the system of constraint equations, called the working set M_{act}, N_{act}.
2. Compute the Lagrange multipliers γ_{act} using (2.7)
3. Drop the constraints corresponding to the negative γ_{act} from the working set M_{act}, N_{act}.
4. Solve the new equality constrained problem for γ_{act} corresponding to the new working set using (2.7).
5. If γ_{act} is non negative, compute the optimal δU using (2.8), else go to step 2.

2.3 Diophantine MPC

Here, we derive the analytic expressions for the input-output MPC by introducing Diophantine equations. This allows us to derive polynomial expressions for the predictor dynamics and consequently the closed loop in the unconstrained case (to be used later in chapter 4 for asymptotic stability).

The k-step ahead prediction of the EPSAC process model leads to:

$$y(t+k|t) = q^{-d} \cdot \frac{B}{A} \cdot u(t+k|t) + \frac{C}{D} \cdot e(t+k|t) \tag{2.9}$$

Now, consider the first Diophantine equation [6]:

$$\frac{C}{D} = E_k + q^{-k} \cdot \frac{F_k}{D} \tag{2.10}$$

The orders of E_k, F_k are $n_e = k-1, n_f = max(n_d - 1, n_c - k)$ respectively. Now, using (2.10), the prediction of the disturbance can be written as:

$$n(t+k|t) = \frac{C}{D} \cdot e(t+k|t) = \frac{F_k}{D} \cdot e(t) + E_k \cdot e(t+k|t) \tag{2.11}$$

Note that, $E_k \cdot e(t+k|t)$ consists of only future terms and hence the best prediction possible is 0. Then, using (2.1) and (2.10), we get:

$$\frac{F_k}{D} \cdot e(t) = \frac{F_k}{C}(y(t) - q^{-d+1} \cdot \frac{B}{A} \cdot u(t-1)) \tag{2.12}$$

which can be substituted back into (2.11) to have

$$n(t+k|t) = \frac{F_k}{C}(y(t) - q^{-d+1} \cdot \frac{B}{A} \cdot u(t-1)). \tag{2.13}$$

The following Diophantine equation is introduced to split up the predictor into parts that are known at time t and future signals (with $\Delta = 1 - q^{-1}$):

$$\frac{B}{A \cdot \Delta} = G_{k-d+1} + q^{-k+d-1} \cdot \frac{H_{k-d+1}}{A \cdot \Delta} \tag{2.14}$$

The orders of G_{k-d+1}, H_{k-d+1} are $n_g = k-d-1, n_h = max(n_a, n_b - k + d - 1)$ respectively. Now, the entire predictor, using (2.13) and (2.14) can be written as:

$$\begin{aligned} y(t+k|t) =& G_{k-d+1} \cdot \Delta u(t+k-d|t) + \frac{H_{k-d+1}}{A} \cdot u(t-1) \\ &+ \frac{F_k}{C}(y(t) - q^{-d+1} \cdot \frac{B}{A} \cdot u(t-1)) \\ \equiv y(t+k|t) =& y_{optimize}(t+k|t) + y_{base}(t+k|t) \end{aligned} \tag{2.15}$$

It is important to understand that computing the prediction in this way is exactly as that in EPSAC. This is because, (i)G_{k-d+1} due to (2.14) is the step response which is convolved with future control increments to form $y_{optimize}(t+k|t)$ and (ii)the rest of the terms constitute the base response $y_{base}(t+k|t)$. Akin to EPSAC, the base response has two components: (i)second term in RHS of (2.15) is the free response of the system due to the past inputs and outputs and (ii)rest of the terms in (2.15) constitute the disturbance prediction. The disturbance prediction computed in this way by letting future white noise to 0 and filtering the current and past disturbance estimates through $1/C$ is same as that used by EPSAC filtering technique. Now, the following cost function is minimized:

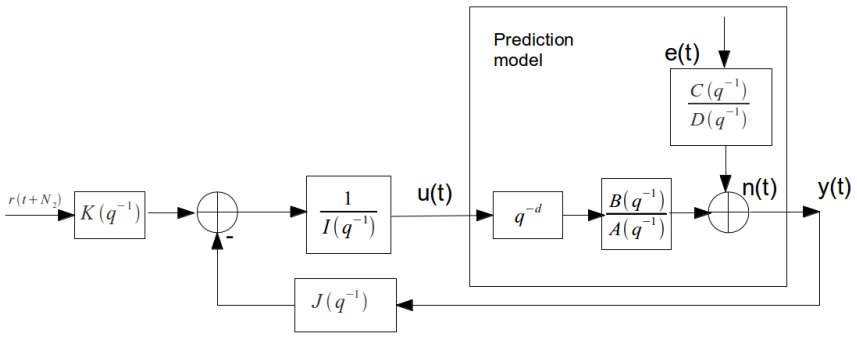


Figure 2.2: Block diagram of the closed loop formulation of EPSAC-MPC strategy. The polynomials I, J, K which have been derived in section 2.3 constitute the closed form solution: $K \cdot r(t+N_2|t) = J \cdot y(t) + I \cdot u(t|t)$.

$$\sum_{k=N_1}^{N_2} [y(t+k|t) - r(t+k|t)]^2 + \lambda[\Delta \cdot u(t+k-N_1|t)]^2$$
$$subj.\ to: \quad \Delta u(t+\tilde{k}|t) = 0,\ \forall \tilde{k} \in [N_u, N_2 - d - 1] \tag{2.16}$$

where $N_1(=d+1), N_2, N_u, \lambda, r(.|.)$ are the minimum, maximum prediction horizon, control horizon, control penalty, and reference trajectory respectively, with d the delay in samples. This gives a closed form solution similar to the original EPSAC formulation:

$$U^* = \Delta u(t|t) = \sum_{k=N_1}^{N_2} \gamma_k [r(t+k|t) - y_{base}(t+k|t)] \tag{2.17}$$

where γ_k are the elements of first row of the matrix $(G^T \cdot G + \lambda \cdot I)^{-1} G^T$, where G is the step response matrix formed by collecting the polynomials G_{k-d+1} for $k \in [1, N_2]$. Only the first value from U^* is applied to the plant. Now, substituting $y_{base}(t+k|t)$ from (2.15) in (2.17) results in:

$$A \cdot C \cdot \sum_{k=N_1}^{N_2} \cdot \gamma_k \cdot q^{-N_2+k} \cdot r(t+N_2|t) = A \cdot \sum_{k=N_1}^{N_2} \gamma_k \cdot F_k \cdot y(t)$$
$$+ (A \cdot C \cdot \Delta + C \cdot \sum_{k=N_1}^{N_2} \gamma_k \cdot q^{-1} \cdot H_{k-d+1} - B \cdot \sum_{k=N_1}^{N_2} \gamma_k \cdot q^{-d} \cdot F_k) \cdot u(t|t)$$
$$\equiv K \cdot r(t+N_2|t) = J \cdot y(t) + I \cdot u(t|t) \tag{2.18}$$

where polynomials K, J, I are the coefficients of $r(t+N_2), y(t), u(t|t)$ respectively as indicated above. Relation (2.18) represents the polynomial form of EPSAC-MPC, depicted in Fig. 2.2. The closed loop transfer function for tracking $T_{y/r}$ and

regulation $T_{y/n}$ can then be derived graphically to be:

$$T_{y/r} = \frac{q^{-d} \cdot B \cdot K}{\phi_c} \tag{2.19}$$

$$T_{y/n} = \frac{A \cdot I}{\phi_c} \tag{2.20}$$

with ϕ_c the characteristic polynomial:

$$\phi_c = A \cdot I + q^{-d} \cdot B \cdot J \tag{2.21}$$

The expression for the controller is obtained by applying conversion to unity feedback on Fig. 2.2:

$$C_{MPC} = \frac{J}{I} \tag{2.22}$$

Note that, the work by [36] also formulates the transfer functions of the unconstrained MPC starting from input-output models, but introduces two extra Diophantine equations to handle the difference between control and prediction horizons and hence the approach presented by us is far more compact and usable. One of the advantages of MPC is that multivariable processes can be handled in a straightforward manner. An input-output model for an n-output, m-input multivariable process can be expressed as:

$$\bar{AD} \cdot \bar{y}(t) = \bar{B} \cdot \bar{u}(t) + \bar{C} \cdot \bar{e}(t) \tag{2.23}$$

where $\bar{AD}, \bar{C}$ are $n \times n, n \times m$ monic polynomial matrices respectively, defined as:

$$\begin{aligned} \bar{AD} &= I_{n\times n} + \bar{ad}_1 q^{-1} + \ldots + \bar{ad}_{n_{ad}} q^{-n_a} \\ \bar{B} &= \bar{b}_1 q^{-1} + \ldots + \bar{b}_{n_b} q^{-n_b} \\ \bar{C} &= I_{n\times n} + \bar{c}_1 q^{-1} + \ldots + \bar{c}_{n_c} q^{-n_c} \end{aligned} \tag{2.24}$$

The variables $\bar{y}(t), \bar{u}(t), \bar{e}(t)$ are the $n \times 1, m \times 1, n \times 1$ output, input, noise vectors respectively (the delay can be absorbed in $\bar{B}$). The matrices $\bar{AD}, \bar{B}, \bar{C}$ can be obtained by matrix fraction description [37]. The Diophantine based analysis is analogous to the derivations made in the single in single out case except that now we get polynomial matrices instead of polynomials.

2.4 State-space MPC

In several publications, the problem of deriving a controller of the same properties based on the state-space description has been considered. Ordys et al. [38] considered one of the early approaches of comparing the state-space and polynomial

approaches to GPC control, followed by [39] where the equivalence is made only for the nominal case. We present here more generalized transformation matrices (from input-output to state-space) in the constrained case under both nominal as well as perturbed settings.

2.4.1 The Mappings

In this thesis, we consider disturbance augmented state-space description of the form [40]:

$$\begin{bmatrix} \hat{x}(t+1) \\ n(t+1) \end{bmatrix} = \begin{bmatrix} \hat{A} & \mathbf{0} \\ \mathbf{0} & \alpha \end{bmatrix} \cdot \begin{bmatrix} \hat{x}(t) \\ n(t) \end{bmatrix} + \begin{bmatrix} \hat{B} \\ \mathbf{0} \end{bmatrix} \cdot u(t) + \begin{bmatrix} \mathbf{0} \\ \beta \end{bmatrix} \cdot e(t) \tag{2.25a}$$

$$y(t) = [\hat{C} \;\; \kappa] \cdot \begin{bmatrix} \hat{x}(t) \\ n(t) \end{bmatrix} \tag{2.25b}$$

where the first state vector $\hat{x}(t)$ and the associated terms represent the dynamics of the nominal model and the vector $n(t)$ and its associated terms are used for disturbance evolution with the output $y(t)$ summing up the respective effects. A concise representation of (2.25) is given below in (2.26) with the respective terms having direct correspondence:

$$x(t+1) = \mathcal{A} \cdot x(t) + \mathcal{B} \cdot u(t) + \mathcal{E} \cdot e(t) \tag{2.26a}$$

$$y(t) = \mathcal{C} \cdot x(t) \tag{2.26b}$$

The disturbances are bounded by:

$$n(t) \in \mathbb{W} \subset \mathbb{R}^w, \;\; w(t) = \mathcal{E}.e(t) \in \mathbb{E} \subset \mathbb{R}^d. \tag{2.27}$$

The system is subject to pointwise-in-time constraints on the control input and/or the states:

$$u(t) \in \mathbb{U} \subset \mathbb{R}^m, \quad \hat{x}(t) \in \mathbb{X} \subseteq \mathbb{R}^n. \tag{2.28}$$

where w, d, m, n are the disturbance, noise, input, state dimensions respectively. The set $\mathbb{U}$ is compact (closed and bounded), while $\mathbb{X}, \mathbb{W}, \mathbb{E}$ are closed. It is assumed that the system and constraints are time invariant.

Theorem 2.4.1. *The input-output process model of* (2.1) *(after absorbing the delay in B) can be transformed to state-space form of* (2.26) *by choosing the state vector $x(t) \in \mathbb{R}^{n_a+n_b+n_d+n_c-2}$ as: $x(t) = [\hat{y}(t), \ldots \hat{y}(t-n_a+1),$ $u(t-1), \ldots u(t-n_b+1), n(t), \ldots n(t-n_d+1), e(t-1), \ldots e(t-n_c+1)]^T$.*

Proof. The time-series model of (2.1) as explained before has two additive parts i.e. the model $\hat{y}(t) = B/A \cdot u(t)$ and the disturbance $n(t) = C/D \cdot e(t)$. Therefore, the time history of all these four variables constitute the state vector and a

final addition gives the output, through the following transformation matrices:
$\mathcal{A}$ =

$$\left(\begin{array}{ccc|ccc|ccc|ccc}
-a_1 & \dots & -a_{n_a} & b_2 & \dots & b_{n_b} & 0 & \dots & 0 & 0 & \dots & 0 \\
1 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & 0 \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\
0 & \dots 1 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & 0 \\
\hline
0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & 0 \\
0 & \dots & 0 & 1 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & 0 \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\
0 & \dots & 0 & 0 & \dots 1 & 0 & 0 & \dots & 0 & 0 & \dots & 0 \\
\hline
0 & \dots & 0 & 0 & \dots & 0 & -d_1 & \dots & -d_{n_d} & c_2 & \dots & c_{n_c} \\
0 & \dots & 0 & 0 & \dots & 0 & 1 & \dots & 0 & 0 & \dots & 0 \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\
0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots 1 & 0 & 0 & \dots & 0 \\
\hline
0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & 0 \\
0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 1 & \dots & 0 \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\
0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots 1 & 0
\end{array}\right)$$

$$\mathcal{B} = \begin{bmatrix} b_1 \\ 0 \\ \dots \\ 0 \\ - \\ 1 \\ 0 \\ \dots \\ 0 \\ - \\ 0 \\ 0 \\ \dots \\ 0 \\ - \\ 0 \\ 0 \\ \dots \\ 0 \end{bmatrix} \quad \mathcal{E} = \begin{bmatrix} 0 \\ 0 \\ \dots \\ 0 \\ - \\ 0 \\ 0 \\ \dots \\ 0 \\ - \\ c_1 \\ 0 \\ \dots \\ 0 \\ - \\ 1 \\ 0 \\ \dots \\ 0 \end{bmatrix} \quad \mathcal{C} = \left[\begin{array}{ccc|ccc|ccc|ccc} 1 & 0 \dots & 0 & 0 & 0 \dots & 0 & 1 & 0 \dots & 0 & 0 & 0 \dots & 0 \end{array}\right] \tag{2.29}$$

This formulation, though non-minimal is crucial for deriving set theoretic properties starting from input-output models. It is now trivial to see that, substitution of the derived $\mathcal{A}, \mathcal{B}, \mathcal{E}, \mathcal{C}$ matrices in the state-space model of (2.26) gives exactly the same output as obtained starting from the input-output equations of (2.1). Absorption of the delay in the B polynomial means inclusion of zeros as the leading

coefficients corresponding to d samples. □

Corollary 2.4.1. *The state transformation can be extended to the multivariable case in a straightforward way by considering the associated variables and coefficients as matrices. Consider the multivariable input-output model given by (2.23), the corresponding state-space representation can be recovered by:* $x(t) = [\hat{y}(t), \ldots \hat{y}(t - n_a + 1),$
$\bar{u}(t-1), \ldots \bar{u}(t - n_b + 1), \bar{n}(t), \ldots \bar{n}(t - n_d + 1), \bar{e}(t-1), \ldots \bar{e}(t - n_c + 1)]^T$.

Proof. The model of (2.23) can be rewritten as:

$$\bar{y}(t) = \frac{\bar{B}}{\bar{A}\bar{D}} \cdot \bar{u}(t) + \frac{\bar{C}}{\bar{A}\bar{D}} \cdot \bar{e}(t) = \hat{y}(t) + \bar{n}(t) \tag{2.30}$$

Now the following transformation matrices completes the proof:
$\mathcal{A}$ =

$$\left(\begin{array}{ccc|ccc|ccc|ccc}
-\bar{ad}_1 & \ldots & -\bar{ad}_{n_{ad}} & \bar{b}_2 & \ldots & \bar{b}_{n_b} & 0 & \ldots & 0 & 0 & \ldots & 0 \\
I & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
0 & \ldots I & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 \\
\hline
0 & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 \\
0 & \ldots & 0 & I & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
0 & \ldots & 0 & 0 & \ldots I & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 \\
\hline
0 & \ldots & 0 & 0 & \ldots & 0 & -ad_1 & \ldots & -\bar{ad}_{n_{ad}} & \bar{c}_2 & \ldots & \bar{c}_{n_c} \\
0 & \ldots & 0 & 0 & \ldots & 0 & I & \ldots & 0 & 0 & \ldots & 0 \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
0 & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots I & 0 & 0 & \ldots & 0 \\
\hline
0 & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 \\
0 & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 & I & \ldots & 0 \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
0 & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 & 0 \ldots I & & 0
\end{array}\right)$$

$$\mathcal{B} = \begin{bmatrix} \bar{b}_1 \\ 0 \\ \dots \\ 0 \\ - \\ I \\ 0 \\ \dots \\ 0 \\ - \\ 0 \\ 0 \\ \dots \\ 0 \\ - \\ 0 \\ 0 \\ \dots \\ 0 \end{bmatrix} \quad \mathcal{E} = \begin{bmatrix} 0 \\ 0 \\ \dots \\ 0 \\ - \\ 0 \\ 0 \\ \dots \\ 0 \\ - \\ \bar{c}_1 \\ 0 \\ \dots \\ 0 \\ - \\ I \\ 0 \\ \dots \\ 0 \end{bmatrix} \quad \mathcal{C} = \begin{bmatrix} I & 0 \dots & 0 & | 0 & 0 \dots & 0 & | I & 0 \dots & 0 & | 0 & 0 \dots & 0 \end{bmatrix} \tag{2.31}$$

□

Lemma 2.4.1. *If the disturbance is represented as an auto-regressive model i.e. with $C = 1$, the state-space system is reduced to:*

$$x(t+1) = \mathcal{A} \cdot x(t) + \mathcal{B} \cdot u(t) \tag{2.32a}$$
$$y(t) = \mathcal{C} \cdot x(t) \tag{2.32b}$$

with the state vector $x(t) \in \mathbb{R}^{n_a+n_b+n_d-1}$ as:
$x(t) = [\hat{y}(t), \dots \hat{y}(t-n_a+1), u(t-1), \dots u(t-n_b+1), n(t), \dots n(t-n_d+1)]^T$.

Proof. The disturbance prediction in this case is of the form
$n(t+1|t) = d_1 \cdot n(t) + \ldots + e(t+1|t)$, but the best prediction of the zero mean stochastic noise is $e(t+1|t) = 0$ and thus the disturbance prediction only depends on its past values, which concludes the proof. The state transformation matrices in this case take the following form:

$$\mathcal{A} = \left(\begin{array}{ccc|ccc|ccc} -a_1 & \dots & -a_{n_a} & b_2 & \dots & b_{n_b} & 0 & \dots & 0 \\ 1 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots 1 & 0 & 0 & \dots & 0 & 0 & \dots & 0 \\ \hline 0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \dots & 0 & 1 & \dots & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & 0 & \dots 1 & 0 & 0 & \dots & 0 \\ \hline 0 & \dots & 0 & 0 & \dots & 0 & -d_1 & \dots & -d_{n_d} \\ 0 & \dots & 0 & 0 & \dots & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots 1 & 0 \end{array}\right)$$

$$\mathcal{B} = \begin{bmatrix} b_1 \\ 0 \\ \dots \\ 0 \\ - \\ 1 \\ 0 \\ \dots \\ 0 \\ - \\ 0 \\ 0 \\ \dots \\ 0 \end{bmatrix} \quad \mathcal{C} = \begin{bmatrix} 1 & 0 \dots & 0 & | 0 & 0 \dots & 0 & | 1 & 0 \dots & 0 \end{bmatrix} \tag{2.33}$$

The result is significant, given that most practical cases can be represented in this way (e.g. constant, ramp, sinusoid disturbances). □

Corollary 2.4.2. *In the nominal case i.e without disturbance, the state-vector* $x(t) \in \mathbb{R}^{n_a+n_b+n_d-1}$ *is given by:*
$x(t) = [y(t), \dots y(t-n_a+1), u(t-1), \dots u(t-n_b+1)]^T$ *and the following transformation matrices can be derived:*

$$\mathcal{A} = \left(\begin{array}{ccc|ccc} -a_1 & \dots & -a_{n_a} & b_2 & \dots & b_{n_b} \\ 1 & \dots & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots 1 & 0 & 0 & \dots & 0 \\ \hline 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \dots & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & 0 & \dots 1 & 0 \end{array}\right)$$

$$\mathcal{B} = \begin{bmatrix} b_1 \\ 0 \\ \dots \\ 0 \\ - \\ 1 \\ 0 \\ \dots \\ 0 \end{bmatrix} \quad \mathcal{C} = \begin{bmatrix} 1 & 0 \dots & 0 & | 0 & 0 \dots & 0 \end{bmatrix} \tag{2.34}$$

Lemma 2.4.2. *The constraints from the input-output model i.e* $\hat{y}(t) \in \mathbb{Y}, u(t) \in \mathbb{U}, n(t) \in \mathbb{W}, e(t) \in \mathbb{E}$ *are mapped to the constraints on the state as follows:* $x(t) \in \mathbb{X} = [\mathbb{Y} \times \dots \times \mathbb{Y} \times \mathbb{U} \times \dots \times \mathbb{U} \times \mathbb{W} \times \dots \times \mathbb{W} \times \mathbb{E} \times \dots \times \mathbb{E}]^T$.

Proof. Directly follows from the transformed state-vector representation in theorem 2.4.1. □

Corollary 2.4.3. *In the case with* $C = 1$*, the time-series constraints are mapped to:*
$x(t) \in \mathbb{X} = [\mathbb{Y} \times \dots \times \mathbb{Y} \times \mathbb{U} \times \dots \times \mathbb{U} \times \mathbb{W} \times \dots \times \mathbb{W}]^T$.

Corollary 2.4.4. *In the nominal case, the input-output constraints are mapped to:* $x(t) \in \mathbb{X} = [\mathbb{Y} \times \dots \times \mathbb{Y} \times \mathbb{U} \times \dots \times \mathbb{U}]^T$ *through the corresponding state-vector transformation.*

Remark 2.4.1. *Note that, the obtained state-space representations are non-minimal and indeed may have higher order due to the inclusion of the memory involving the inputs and disturbances. However, there exists a strong reason that motivates this choice, which in principle has to do with lemma 2.4.1. This is in fact a compact way of representing disturbances and dealing with them by including them as a part of the state that makes redundant any additional terms for disturbance dynamics (which are needed in the case of canonical state-space forms). In doing so, nominal set theoretic tools can now be applied directly to the disturbed case, which will be the way forward in chapter 3. The non-minimal representation also gives an exact correspondence to the way input-output models are formulated (i.e. keeping record of past values of variables).*

Theorem 2.4.2. *The finite horizon constrained optimal control problem in the input-output formulation of (2.5) translates exactly to the minimization over* $u(.|.)$ *of the following cost in the state-space framework:*

$$\begin{aligned} & V = \sum_{k=N_1}^{N_2} [r(t+k|t) - \mathcal{C}.x(t+k|t)]^2 + \gamma \\ subj.\ to \quad & \Delta u(t+k|t) = 0 \quad \forall k \in [N_u, N_2) \\ & u(t+k|t) \in \mathbb{U}, \ \ \forall k \in [0, N_2), \ \ x(t+k|t) \in \mathbb{X}, \ \ \forall k \in [N_1, N_2] \end{aligned} \tag{2.35}$$

where all the terms have exactly the same meaning as before and γ is defined in (2.6). The predictions are made by using the state-space maps of (2.26). Note that, the formulation is also referred to as independent model with augmented disturbance dynamics which has similar interpretation as realigned model for open-loop stable linear systems.

Proof. Since the state-space model of (2.26) has been proven to be equivalent to the input-output model of (2.1) in theorem 2.4.1, the cost function of (2.35) exactly matches that of (2.6) through the transformation vector $\mathcal{C}$. Further, the state constraints in (2.35) are obtained from the input-output constraints by using lemma 2.4.2. □

By substituting

$$x(t+k|t) = \mathcal{A}^k \cdot x(t) + \sum_{i=1}^{k-1} \mathcal{A}^i \cdot \mathcal{B} \cdot u(k-1-i|t) \tag{2.36}$$

into the cost function of problem (2.35), the optimization can be rewritten as

$$\begin{aligned} U^*_\Phi(x(t)) = \arg\min_U \ & \frac{1}{2} U^T \cdot \Phi \cdot U + U^T \cdot \Omega \cdot x(t) + x(t)^T \cdot \Psi \cdot x(t) \\ subject\ \ to \quad & \Xi \cdot U \preceq f + \Pi \cdot x(t) \end{aligned} \tag{2.37}$$

where $U \triangleq [u(t), \ldots, u(t+N_u-1)]^T$. The matrices and vectors $\Psi, \Omega, \Xi, \Pi, f$ and $\Phi \succ 0$ are obtained by collecting terms [7]. The term involving Ψ is usually dropped, since it does not affect the optimal solution $U^*_\Phi(x(t))$.

2.4.2 Explicit solution

Note that both the cost function and the constraints, and hence the optimal solution (and hence the corresponding Lagrange multiplier μ^*), are dependent on $x(t)$. The MPC problem can therefore be treated as an multi-parametric Quadratic Program (mp-QP) for which an explicit solution can be computed offline [33]. The feasible set X_F is the set of states $x(t)$ for which a feasible control sequence U to the MPC problem (2.37) exists.

$$X_F \triangleq \{x(t) \in \mathbb{R}^n | \exists U : \Xi \cdot U \preceq f + \Pi \cdot x(t)\}. \tag{2.38}$$

The Kahrush-Kuhn-Tucker (KKT) conditions provide some insight into the relation of the Lagrange multipliers to $x(t)$. The Lagrangian of the optimization problem (2.37) is

$$\begin{aligned} \mathcal{L}(U, \mu, x(t)) = \frac{1}{2} U^T \cdot \Phi \cdot U + U^T \cdot \Omega \cdot x(t) + x(t)^T \cdot \Psi \cdot x(t) + \\ \mu^T (\Xi \cdot U - f - \Pi \cdot x(t)) \end{aligned} \tag{2.39}$$

A stationary point for the Lagrangian occurs when $\bigtriangledown_U \mathcal{L}(U, \mu, x(t)) = 0$, hence the corresponding KKT optimality conditions are [41]

$$\Phi \cdot U + \Omega \cdot x(t) + \Xi^T \cdot \mu = 0 \tag{2.40a}$$
$$\mu \succeq 0, \mu \in \mathbb{R}^q \tag{2.40b}$$
$$\Xi \cdot U - f - \Omega \cdot x(t) \preceq 0 \tag{2.40c}$$
$$\text{diag}(\mu)(\Xi \cdot U - f - \Pi \cdot x(t)) = 0 \tag{2.40d}$$

where q is the number of non-redundant linear inequalities in (2.37). Provided $\Phi \succ 0$ (as is the case when control penalty matrix $\lambda \cdot I \succ 0$, from (2.40a) one can solve for the unique

$$U = -\Phi^{-1}(\Omega \cdot x(t) + \Xi^T \cdot \mu) \tag{2.41}$$

and check that U satisfies (2.40) . For a given $x(t)$, the U and μ which solve (2.40) are equal to the solution $U^*_\Phi(x(t))$ and Lagrange multipliers μ^* of (2.37).

Theorem 2.4.3. *[33] For a given $x(t)$, let $\breve{\mu}(x(t)) = 0$ and $\tilde{\mu}(x(t))$ denote the Lagrange multipliers corresponding to the inactive and active constraints at the optimal solution, respectively. The Lagrange multipliers corresponding to the active constraints are given by*

$$\tilde{\mu}(x(t)) = S \cdot x(t) + v \tag{2.42}$$

and the optimal solution is given by

$$U^*_\Phi(x(t)) = (-\Phi^{-1} \cdot \Omega - \Phi^{-1} \cdot \tilde{\Xi}^T \cdot S)x(t) - \Phi^{-1} \cdot \tilde{\Xi}^T \cdot v \tag{2.43}$$

where

$$S = -(\tilde{\Xi} \cdot \Phi^{-1} \cdot \tilde{\Xi}^T)^{-1}(\tilde{\Pi} + \tilde{\Xi} \cdot \Phi^{-1} \cdot \Omega) \tag{2.44a}$$
$$v = -(\tilde{\Xi} \cdot \Phi^{-1} \cdot \tilde{\Xi}^T)^{-1}\tilde{f} \tag{2.44b}$$

and $\tilde{\Xi}, \tilde{f}, \tilde{\Pi}$ correspond to the set of active constraints. Furthermore, these expressions are valid for all $x(t)$ contained in the polyhedron

$$CR = \{x(t) \in \mathbb{R}^n | \begin{pmatrix} -\Xi \cdot \Phi^{-1} \cdot \Omega - \Xi \cdot \Phi^{-1} \cdot \tilde{\Xi}^T \cdot S - \Pi \\ -S \end{pmatrix} x(t) \\ \preceq \begin{pmatrix} f + \Xi \cdot \Phi^{-1} \cdot \tilde{\Xi}^T \cdot v \\ v \end{pmatrix}\} \tag{2.45}$$

Proof. Substitute (2.41) into (2.40d) to obtain the complementary slackness condition

$$\text{diag}(\mu)(\Xi \cdot (-\Phi^{-1}(\Omega \cdot x(t) + \Xi^T \cdot \mu)) - f - \Pi \cdot x(t)) = 0 \tag{2.46}$$

For the inactive constraints $\breve{\mu}(x(t)) = 0$.
Let the rows of $\tilde{\Xi}, \tilde{f}, \tilde{\Pi}$ correspond to the set of active constraints. For the active constraints $\tilde{\mu} \succ 0$ and hence (2.40d) implies that

$$\tilde{\Xi}(-\Phi^{-1}(\Omega \cdot x(t) + \tilde{\Xi}^T \cdot \tilde{\mu})) - \tilde{f} - \tilde{\Pi} \cdot x(t)) = 0 \tag{2.47}$$

and solving for $\tilde{\mu}$ it follows that

$$\tilde{\mu}(x(t)) = -(\tilde{\Xi} \cdot \Phi^{-1} \cdot \tilde{\Xi}^T)^{-1}(\tilde{f} + (\tilde{\Pi} + \tilde{\Xi} \cdot \Phi^{-1} \cdot \Omega)x(t)) \tag{2.48}$$

The inverse exists because the rows of $\tilde{\Xi}$ are assumed to be linearly independent (valid in most practical cases). By defining S and v as in (2.44), the expression $\tilde{\mu}(x(t)) = S \cdot x(t) + v$ results.
Substituting the expression for $\tilde{\mu}(x(t))$ into (2.41) one gets

$$\begin{aligned} U^*_\Phi(x(t)) = -\Phi^{-1}(\Omega \cdot x(t) + \tilde{\Xi}^T \cdot (S \cdot x(t) + v)) = \\ (-\Phi^{-1} \cdot \Omega - \Phi^{-1} \cdot \tilde{\Xi}^T \cdot S)x(t) - \Phi^{-1} \cdot \tilde{\Xi}^T \cdot v \end{aligned} \tag{2.49}$$

$U^*_\Phi(x(t))$ has to satisfy the constraints (2.37) and the Lagrange multipliers $\tilde{\mu}(x(t))$ corresponding to the active constraints have to be non-negative. These two constraints combine to define the critical region

$$\begin{aligned} CR &= \{x(t) \in \mathbb{R}^n | \Xi(-\Phi^{-1} \cdot \Omega - \Phi^{-1} \cdot \tilde{\Xi}^T \cdot S) \preceq f + \Pi \cdot x(t), S \cdot x(t) + v \succeq 0\} \\ &= \{x(t) \in \mathbb{R}^n | \begin{pmatrix} -\Xi \cdot \Phi^{-1} \cdot \Omega - \Xi \cdot \Phi^{-1} \cdot \tilde{\Xi}^T \cdot S - \Pi \\ -S \end{pmatrix} x(t) \\ &\qquad \preceq \begin{pmatrix} f + \Xi \cdot \Phi^{-1} \cdot \tilde{\Xi}^T \cdot v \\ v \end{pmatrix}\} \end{aligned} \tag{2.50}$$

□

The result implies that the resulting MPC control law is a continuous, piecewise-affine function with domain X_F. The following algorithm is an effective procedure to determine the complete expression over all of X_F:

1. Set $i \leftarrow 0$.
2. Choose an arbitrary $x(t) \in X_F$.
3. Solve the corresponding QP.
4. By looking at the constraints which are active at the solution of this QP, compute the affine functions for $U^*_\Phi(x(t))$ and $\mu^*(x(t))$ as in theorem 2.4.3.
5. Compute the resulting critical region CR_i and remove the redundant constraints.

6. Terminate if $\cup_{j=0}^{i} CR_j = X_F$, else set $i \leftarrow i+1$ and continue.

7. Choose an arbitrary $x(t) \in X_F$ but not in $\cup_{j=0}^{i-1} CR_j$ (an obvious procedure would be to explore the neighbouring critical regions) and go to step 3.

The procedure guarantees that all feasible combinations of active constraints will be computed (efficiency depends on procedure for choosing $x(t)$ in step 7 [33]).

2.5 Summary

This chapter brought together a number of ways of defining, analyzing and solving the predictive control problem. An input-output MPC formulation i.e. EPSAC and its algorithm has been described. Next, Diophantine equations based analysis is performed to derive the analytical equivalence to the input-output MPC, in the unconstrained case.

A state-space equivalence to EPSAC is then derived through some transformation techniques, which are valid for nominal as well as perturbed case, together with mapping of the constraint sets. A solution to the constrained optimal problem is then derived as a piecewise affine control law, affine in the state variable by making use of the Lagrangian function. Thanks to the state transformation, this solution is exactly the same as that to the original input-output MPC problem.

The transformations described in this chapter are used for proving stability in the following chapters, therefore the novelty in this chapter itself is marginal but can still be stated as the following: (i) the Diophantine based analysis is carried out for a very general disturbance model C/D as opposed to $C/(A \cdot \Delta)$ which is vastly considered in the literature (ii) a lemma which explicitly gives the input-output to state-space mappings for constrained systems with frequently encountered disturbance dynamics $1/D$.

3

Certified Predictive Control without Terminal Conditions

The vast majority of research in stabilizing MPC and guaranteeing infinite time feasibility under constraints has invariably enforced one or all of the three ingredients: terminal penalty, terminal constraints, terminal control law [9]. The desirable features of this approach are [42]:

1. Nominal stability follows easily from the properties of stabilizing ingredients.
2. The problem is infinite time feasible (i.e. a solution exists that satisfies the constraints every time).

The objections raised to this method of stabilization include:

1. The stabilizing ingredients may be difficult to compute.
2. Adding a fictitious terms may compromise performance.
3. The region where the problem is feasible may shrink.
4. Most of the stabilizing ingredients are not used in the process industry.

The last point is not an objection per say, but the industry does not use them as stabilization happens anyway. Mostly it is because, engineers do not want to compromise feasibility (at the price of having to switch to hard safety mechanisms). It

is not always wanted nor easy to construct an MPC controller which has a-priori guarantee of infinite feasibility and stability, either due to theoretical complications, or pragmatic decisions in practice [43]. Instead, we might have a situation where we are given an MPC controller, and the goal is to deduce a region where the problem is infinite time feasible/stable. This is common for industry where the criticality of the process inhibits one from stopping and re-designing the controller, in which case providing the knowledge about a safe region of operation could be indispensable. The principal contribution of the thesis and in particular this chapter is to deduce such a region of inputs and outputs where the industrial controllers without stabilizing ingredients are certifiable with respect to stability through infinite feasibility. Further new tuning guidelines are given which can give a-priori guarantee on closed-loop feasibility/stability.

3.1 Introduction

The research literature on MPC has adopted the following formulation as standard (which is a regulation problem in the nominal case) [44]:

Problem 3.1.1.

$$V^*(x(t)) = \min_{u(.|.)} \; T(x(t+N_2|t)) + \sum_{k=0}^{N_2-1} L(x(t+k|t), u(t+k|t)) \tag{3.1}$$
$$subj.\ to \;\; u(t+k|t) \in \mathbb{U}, \;\; x(t+k|t) \in \mathbb{X}, \; x(t+N_2|t) \in \mathbb{T}, \;\; \forall k \in [0, N_2)$$

under the state-space model $x(t+1) = f(x(t), u(t))$ together with the assumption that all the sets contain origin within their interior.

Theorem 3.1.1. *The MPC problem 3.1.1 is infinite time feasible and the origin is asymptotically stable fixed point if the stage cost $L(.)$ and terminal cost $T(.)$ are positive definite; $T(.)$ is a control Lyapunov function (CLF) for the closed-loop system under a terminal control law $h(x(t+k|t)), \forall k \geq N_2$ that ensures a minimum cost decrement by $-L(.)$; terminal constraint set $\mathbb{T}$ is invariant under $h(.)$ (which is admissible with respect to the state and input constraints) and the optimization is initially feasible.*

Proof. Consider the optimal input sequence: $U^* = \{u^*(t), \ldots, u^*(t+N_2-1)\}$. At the next time step, the shifted input sequence will have a tail under the terminal control law: $\tilde{U} = \{u^*(t+1), \ldots, h(x(t+N_2|t))\}$. Now the cost associated with the shifted sequence (after applying $u^*(t)$) can be written down as follows:

$$V(f(x(t), U^*(x(t))) = V^*(x(t)) - L(x(t), u^*(t)) - T(x(t+N_2|t)) + L(x(t+N_2|t), h(x(t+N_2|t))) + T(f(x(t+N_2|t), h(x(t+N_2|t)))) \tag{3.2}$$

For stability, the last three terms must be negative, which is in fact ensured by choosing the terminal cost $T(.)$ as a CLF under the terminal control law $h(.)$, giving:

$$T(f(x(t),h(x(t))) - T(x(t)) \leq -L(x(t),h(x(t))), \forall x(t) \in \mathbb{T} \tag{3.3}$$

From this condition, it follows (with an optimally computed sequence U^*):

$$V^*(f(x(t),U^*(x(t))) - V^*(x(t)) \leq -L(x(t),u^*(t)) \tag{3.4}$$

and therefore the optimal cost is a Lyapunov function for the closed-loop system, and convergence to the origin is guaranteed.
Next, since $\mathbb{T}$ is chosen to be invariant under constraint admissible $h(.)$, given an initial feasible input trajectory U at time t, the shifted input sequence $\tilde{U}$ is bound to be feasible at time $t+1$, after implementing $u^*(t)$. This concludes the proof of infinite time feasibility.

Note that the terminal control law is actually never used to control the system, it is only used as a proof argument and to compute the terminal cost and set. □

However in the industrial MPC formulation, none of the three mentioned ingredients are considered in the design phase. For the sake of clarity, we once again state the industrial MPC regulation problem in the nominal case:

Problem 3.1.2.

$$V^*(x(t)) = \min_{u(.|.)} \sum_{k=N_1}^{N_2} (\mathcal{C} \cdot x(t+k|t))^2 + \lambda \sum_{k=0}^{N_u-1} (u(t+k|t))^2 \tag{3.5}$$
$$subj.\ to\ \ u(t+k|t) \in \mathbb{U},\ \ \forall k \in [0, N_2),\ \ x(t+k|t) \in \mathbb{X},\ \ \forall k \in [N_1, N_2]$$

under the state-space model $x(t+1) = \mathcal{A} \cdot x(t) + \mathcal{B} \cdot u(t)$ and $y(t) = \mathcal{C} \cdot x(t)$, where $\mathcal{A}, \mathcal{B}, \mathcal{C}$ are the transformation matrices derived in chapter 2. Note that, the extension to the multivariable case is straightforward by considering another summation over all the outputs and/or inputs; a more elegant way of formulating the problem in a distributed sense is dealt with in chapter 6. One can immediately write down the differences between the academic MPC of problem 3.1.1 to the industrial MPC of problem 3.1.2 as follows:

D1 The predictions are made over input-output models together with disturbance filter.

D2 The control and prediction horizons are not equal.

D3 There is no terminal constraint $\mathbb{T}$.

D4 There is no terminal cost $T(.)$.

D5 There is no terminal control law $h(.)$.

In this case neither stability can be guaranteed (as the optimal cost is not explicitly constructed to be a Lyapunov function for the closed-loop), nor that the problem remains infinite time feasible (due to lack of invariance). Some authors have indeed considered omission of terminal constraint [D3], in which the control horizon (=prediction horizon) is made sufficiently large to ensure that the terminal constraint is automatically satisfied [45, 46]. A recent book [47] deals extensively in solving [D3-5] via the choice of an appropriate horizon employing the assumption that the system is asymptotically controllable through the cost function. Finally, some tools for showing infinite-feasibility have been developed in [48, 49] using invariance and a means of detection of initial conditions that lead to infeasibility of a MPC controller with [D3-5] has been proposed in [43] by using bi-level programming.

To our knowledge, there has been no prior-work in dealing with all the five mentioned differences [D1-5] together and delivering to the industry a region of attraction for which the controller (either optimal or not) is certified infinite time feasible/stable. Thus, we build upon the necessary set-theoretic tools in the next section to be able to develop a procedure to certify stability through infinite time feasibility. A feature that distinguishes our method is that it does not need convergence of the associated sets (which is hard to obtain in all cases).

3.2 Invariant Set Theory

Invariant set theory has been shown to be crucial in understanding the behavior of constrained systems, since constraints can be satisfied if and only if the initial state is bounded in a set which is invariant (i.e. trajectories do not exit this set).
Notation: To distinguish robust sets, i.e. the ones which consider effect of disturbances, a 'tilde' will be appended. The symbols $\oplus, \ominus$ denote the Minkowski sum and Pontryagin difference respectively, which are analogous operations to $+, -$ but over sets. The set $A \setminus B$ is the complement of B contained in A. The notation $A \subset B, A \subseteq B$ is used to denote A is a proper subset, subset of B respectively. The symbols $\exists, \in, |, :$ mean there exists, belongs to, constrained to, such that respectively.
Consider once again the following discrete-time system:

$$x(t+1) = f(x(t), u(t), n(t)) \tag{3.6a}$$
$$y(t) = g(x(t), n(t)) \tag{3.6b}$$
$$u(t) \in \mathbb{U}, \quad x(t) \in \mathbb{X} \tag{3.6c}$$
$$n(t) \in \mathbb{W} \tag{3.6d}$$

Note that, in the case of non-minimal state-space derived from input-output representation, $n(t)$ is implicitly a part of the state vector $x(t) = [\hat{y}(t), \dots u(t-1), \dots n(t), \dots]$, not necessarily otherwise. As usual, the set $\mathbb{U}$ is compact, while $\mathbb{X}, \mathbb{W}$ are closed. In the nominal case the map reduces to $x(t+1) = f(x(t), u(t))$. An admissible control input sequence or law is one that satisfies the input constraints.

3.2.1 Invariant sets

Definition 3.2.1. *[50] The set $\mathbb{X} \subset \mathbb{R}^n$ is **robust positively invariant** for the autonomous system $x(t+1) = f(x(t), n(t))$ iff $\forall x(t) \in \mathbb{X}, \forall n(t) \in \mathbb{W}$, the system evolution satisfies $x(t+k) \in \mathbb{X}, \forall k$.*

Definition 3.2.2. *[51] The set $\tilde{O}_\infty(\mathbb{X})$ is the **maximal robust positively invariant** set contained in $\mathbb{X}$ for the autonomous system $x(t+1) = f(x(t), n(t))$ iff $\tilde{O}_\infty(\mathbb{X})$ is robust positively invariant and contains all robust positively invariant sets contained in $\mathbb{X}$.*

A closed loop system is also viewed as an autonomous system with $x(t+1) = f(x(t), h(x(t)), n(t))$ evolving under the control law $h(.)$.

Definition 3.2.3. *[50] The set $\mathbb{X} \subset \mathbb{R}^n$ is **robust control invariant** for the system $x(t+1) = f(x(t), u(t), n(t))$ iff there exists a feedback control law $u(t) = h(x(t))$ such that $\mathbb{X}$, is robust positively invariant set for the closed-loop system $x(t+1) = f(x(t), h(x(t)), n(t))$ and $u(t) \in \mathbb{U}, \forall x(t) \in \mathbb{X}$.*

Definition 3.2.4. *[50] The set $\tilde{C}_\infty(\mathbb{X})$ is the **maximal robust control invariant** set contained in $\mathbb{X}$ for the system $x(t+1) = f(x(t), n(t))$ iff $\tilde{C}_\infty(\mathbb{X})$ is robust control invariant and contains all robust control invariant sets contained in $\mathbb{X}$.*

Theorem 3.2.1. *Given the time-invariant system of (3.6a), the constraints (3.6c) can be satisfied for all time iff the initial state $x(t) \in \tilde{C}_\infty(\mathbb{X})$.*

Remark 3.2.1. *A similar condition holds for autonomous systems and the corresponding maximal robust positively invariant set.*

Definition 3.2.5. *[52] The **robust pre set** $\tilde{Q}(\mathbb{X})$ is defined as the set of states in $\mathbb{R}^n$ for which an admissible control input exists which will drive the system to $\mathbb{X}$ in one step, for all allowable disturbances i.e.*

$$\tilde{Q}(\mathbb{X}) \triangleq \{x(t) \in \mathbb{R}^n | \exists u(t) \in \mathbb{U} : f(x(t), u(t), n(t)) \in \mathbb{X}, \forall n(t) \in \mathbb{W}\} \quad (3.7)$$

Theorem 3.2.2. *[52] Geometric condition for invariance: The set $\mathbb{X}$ is robust control/positive invariant set iff $\mathbb{X} \subseteq \tilde{Q}(\mathbb{X})$ i.e. $\tilde{Q}(\mathbb{X}) \cap \mathbb{X} = \mathbb{X}$.*

Definition 3.2.6. *The **robust output admissible set** $\tilde{X}^g$ is the set of states for which the output constraints are satisfied for all allowable disturbances, i.e.*

$$\tilde{X}^g \triangleq \{x(t) \in \mathbb{X} | y(t) \in \mathbb{Y}, \forall n(t) \in \mathbb{W}\} \tag{3.8}$$

Definition 3.2.7. *The **robust reach set** $\tilde{R}(\mathbb{X})$ is the set of states to which the system will evolve at the next time step given any $x(t)$, admissible control input and allowable disturbance, i.e.*

$$\begin{aligned}\tilde{R}(\mathbb{X}) \triangleq \{x(t+1) \in \mathbb{R}^n | \exists x(t) \in \mathbb{X}, u(t) \in \mathbb{U}, n(t) \in \mathbb{W} : \\ x(t+1) = f(x(t), u(t), n(t))\}\end{aligned} \tag{3.9}$$

At this point, the following couple of definitions on stability of dynamical systems would be handy.

Definition 3.2.8. *An equilibrium point x_e is **practical Lyapunov stable** if all the solutions to the corresponding dynamical system that start out near x_e stay near x_e forever.*

Definition 3.2.9. *An equilibrium point x_e is **asymptotically stable** if all the solutions to the corresponding dynamical system that start out near x_e converge to x_e.*

3.2.2 Computation of invariant sets

The robust pre set $\tilde{Q}(\mathbb{X})$ is the orthogonal projection of the set $\{(x(t), u(t)) \in \mathbb{R}^n \times \mathbb{R}^m : f(x(t), u(t), n(t)) \in \mathbb{X}, u(t) \in \mathbb{U}, n(t) \in \mathbb{W}\}$ onto the first coordinate (i.e. $x(t)$).

Definition 3.2.10. *The $i-step$ **robust controllable set** $\tilde{K}_i(\mathbb{X}, \mathbb{T})$ is the set of states in $\mathbb{X}$ which can be driven by an admissible input sequence of length i to an arbitrary target set $\mathbb{T}$ in exactly i steps, while keeping the evolution of the state inside $\mathbb{X}$ for the first $i-1$ steps, for all allowable disturbances i.e.*

$$\begin{aligned}\tilde{K}_i(\mathbb{X}, \mathbb{T}) \triangleq \{x(t) \in \mathbb{R}^n | \exists \{u(t+k) \in \mathbb{U}\}_{k=0}^{i-1} : \{x(t+k) \in \mathbb{X}\}_{k=1}^{i-1} \in \mathbb{X}, \\ \{n(t+k) \in \mathbb{W}\}_{k=0}^{i-1}, x(t+i) \in \mathbb{T}\}\end{aligned} \tag{3.10}$$

where $\tilde{K}_\infty(\mathbb{X}, \mathbb{T})$ is the robust infinite-time controllable set with determinedness index $i_K^* = i$ for which (3.11c) is satisfied.

Definition 3.2.11. *The $i-step$ **robust admissible set** $\tilde{C}_i(\mathbb{X})$ contained in $\mathbb{X}$ is the set of states for which an admissible control sequence of length i exists, while keeping the evolution of the state inside $\mathbb{X}$ for i steps, for all allowable disturbances i.e.*

$$\begin{aligned}\tilde{C}_i(\mathbb{X}) \triangleq \{x(t) \in \mathbb{R}^n | \exists \{u(t+k) \in \mathbb{U}\}_{k=0}^{i-1} : \{x(t+k) \in \mathbb{X}\}_{k=1}^{i} \in \mathbb{X}, \\ \forall \{n(t+k) \in \mathbb{W}\}_{k=0}^{i-1}\}\end{aligned} \tag{3.12}$$

The robust controllable sets of a system can be computed via the following iterative procedure:

$$\tilde{K}_0(\mathbb{X}, \mathbb{T}) = \mathbb{T} \tag{3.11a}$$

$$\tilde{K}_{i+1}(\mathbb{X}, \mathbb{T}) = \tilde{Q}(\tilde{K}_i(\mathbb{X}, \mathbb{T})) \cap \mathbb{X} \tag{3.11b}$$

$$If \;\; \tilde{K}_i(\mathbb{X}, \mathbb{T}) = \tilde{K}_{i+1}(\mathbb{X}, \mathbb{T}), \;\; then \;\; \tilde{K}_\infty(\mathbb{X}, \mathbb{T}) = \tilde{K}_i(\mathbb{X}, \mathbb{T}), \;\; terminate. \tag{3.11c}$$

algorithm 1: Robust controllable sets

The maximal robust control invariant set $\tilde{C}_\infty(\mathbb{X})$ can be computed using algorithm 1 by noting that:

$$\tilde{C}_i(\mathbb{X}) = \tilde{K}_i(\mathbb{X}, \mathbb{X}) \tag{3.13a}$$

$$If \;\; \tilde{C}_i(\mathbb{X}) = \tilde{C}_{i+1}(\mathbb{X}) \;\; then \;\; \tilde{C}_\infty(\mathbb{X}) = \tilde{C}_i(\mathbb{X}), \;\; terminate. \tag{3.13b}$$

algorithm 2: Maximal robust control invariant set

If the condition (3.13b) holds for a finite i_C^*, then $\tilde{C}_\infty(\mathbb{X})$ is finitely determined and i_C^* is called the determinedness index. If the input, state constraints are compact and the state update is continuous, then the convergence is guaranteed [53].

Now, we introduce a new set tailored towards solving problems with control horizons shorter then prediction horizons.

Definition 3.2.12. *An $i-step$* ***robust tunnel set*** *$\tilde{L}_i(\mathbb{X})$ is an i-step robust admissible set $\tilde{C}_i(\mathbb{X})$ subject to the constraint that the admissible control sequence remains constant i.e.*

$$\tilde{L}_i(\mathbb{X}) \triangleq \{\tilde{C}_i(\mathbb{X}) | \{\Delta u(t+k) = 0\}_{k=1}^{i-1}\} \tag{3.14}$$

Lemma 3.2.1. *The i-step robust tunnel set $\tilde{L}_i(\mathbb{X})$ can be derived by computing the orthogonal projection of the $i-$step robust positively invariant set (i.e. robust positively invariant for i steps) for the system with augmented state vector $[x(t), u(t)]^T$ onto the first coordinate.*

Proof. Follows from the fact that augmenting the input as a state ensures that constancy on the future inputs is enforced through state evolution, i.e. choosing the state as: $[x(t)^T, u(t)^T]^T$ and the augmented state dynamics for a LTI system as: $\begin{bmatrix} \mathcal{A} & \mathcal{B} \\ 0 & I \end{bmatrix}$ □

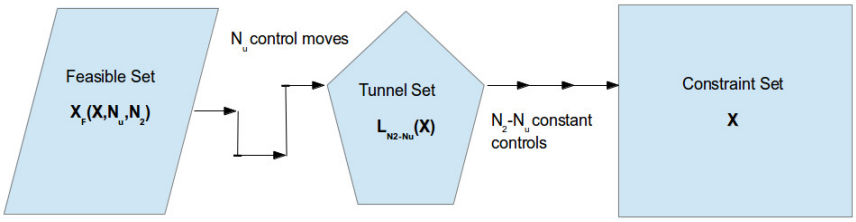


Figure 3.1: A conceptual depiction of the derivation of the feasible set starting from the constraint set through the tunnel set: $X_F(\mathbb{X}, N_u, N_2) = K_{N_u}(\mathbb{X}, L_{N_2-N_u}(\mathbb{X}))$

Remark 3.2.2. *The* ***maximal robust tunnel set*** $\tilde{L}_\infty(\mathbb{X})$*, contained in* $\mathbb{X}$ *(i.e. a set which is never exited by holding a constant control) is obtained by noting the relationship* (3.14) *and using it in algorithm 2. If the algorithm converges for a finite value of* i_L^* *then,* $\tilde{L}_\infty(\mathbb{X}) = \tilde{L}_{i^*}(\mathbb{X})$*, where* i_L^* *is the determinedness index.*

Lemma 3.2.2. *The maximal robust tunnel set* $\tilde{L}_\infty(\mathbb{X})$ *is convex and converges.*

Proof. For linear systems with convex, polytopic input, state constraints, the resulting maximal robust control invariant set $\tilde{C}_\infty(\mathbb{X})$ is convex and converges [53]. Since, the maximal robust tunnel set $\tilde{L}_\infty(\mathbb{X})$ uses augmented system which is again linear with convex polytopic constraints and is derived from $\tilde{C}_\infty(\mathbb{X})$ with a projection operation which preserves the properties, the $\tilde{L}_\infty(\mathbb{X})$ set inherits the characterization of the $\tilde{C}_\infty(\mathbb{X})$ set and hence is convex and converges. □

Note that, all the tools developed for robust sets remain perfectly valid for nominal systems, in which case $n(t) = 0$ and the invariant sets are represented devoid of the 'tilde'. The development of the tunnel set is not to be confused with the literature on tube-based methods where the control applied to the uncertain system is the composition of conventional MPC control for the nominal system added to gain times difference between nominal and actual state trajectories [54] .

3.3 Persistent Feasibility

In practice, especially when the system is nonlinear, one cannot guarantee that the solution is unique nor that the solver will return the optimal solution to the problem 3.1.2. It would therefore be useful if a result could be derived which allowed one to guarantee that the MPC controller is feasible for all time and for all disturbance sequences, even if a suboptimal control input is computed each time i.e. checking the existence of 'a' feasible solution. In this section, we derive the region of attraction for the given MPC problem 3.1.2:

1. that satisfies conditions $[D2 - D5]$

2. and without requiring optimality of the solution

to be infinite-time (persistent) feasible/stable (recollect that, the requirement $D1$ is already satisfied by the transformation matrices introduced in chapter 2). Now, we introduce new definitions and characterizations for the feasible region of our MPC problem 3.1.2 with short control horizon and no terminal conditions.

Definition 3.3.1. *The **robust feasible set** $\tilde{X}_F$ is the set of states for which an admissible control sequence exists that satisfies the state/output constraints i.e. a feasible control sequence to the MPC problem 3.1.2 exists, for all disturbance sequences.*

Theorem 3.3.1. *The robust feasible set $\tilde{X}_F(\mathbb{X}, N_u, N_2)$ of the MPC regulation problem 3.1.2 is given by:*

$$\tilde{X}_F(\mathbb{X}, N_u, N_2) = \tilde{K}_{N_u}(\mathbb{X}, \tilde{L}_{N_2-N_u}(\mathbb{X})) \tag{3.15}$$

Proof. The robust feasible set can be divided into two parts in time by approaching it from the end.
Consider the second part of the MPC problem 3.1.2 i.e. between control and prediction horizon $N_2 - N_u$, where the requirement is to keep the control moves constant to an admissible set $\mathbb{U}$ and satisfy the state constraints $\mathbb{X}$, for disturbance set $\mathbb{W}$. This set by definition 3.2.12 is the robust tunnel set $\tilde{L}_{N_2-N_u}(\mathbb{X})$.
Next, we consider the first part of the MPC problem 3.1.2 i.e. the window over the control horizon N_u, now the requirement is again to be able to find a control sequence in $\mathbb{U}$ that satisfies the state constraints $\mathbb{X}$ and lies in the target set $\tilde{L}_{N_2-N_u}(\mathbb{X})$ after N_u moves, for all disturbances. This by definition 3.2.10, is the robust controllable set $\tilde{K}_{N_u}(\mathbb{X}, \tilde{L}_{N_2-N_u}(\mathbb{X}))$, which is the overall robust feasible set and completes the proof. The entire process is depicted conceptually in Fig. 3.1 for the nominal case. □

Remark 3.3.1. *Due to finite-horizon nature of MPC, it is possible that a bad choice of design variables could lead to a solution with $\hat{x}^*(t+1|t) \in \mathbb{X} \setminus \tilde{X}_F$. This will result in infeasible problem next time instant, even in the absence of disturbances.*
Next, if $\tilde{X}_F \setminus \tilde{C}_\infty(\mathbb{X}) \neq 0$, it is possible that $\hat{x}^*(t+1|t) \in \tilde{X}_F \setminus \tilde{C}_\infty(\mathbb{X})$ which will cause $x(t+1) \notin \tilde{C}_\infty(\mathbb{X})$. Since there does not exist a control sequence that satisfy the constraints if the state is outside the robust maximal control invariant set, the MPC problem will become infeasible eventually.

Definition 3.3.2. *The MPC problem is **robust persistently feasible** iff the initial state and future evolutions belong to the robust feasible set i.e. $x(t+k) \in \tilde{X}_F, \forall k \in \mathbb{N}$.*

Theorem 3.3.2. *The MPC problem is robust persistently feasible iff* $x(t) \in \tilde{X}_F \cap \tilde{C}_\infty(\mathbb{X})$ *and* $\hat{x}^*(t+1|t) \in \tilde{X}_F \cap \tilde{C}_\infty(\mathbb{X})$*, that must hold for all* t*.*

Proof. If $x(t) \notin \tilde{X}_F \cap \tilde{C}_\infty(\mathbb{X})$, then it results in infeasibility from remark 3.3.1, so by contradiction $x(t) \in \tilde{X}_F \cap \tilde{C}_\infty(\mathbb{X})$ is necessary for feasibility. The requirement $\hat{x}^*(t+1|t) \in \tilde{X}_F \cap \tilde{C}_\infty(\mathbb{X})$ means that the problem is robust persistently feasible by mathematical induction, thereby proving the sufficiency. □

If the dynamics are linear and the constraints are compact, convex polyhedra, then $\tilde{X}_F$ is also a compact convex polyhedron.

3.3.1 Guidelines for stabilizing horizons

First, we establish through the following proposition that robust stability is a derivative of robust persistent feasibility.

Proposition 3.3.1. *If the robust feasible set* $\tilde{X}_F$ *is bounded and the MPC problem is robust persistently feasible, then the system is* ***robust stable*** *in a practical Lyapunov sense (i.e. boundedness of the closed-loop trajectories).*

Now, for the MPC problem to be robust stabilizing, we must ensure robust persistently feasibility through the parameters which are the control and the prediction horizon. Note that, the robust persistently feasible set $\tilde{X}_F$ is independent of the cost function and optimality of the solution. Therefore N_1 does not play a role as a non-zero value of N_1 is the same as choosing an appropriate penalty on the state predictions and the penalties being a part of the cost function have no effect on $\tilde{X}_F$.

Theorem 3.3.3. *If* $\tilde{X}_F(\mathbb{X}, N_u, N_2)$ *is robust control invariant, then the same MPC problem with control, prediction horizons* $N_u + n, N_2 + n$ *respectively producing* $\tilde{X}_F(\mathbb{X}, N_u + n, N_2 + n), n \geq 1$ *is robust persistently feasible.*

Proof. The robust control invariance of $\tilde{X}_F(\mathbb{X}, N_u, N_2)$ by definition implies that $\tilde{X}_F(\mathbb{X}, N_u, N_2) \subseteq \tilde{K}_1(\mathbb{X}, \tilde{X}_F(\mathbb{X}, N_u, N_2))$. However,
$\tilde{X}_F(\mathbb{X}, N_u + 1, N_2 + 1) = \tilde{K}_1(\mathbb{X}, \tilde{X}_F(\mathbb{X}, N_u, N_2)$ by application of the MPC control law. This implies that $\tilde{X}_F(\mathbb{X}, N_u+n, N_2+n) \subseteq \tilde{X}_F(\mathbb{X}, N_u, N_2)$, for $n = 1$ and is true $\forall n > 1$ by induction, which concludes robust persistent feasibility. □

Theorem 3.3.4. *If* $N_2 \geq N_u + i_L^*$*, then the size of the robust feasible set* $\tilde{X}_F$ *increases with the control horizon* N_u *until it exceeds the determinedness index* i_K^* *of the infinite robust controllable set* $\tilde{K}_\infty(\mathbb{X}, \tilde{L}_{N_2 - N_u}(\mathbb{X}))$*.*

Proof. The maximal robust tunnel set $\tilde{L}_{i_L^*}$ being control invariant, induces control invariance on $\tilde{K}_{N_u}(\mathbb{X}, \tilde{L}_{i_L^*}(\mathbb{X})), \forall N_u \in \mathbb{N}^+$. The sets being enclosed inside

the other with increasing N_u is a property of robust control invariant sets. The size increase stops beyond the determinedness index i_K^* of controllable sets, as by definition the sets converge. □

Corollary 3.3.1. *For a fixed control horizon N_u, the size of the feasibility set decreases with increasing prediction horizon until the determinedness index of the tunnel set i.e. $\tilde{X}_F(\mathbb{X}, N_u, N_{2a}) \subseteq \tilde{X}_F(\mathbb{X}, N_u, N_{2b}), N_{2a} > N_{2b}$ for all $N_2 \leq i_L^*$.*

Theorem 3.3.5. *The MPC problem is robust persistently feasible if the difference between the prediction and control horizons is larger than the determinedness index i_L^* of the maximal robust tunnel set $\tilde{L}_\infty(\mathbb{X})$ i.e. $N_2 - N_u \geq i_L^*$.*

Proof. Since, $\tilde{L}_\infty(\mathbb{X})$ is control invariant under constant control, any $i-$step controllable set to it i.e $\tilde{K}_i(\mathbb{X}, \tilde{L}_\infty(\mathbb{X})), \forall i \geq 1$ is also control invariant, which is in fact also the robust feasible set, and by the nesting property of invariant sets $\tilde{X}_F(\mathbb{X}, N_{u1}, i_L^*) \subseteq \tilde{X}_F(\mathbb{X}, N_{u2}, i_L^*), N_{u1} < N_{u2}$, which is necessary for robust strong feasibility. Note that, this is same as explicitly enforcing a positively invariant terminal constraint under the constant terminal control. The cost between the control and prediction horizons sum up to form the terminal cost. □

Based on the above three theorems, the following algorithms for tuning the horizons are proposed to stabilize the MPC problem without terminal conditions.

1. Compute determinedness of the robust tunnel set i_L^*, if finitely determined.
2. The stabilizing horizons are $N_u = 1, N_2 = i_L^* + 1$.

algorithm 3: Control horizon-1 tuning procedure

Now, if an additional requirement is to obtain the largest possible robust feasible region $\tilde{X}_F$, then:

1. Compute determinedness of the robust tunnel set i_L^*, if finitely determined.
2. Compute determinedness of the robust controllable set i_K^*, if finitely determined.
3. The stabilizing horizons are $N_u = i_K^*, N_2 = i_L^* + i_K^*$.

algorithm 4: Maximal robust feasible region $\tilde{X}_F$ tuning procedure

If, the robust tunnel set is not finitely determined, then:

1. Iterate over different values of the horizons with $N_u < N_2$ until $\tilde{X}_F(\mathbb{X}, N_u, N_2)$ is robust control invariant for N_u^*, N_2^*.
2. The stabilizing horizons are $N_u = N_u^* + 1, N_2 = N_2^* + 1$.

algorithm 5: Heuristic robust persistent $\tilde{X}_F$ tuning procedure

3.3.2 A posteriori certification of stability

In order to guarantee robust constraint satisfaction in safety-critical applications it is desirable that infeasibility of the MPC optimization problem is avoided at all costs. In other words, once inside the feasible set the system evolution should remain inside the feasible set for all time and for all disturbance sequences. Here, we develop test for checking robust persistently feasibility, given the MPC problem 3.1.2 without terminal conditions and shorter control horizon has already been implemented.

Theorem 3.3.6. *The MPC regulator that solves problem 3.1.2 is robust persistently feasible iff:*

$$\tilde{R}(\tilde{X}_F(\mathbb{X}, N_u, N_2)) \cap \tilde{X}_F(\mathbb{X}, N_u - 1, N_2 - 1) \subseteq \tilde{X}_F(\mathbb{X}, N_u, N_2) \tag{3.16}$$

Proof. $\tilde{R}(\tilde{X}_F(\mathbb{X}, N_u, N_2))$ is the set of states reachable from the robust feasible set $\tilde{X}_F(\mathbb{X}, N_u, N_2)$ using admissible inputs, while the set $\tilde{R}(\tilde{X}_F(\mathbb{X}, N_u, N_2)) \cap \tilde{X}_F(\mathbb{X}, N_u - 1, N_2 - 1)$ is the subset which is reachable using feasible control inputs which obey the state constraints. Therefore, after applying the feasible control computed by the MPC regulator, the next state $\hat{x}(t+1|t) \in \tilde{R}(\tilde{X}_F(\mathbb{X}, N_u, N_2)) \cap \tilde{X}_F(\mathbb{X}, N_u - 1, N_2 - 1)$. Now, if $\hat{x}(t+1|t) \in \tilde{X}_F(\mathbb{X}, N_u, N_2)$, then by mathematical induction all future evolutions of the system remain within the robust feasible set, which completes the proof. □

Corollary 3.3.2. *In the special case of control horizon $N_u = 1$, the robust persistent feasiblity test reduces to:*

$$\tilde{R}(\tilde{X}_F(\mathbb{X}, 1, N_2)) \cap \tilde{L}_{N_2-1}(\mathbb{X}) \subseteq \tilde{X}_F(\mathbb{X}, 1, N_2) \tag{3.17}$$

Proof. In the case with $N_u = 1$, $\tilde{X}_F(\mathbb{X}, 0, N_2 - 1) = L_{N_2-1}(\mathbb{X})$. □

Lemma 3.3.1. *In case of output disturbance, the state constraints $\mathbb{X}$ must be replaced by the robust output admissible set $\tilde{X}^g$, refer definition 3.2.6. Note that, input disturbances can be moved to the output by filtering it through the plant denominator A.*

Property	Tunnel set	Terminal set
Offline Computation	Admissible Set	Positively invariant set
Dependency	Controllable set, reach set	Terminal cost, control
Convergence	Not necessary	Strict requirement
Online Computation	No extra terms	Terminal set, cost added
Feasible region	Largest possible	Shrunk
Stability	Practical	Asymptotic
Certification	A priori, a posteriori	A priori

Table 3.1: The main differences between Tunnel set based certification and Terminal set based stabilization

This result in practice can be conservative, as optimality of the solution is not considered. Note that all the derived results, as usual, hold for the nominal case with zero disturbance i.e. $n(t) = 0$ and the corresponding sets are represented without the 'tilde'. Further, if the summation of cost between the control and prediction horizons is considered as a terminal cost resulting in equal control and prediction horizons and this terminal cost turns out to be a control Lyapunov function, asymptotic stability can be inferred. A sketch of the entire certification algorithm, together with the detailed computation of the associated sets is given in appendix A. Table 3.1 highlights the main differences between the MPC certification mechanisms based on the development of the tunnel set and the one that uses terminal constraint.

3.4 Recursive Feasibility

This section is concerned with robust feasibility when the solution to the finite horizon optimal control problem 3.1.2 is unique (e.g. optimal solution to a convex optimization problem).

Assumption 3.4.1. *For all $x(t) \in \tilde{X}_F$, a unique solution to problem 3.1.2 exists.*

The above assumption is made to guarantee that the MPC controller $\mathcal{U} : \tilde{X}_F \to \mathbb{U}$ is a single-valued map. It can be relaxed if the optimum is global as then an arbitrary selection can be made from the optimal set valued map to obtain a single-valued input each time.

Definition 3.4.1. *The MPC controller is **robust recursively feasible** iff for a subset $\tilde{X}_F^s$ of all states inside the feasible set $\tilde{X}_F$ and for all disturbances inside $\mathbb{W}$, the state of the plant at the next time instant lies inside the set $\tilde{X}_F^s$, i.e.*

$$\forall x(t) \in \tilde{X}_F^s : f(x(t), \mathcal{U}(x(t)), \mathbb{W}) \subseteq \tilde{X}_F^s \tag{3.18}$$

Once more, robust recursively feasibility is very strongly related to the ideas in set invariance. The concept of the robust reach set of the closed-loop is particularly useful in this context.

Theorem 3.4.1. *The MPC problem is robust recursively feasible iff the robust feasible set is a robust positively invariant set for the closed-loop system* $x(t+1) = f(x(t), \mathcal{U}(x(t)), \mathbb{W})$.

Proof. In this case, by definition of robust positive invariance, the closed-loop system trajectories never leave the robust feasible set, thereby ensuring robust recursive feasibility. □

Now, a given set is robust positively invariant iff the set of states reachable from the given set is contained within itself. From this statement, the following can be said:

Proposition 3.4.1. *The MPC controller* $\mathcal{U}$ *is robust recursively feasible iff* $\tilde{R}(\tilde{X}_F^s) \subseteq \tilde{X}_F^s$ *for the autonomous system comprised of the original plant in closed loop with the MPC controller* $\mathcal{U}$.

From the above proposition, it follows that for the MPC controller to be robust recursively feasible, it is necessary that $\tilde{X}_F^s$ is robust control invariant.

In order to derive a test or algorithm which implements the above condition, some structure regarding the problem has to be known. An explicit expression for the MPC control law needs to be derived, which we have duly presented in chapter 2.

Theorem 3.4.2. *Consider the constrained finite time optimal control problem 3.1.2. Then, the feasible set* X_F *is polyhedral, the optimizer* $\mathcal{U}^* : X_F \to \mathbb{R}$ *is continuous and PWA, i.e.*

$$\mathcal{U}^*(x(t)) = F_r \cdot x(t) + G_r \quad if \quad x(t) \in P_r \tag{3.19}$$

$$P_r = \{x(t) \in \mathbb{R}^n | H_r \cdot x(t) \leq K_r\}, \tag{3.20}$$

and the optimal solution J^* *is continuous, convex and piecewise quadratic (PWQ).*

Proof. Exactly the same as that of theorem 2.4.3. The two equations above should be matched with (2.43), (2.45) to get the expanded expressions of the terms involved. In particular $\{P_r\}_{r=1}^R$ is a polyhedral partition of $\tilde{X}_F$ with region indexed by r. □

Note that the evaluation of the PWA solution provides the same result as solving the quadratic program. Due to finite horizon, even in case of no model mismatch, the optimal open-loop trajectory is different from the trajectory which results form the closed-loop control [9]. This may therefore affect not only feasibility

but stability as well. As stated in introduction, this problem can be dealt with by enforcing terminal constraints but this inevitably results in large optimization problems which are unsuitable for fast systems [9]. Therefore the scheme here consists of implementing controllers with short horizons and subsequently analyzing for stability through feasibility of the closed-loop system.

The associated closed-loop dynamics are given by:

$$x(t+1) = (\mathcal{A} + \mathcal{B} \cdot F_r)x(t) + \mathcal{B} \cdot G_r \;\; if \;\; H_r \cdot x(t) \leq K_r. \tag{3.21}$$

Lemma 3.4.1. *For a given robust feasible compact set $\tilde{X}_F$, a robust positively invariant subset $\tilde{O}(\tilde{X}_F) \subseteq \tilde{X}_F$ induces robust recursive feasibility and practical Lyapunov stability, when the controller is applied in receding horizon manner.*

Proof. Follows from theorem 3.4.1. Note that, stability is in the sense of boundedness as limit cycles cannot be ruled out. □

One way to find such a positively invariant set is to iteratively remove states from which the trajectory exits the feasible set until no such states can be found. Let the transition matrix $\mathcal{V} \in \{0,1\}^{R \times R}$ (where R denotes the number of regions) and the modification vector $\mathcal{M} \in \{0,1\}^R$ store the feasible transitions and keep track of set modifications respectively.

$$\mathcal{V}(s,v) = 1, \; if \; \exists x(t) \in P_s : x(t+1) \in P_v, \; else \; \mathcal{V}(s,v) = 0; \tag{3.22}$$

$$\mathcal{M}(r) = 1, \; if \; P_r^{i+1} \neq P_r^i, \; else \; \mathcal{M}(r) = 0. \tag{3.23}$$

where i is the iteration number used in the algorithm 6, which is presented for completeness [55] (may be skipped without loss of continuity), following which the union of all remaining regions is equal to $\tilde{O}(\tilde{X}_F)$.

Lemma 3.4.2. *The presented polyhedral-invariant-set algorithm always converges (though not necessarily in finite time), as the initial feasible region X_F is finite and the volume is strictly decreasing. Further the convexity of the robust positively invariant set $\tilde{O}(\tilde{X}_F)$ is a necessary condition for it to be maximal i.e. $\tilde{O}_\infty(\tilde{X}_F)$ [55].*

Remark 3.4.1. *In case of output disturbance, the state constraints $\mathbb{X}$ must be replaced by the robust output admissible set $\tilde{X}^g$. This is called constraint tightening.*

Remark 3.4.2. *If constraints are tightened adaptively by estimating the disturbance online, then the region of attraction must be within the robust persistently feasible set $\tilde{O}(\tilde{X}_F)$. $\tilde{O}(\tilde{X}_F)$ guarantees that even after the disturbance, the controller will not exit the nominal $O(X_F)$, thus remaining feasible for all time.*

1. Given a controller partition $\{P_r\}_{r=1}^{R}$, compute the entries of the transition matrix $\mathcal{V}$. Subsequently create a transition set $\Pi = \{s, v \in \{1, \ldots, R\} | \mathcal{V}(s, v) = 1\}$, initialize $\mathcal{M}(r) = 1, \forall r \in \{1, \ldots, R\}$ and set $i = 0$.

2. For all $s, v \in \Pi$, do:
   If $\mathcal{M}(v) = 1$, compute and store the subset $\mathcal{S}_{s,v}^{i+1} = \{x(t) \in \mathbb{R}^n | x(t) \in \mathcal{P}_s^i, x(t+1) \in \mathcal{P}_v^i\}$ for the dynamics in (3.21). If $M(v) = 0$, set $\mathcal{S}_{s,v}^{i+1} = \mathcal{S}_{s,v}^{i}$.

3. For each start region $\mathcal{P}_s^i$, attempt to create the union $\mathcal{P}_s^{i+1} = \cup_{(s,v)\in\Pi} \mathcal{S}_{s,v}^{i+1}$. $\mathcal{P}_s^{i+1} \subseteq \mathcal{P}_s^i$ corresponds to the set of points $x(t) \in \mathcal{P}_s^i$ which remains within the partition $\cup_{\{1,\ldots,R)\}} \mathcal{P}_r$ in one time step.

   (a) The union $\mathcal{P}_s^{i+1}$ is convex:
   Set $\mathcal{M}(s) = 0$ if $\mathcal{P}_s^{i+1} = \mathcal{P}_s^i$ and set $\mathcal{M}(s) = 1$ if $\mathcal{P}_s^{i+1} \subset \mathcal{P}_s^i$.

   (b) The union $\mathcal{P}_s^{i+1}$ is non-convex:
   Add new regions to the controller partition, i.e. $\forall \mathcal{S}_{s,v}^{i+1} \neq 0$, set $\mathcal{P}_{R+1}^{i+1} = \mathcal{S}_{s,v}^{i+1}$ and $R = R + 1$. In addition, the matrices $\mathcal{V}, \Pi$ and vector $\mathcal{M}$ are updated accordingly.

4. If no region has been modified ($\mathcal{M}(r) = 0 \; \forall r \in \{1, \ldots, R\}$), the algorithm has converged and the invariant subset is found. Otherwise set $i = i + 1$ and goto step 2.

algorithm 6: Positive invariant subset computation

3.5 Examples

In this section, we demonstrate the procedure of obtaining and testing persistent feasibility in the nominal and perturbed cases, starting from transfer function model and using horizon $N_u \ll N_2$ without any terminal conditions, over a mass-spring-damper (MSD) setup followed by an unstable double integrated pitch dynamics.

3.5.1 Mass-Spring-Damper

The continuous time input-output model of the MSD Fig. 3.2 is given by,

$$m \cdot \ddot{y}(t) + c \cdot \dot{y}(t) + k \cdot y(t) = F(t) = u(t) \tag{3.24}$$

where $y(t), u(t) = F(t)$ are the measured output displacement and input force respectively with parameters m, c, k being the mass, damping and spring constant

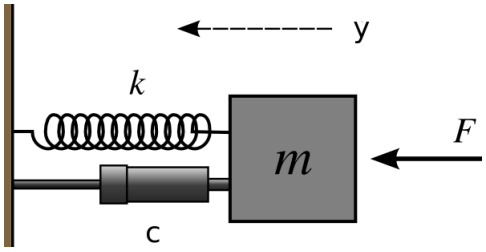


Figure 3.2: A schematic representation of the mass-spring-damper system.

respectively. Now, this is discretized with sampling time $T_s = 10\text{ms}$ with the parameter values $m = 1.7\text{kg},\ c = 9\text{N/m/s},\ k = 450\text{N/m}$ to obtain the following system:

$$y(t)[mm] = \frac{0.0545}{1 - 1.902q^{-1} + 0.9264q^{-2}}u(t)[\text{N}] + n(t)$$
$$subj.\ to\ \ ||y(t)||_\infty \leq 2\text{mm},\ \ ||u(t)||_\infty \leq 1\text{N}\ \ ||n(t)||_\infty \leq \gamma \tag{3.25}$$

An input-output MPC regulator is designed with control horizon $N_u = 1$ and prediction horizon $N_2 = 8$. In the nominal case $\gamma = 0$ and in the robust case $\gamma > 0$ is the upper bound on the output disturbance. Note that, no other information is required as the persistent feasibility technique is independent of the cost function.

The first-step is to deduce the state-space representation. The state vector is $x(t) = [y(t), y(t-1), u(t-1)]^T$ and the transformation matrices:

$$x(t+1) = \begin{bmatrix} 1.902 & -0.9264 & 0.0545 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot x(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \cdot u(t)[\text{N}] \tag{3.26a}$$
$$y(t)[\text{mm}] = [1\ \ 0\ \ 0] \cdot x(t) + n(t) \tag{3.26b}$$
$$subj.\ to\ \ ||x(t)||_\infty \leq [2 \times 2 \times 1]^T \ominus [\gamma \times 0 \times 0]^T,\ \ ||u(t)||_\infty \leq 1\text{N} \tag{3.26c}$$

Recollect that, in case of output disturbance, the state constraints are mapped to the output admissible set defined in 3.2.6. This is equivalent to tightening the constraint set on the first state, which is the output by the disturbance set acting on the output and is achieved by the Pontryagin difference operator in (3.26c).

In the nominal case $\gamma = 0$, the feasibility set is computed as per theorem 3.3.1 to:

$$X_F(\mathbb{X}, 1, 8) = K_1(\mathbb{X}, L_7(\mathbb{X})) \tag{3.27}$$

after the computation of the 7-steps tunnel set $L_7(\mathbb{X})$ and the 1-step controllable set to it i.e. $K_1(\mathbb{X}, L_7(\mathbb{X}))$. Next, the reach set of the feasibility set $R(X_F(\mathbb{X}, 1, 8))$ is computed and to check nominal persistent feasibility of the MPC problem, we

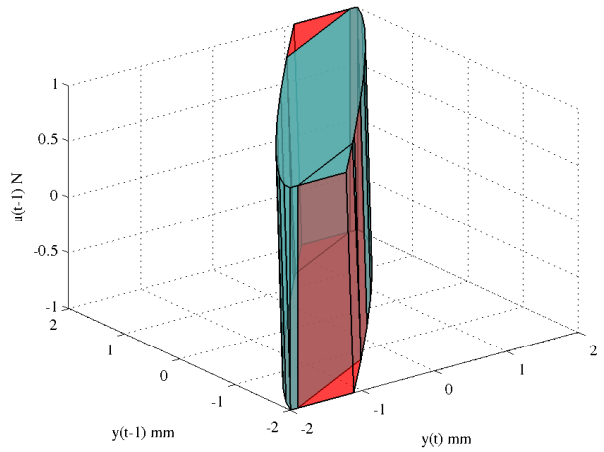


Figure 3.3: Persistent feasibility test:
$R(X_F(\mathbb{X},1,8)) \cap L_7(\mathbb{X})$ $(cyan) \subseteq X_F(\mathbb{X},1,8)$ (red)

make use of corollary 3.3.2:

$$R(X_F(\mathbb{X},1,8)) \cap L_7(\mathbb{X}) \subseteq X_F(\mathbb{X},1,8) \tag{3.28}$$

Indeed, this test is fulfilled as can be seen graphically in Fig. 3.3 and thus a certificate of practical nominal stability can now be issued to this MPC controller.

Next, a perturbation in the form of 10% additive output disturbance is considered i.e. with $\gamma = 0.2\text{mm}$ in (3.25). In this case, the robust feasibility set is computed as per theorem 3.3.1 to:

$$\tilde{X}_F(\mathbb{X},1,8) = \tilde{K}_1(\mathbb{X}, \tilde{L}_7(\mathbb{X})) \tag{3.29}$$

Next, to prove robust persistent feasibility of the MPC problem, we make use of corollary 3.3.2:

$$\tilde{R}(\tilde{X}_F(\mathbb{X},1,8)) \cap \tilde{L}_7(\mathbb{X}) \subseteq \tilde{X}_F(\mathbb{X},1,8) \tag{3.30}$$

As one would expect, there is a reduction in the region of attraction as compared to the nominal case, to account for the disturbance and the test is fulfilled. A graphical verification can be made through Fig. 3.4.

Thus far, we have shown a posteriori certification of the MPC controller. However, one may design a persistently feasible and stabilizing MPC controller to start with, by following the guidelines of 3.3.1. The resulting algorithms 3-5 can be applied to the MSD system of (3.26). Following algorithm 3, the determinedness index of the tunnel set is found to be $i_L^* = 31$ leading to the stabilizing horizons

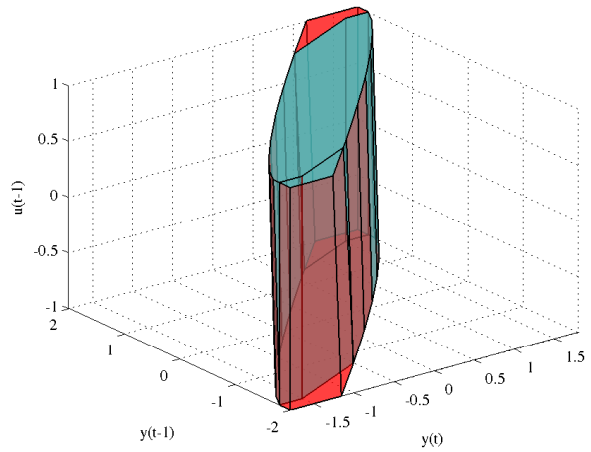


Figure 3.4: Robust Persistent feasibility test:
$\tilde{R}(\tilde{X}_F(\mathbb{X}, 1, 8)) \cap \tilde{L}_7(\mathbb{X})$ $(cyan) \subseteq \tilde{X}_F(\mathbb{X}, 1, 8)$ (red)

$N_u = 1, N_2 = 31 + 1 = 32$. Note that the MPC controller with these parameters returns a feasibility region with volume $7.6\mathrm{Nmm}^2$ compared to $8.6\mathrm{Nmm}^2$ for the case before with $N_u = 1, N_2 = 8$. This reduction in the region of attraction is because of making the tunnel set control invariant, which is an implicit way of imposing terminal constraint.

In case, a larger N_u is admissible, algorithm 4 may be used. It further requires the computation of the determinedness index of the controllable set which in this case is $i_k^* = 9$, suggesting the stabilizing horizons to be $N_u = 9, N_2 = 9 + 31 = 40$. A shorter but closer pair of control, prediction horizons may be obtained by the application of algorithm 5. It requires the horizons which make the feasible set control invariant and for the MSD the numbers are $N_u = 2, N_2 = 3$, leading to the stabilizing horizons of $N_u = 2 + 1 = 3, N_2 = 3 + 1 = 4$. All the three algorithms, of course satisfy the persistent feasibility subset test of theorem 3.3.6.

3.5.2 Longitudinal Flight Control

It is well know that the relationship between the flight moment to pitch angle is that of a double integrator. However, while performing an automatic take-off, the available moment is limited which makes it a constrained control problem. Such a double integrator plant is one of the most fundamental systems in control applications, representing single-degree-of-freedom translational and rotational motion comprising of several other applications of low-friction, free rigid body mo-

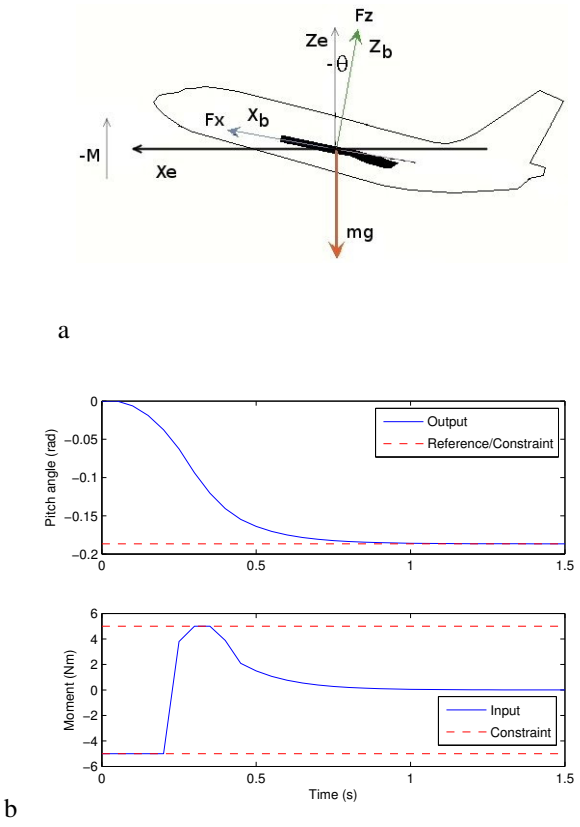


Figure 3.5: (a): Longitudinal dynamics of an aircraft, (b): Pitch control with MPC using input moment.

tion. Saturated control of double integrator has been studied in [56] using Linear Quadratic Gaussian, output feedback, directive adaptive controllers amongst others and are shown to have performance, stabilizing problems.

The (simplified) equations of motion describing longitudinal flight dynamics of a fixed wing aircraft can be written down as [57]:

$$m \cdot \ddot{x}_b = F_x + m \cdot g \cdot sin\theta - m \cdot \dot{z}_b \cdot \dot{\theta} \tag{3.31}$$

$$m \cdot \ddot{z}_b = F_z - m \cdot g \cdot cos\theta + m \cdot \dot{x}_b \cdot \dot{\theta} \tag{3.32}$$

$$I_{yy} \cdot \ddot{\theta} = M \tag{3.33}$$

$$\dot{T}_l = F_x, \quad F_z = max(sat(T_l), m \cdot g) \tag{3.34}$$

$$\dot{x}_e = \dot{x}_b \cdot cos\theta + \dot{z}_b \cdot sin\theta \tag{3.35}$$

$$\dot{z}_e = \dot{z}_b \cdot cos\theta - \dot{x}_b \cdot sin\theta \tag{3.36}$$

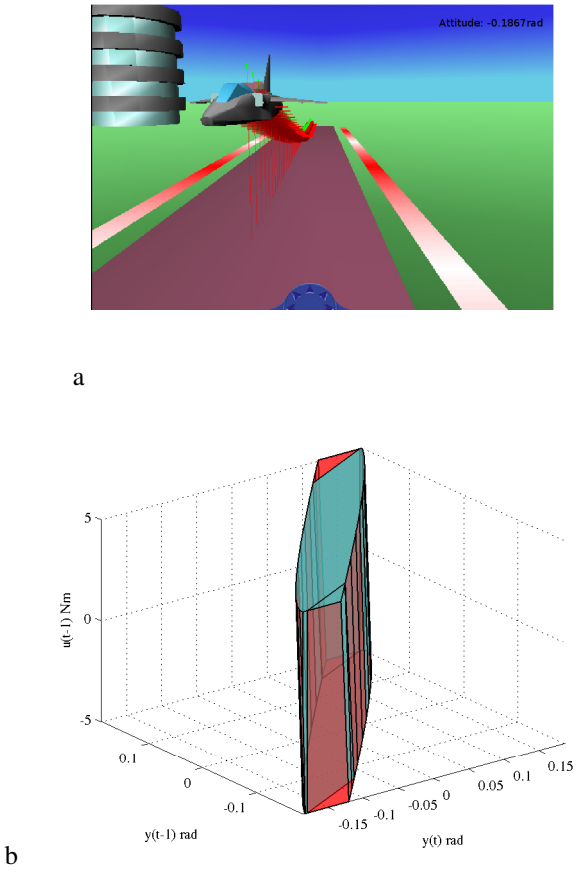


Figure 3.6: (a): A pitch controlled automatic take-off scenario, (b): Persistent feasibility test: $R(X_F(\mathbb{X}, 2, 12)) \cap X_F(\mathbb{X}, 1, 11)$ $(cyan) \subseteq X_F(\mathbb{X}, 2, 12)$ (red).

where $m = 1000\text{kg}, I_{yy} = 1000\text{kgm}^2, g = 9.81\text{m/s}^2$ are the flight mass, moment of inertia about y-axis, acceleration due to gravity respectively, F_x, T_l, F_z, x_b, z_b are the net thrust, take-off lift, lift, forward, vertical displacements with respect to the body axis respectively and x_e, z_e, M, θ are the forward, vertical displacements with respect to the earth axis, moment, pitch angle respectively. The first two, third are the translational, rotational equations of motion in body axis respectively and the last two relate to the earth axis. The fourth equation develops the lift as the maximum between saturated take-off lift and plane weight ground compensation, with the upper saturation set to 11000N. The schematic of the longitudinal dynamics of a fixed wing aircraft is given in Fig. 3.5(a).

Consider an automatic take-off scenario where the the maximum net throttle

of $F_x = 3000\text{N}$ is applied and after 1s as the lift force develops, we are required to pitch the aircraft to a certain commanded angle $\theta_r = -0.1867\text{rad}$ by manipulating the input moment constrained to $||M||_\infty \leq 5\text{Nm}$ as part of the aircraft take-off regulations to quickly attain height. A further constraint on the pitch angle $||\theta||_\infty \leq 0.1867\text{rad}$ is imposed to guarantee no overshoot. Naturally, certification of the associated controllers is quintessential to passenger safety.

The above set of equations in continuous time is discretized with a sampling time of $T_s = 0.05\text{s}$ and an output feedback MPC controller is designed with $N_u = 2, N_2 = 12$; note that the control horizon is chosen to be at least equal to the number of unstable plant modes. The results are plotted in Fig. 3.5(b), the desired pitch is attained very fast while obeying the input constraints. A snapshot (not to scale) of the take-off animation is shown in Fig. 3.6(a). Next, as part of our certification procedure, the feasibility set is computed as:

$$X_F(\mathbb{X}, 2, 12) = K_2(\mathbb{X}, L_{10}(\mathbb{X})) \tag{3.37}$$

followed by the subset test from theorem 3.3.6:

$$R(X_F(\mathbb{X}, 2, 12)) \cap X_F(\mathbb{X}, 1, 11) \subseteq X_F(\mathbb{X}, 2, 12) \tag{3.38}$$

The MPC pitch controller satisfies the above subset test as can be seen graphically in Fig. 3.6(b) and thus is certified persistent feasible and stable under nominal conditions.

Note that even though the examples consider a single input single output system, the technique can be applied seamlessly to multivariable input-output systems by formulating the equivalent state-space by using corollary 2.4.1. The nominal feasible region is then composed of the time history of multiple outputs and inputs. Subsequently, the developed tests and guidelines for persistent/recursive feasibility/stability can be used as usual.

3.6 Summary

This chapter makes the most fundamental contribution of the thesis i.e. it develops an apologia for not using terminal stabilising conditions in industrial model predictive control. The reasons for the industry not using the terminal conditions in their MPC formulations like difficulty in computation, compromising performance, reduction of feasible region etc. have been highlighted. This gave us enough motivation to construct a theoretical framework to be able to certify the industrial implementations that in addition use a shorter control horizon than prediction horizon.

The foundations of our work is through invariant set theory. Therefore, a number of ideas, definitions and results from invariance theory are presented together.

A robust controllable set is shown to be one of the key ingredients for computation of feasible sets i.e. the region in which the MPC has a solution. Further, a new set called the tunnel set is introduced to tackle the industrial MPC scenario where control horizon is shorter then prediction horizon. A brief note on computation of these sets and on the important concept of control invariance i.e. existence of a feasible control that would restrict the evolution to the same feasible set has been included.

The notion of robust persistent feasibility was introduced. It was particularly tailored towards MPC with shorter control horizon compared to prediction horizon through tunnel sets. An MPC problem is robust persistently feasible if and only if it is feasible for all time under the maximum allowed disturbances, and relies on the existence of 'a' solution. The robustness against output disturbances is guaranteed through output admissible sets, which are particularly easy to construct as the state vector in our case is formed by the time-history of the outputs.

New guidelines have been developed and presented adequately for choosing control and prediction horizons such that the MPC problem without terminal conditions is guaranteed robust persistently feasible. In case of fast applications, these rules may be deliberately violated in order to have very short horizons resulting in simpler MPC problem. In such cases, a sufficient condition was derived for guaranteeing a posteriori robust persistent feasibility, for the MPC without terminal conditions.

The concept of recursive feasibility is introduced to reduce the conservativeness compared to persistent feasibility as it takes into account the structure of the problem (i.e. uses the cost function to derive an explicit solution). The downside is that it requires the off-line computation of the explicit solution to the MPC problem and then deriving a positively invariant subset. The robust recursive feasibility, once again, is through the computation of output admissible sets.

Finally, we successfully certify (robust) persistent feasibility/stability of input-output MPC of constrained mass-spring-damper and longitudinal flight dynamics without any terminal conditions, both in the nominal and perturbed cases and apply the prescribed stabilizing guidelines to obtain safe horizons.

4

Penalty Adaptive Predictive Control

The MPC controller's popularity can almost solely be attributed to the fact that it is a means of systematically handling constrained optimal control problems [44]. However, the two immediate bottlenecks which stem out of it are: (1) the linear problem suddenly becomes nonlinear due to constraints and thus none of the powerful classical linear system tools can be used for analysis and (2) constrained optimization can be expensive due to online quadratic programming (especially for fast processes).

The only comprehensive set of tools for nonlinear systems are the Lyapunov based methods, which have been used by far in the literature [9]. We proposed to use invariant sets based methods in chapter 3 to guarantee stability in the sense of boundedness of trajectories and also suggested the stabilizing lengths of horizons without terminal conditions. Now, the natural question is to ask, can we go another step further towards asymptotic stability of MPC without terminal conditions and give some guidelines for tuning the penalties on control moves to achieve this? Well, turns out that through an appropriate penalty adaptive mechanism, asymptotic stability without terminal conditions can be given for the adapted input constrained system in the nominal case, and we like to call this penalty adaptive MPC (PAMPC). Since this method preserves linearity of the problem in the case of repetitive applications like positioning systems in nominal settings, there is no need of quadratic programming and hence is much efficient as well, compared to [47] where similar properties are obtained but in a nonlinear setting. Thus PAMPC provides a solution for both the problems stated above.

4.1 Introduction

Some ideas have existed in the literature which mention removing a constraint from the optimization problem and thereby making it computationally attractive, by penalizing the excess control action, thus removing the constraints on the minimum and maximum sizes of inputs [44]. Some others [58] have redefined the objective function to include the optimization of the weights on the magnitude of the inputs. However, neither of these approaches have come up with a systematic means of adapting the weights such that the closed-loop system becomes linear. Moreover, we adapt penalties on the control increments which is more logical to get zero steady state error in the perturbed case. An efficient offline formulation of robust MPC is proposed in [59] but at the expense of using linear matrix inequalities and online bisection search. Another track of research [60] made use of reference governors which make the constraints of the inner loop inactive at the expense of introducing another QP problem for generating references to be solved at the outer loop.

In this chapter, we present a constraint management method, PAMPC that adapts the weights on the control increments. PAMPC is a constrained MPC formulation that allows to derive closed loop transfer functions in the case of input constrained repetitive systems under nominal conditions, thus enabling the use of linear system tools for instance to prove asymptotic stability for the adapted MPC without terminal conditions. All this can be achieved by first fixing the control and prediction horizons to the stabilizing values prescribed in section 3.3.1 or by alternate tuning mechanisms like relating controller parameters to the structure of the model. Then the penalty adaptation adapts the control increment weight matrix based on the current state, reference trajectory and active constraints to satisfy the input constraints. Further, we present tunneling as a straightforward method to recover performance and feasibility under process disturbances as opposed to more conservative and computationally demanding approaches like min-max MPC [7]. In the second part of the chapter the PAMPC is tested on the control of challenging non-collocated systems in which the actuation and sensing occurs at different locations.

When the sensor measures at the same point where the actuation occurs, such systems are termed as collocated. It turns out that the dynamic characteristics of collocated systems are favourable for control system design. However, in real life, mechatronic applications generally are non-collocated and in addition underdamped which pose unique challenges for the control engineer. Some such cases include, bridges or flexible beams [61] and production machines e.g. harvesters [62]. When controllers are designed for lightly damped structures, shifting or damping resonances is often the main concern. However, when it comes to non-collocated systems, anti-resonances should be considered as well. Further, an

inaccurate model estimation may result in interchanging the order of poles and zeros. This, together with the presence of hard actuator constraints, could render the closed-loop unstable. Therefore, classical control techniques like pole-zero compensators with no systematic means of handling constraints can perform poorly given these characteristics of non-collocated systems [63].

In a collocated setting, GPC has been found to be suitable for damping the first vibration mode of a pinned-free beam model by [64]. In [65] GPC is once again used for reducing the vibration of ground vehicles but without constraints. The work carried by [66] highlighted the usefulness of MPC in handling constraints for active noise and vibration control. However, to our knowledge no results are reported on model based predictive control of non-collocated systems subjected to constraints. Here, we demonstrate the effectiveness of PAMPC for the control of constrained non-collocated system.

4.2 Constrained MPC by Penalty Adaptation

We consider the following cost function formulation, without terminal conditions:

$$\begin{aligned} V^* = &\min_{\Delta U} \ (R-Y)^T(R-Y) + \Delta U^T \cdot \Lambda \cdot \Delta U \\ &subj. \ \ to : \Delta U \in \Delta \mathbb{U}_c \end{aligned} \tag{4.1}$$

where $\Delta \mathbb{U}_c$ is the set of point-wise in time convex constraints. In addition, $R, Y, \Delta U$ are now vectors of references, predicted outputs, control increments respectively introduced in section 2.3 and Λ a diagonal matrix of penalties. Further, it is possible to define a first order reference trajectory over set-point with time-constant τ. Note, that the cost function is the same as that introduced in chapter 2 formulated in the input-output domain.

The advantage of directly penalizing ΔU makes sense as it penalizes the high frequencies, which is of particular relevance for underdamped systems. It is well known that all forms of constraints i.e. on input, output, input rate can be accommodated in $\Delta \mathbb{U}_c$. For instance, $\Delta \mathbb{U}_c$ can be obtained from the input constraint vector $|U| \leq U_c$ through the relation $|u(t-1) + L \cdot \Delta U| \leq \Delta \mathbb{U}_c$, where L is a lower triangular matrix of ones. Now we have a quadratic programming (QP) problem which can be solved either by interior-point, active-set based iterative optimizers or even fast gradient methods [44]. Note that, by doing so not only do we increase the computational burden but also loose the analytical solution to (4.1). Moreover a whole set of controller parameters need to be tuned.

The original contribution of this chapter lies in re-formulating the entire constrained optimization problem to an equivalent unconstrained one with adapted penalties such that all the constraints are satisfied. Thus, as a first step all the controller parameters like horizons, etc., must be fixed beforehand either by the

stabilizing conditions derived in chapter 2 or any other tuning mechanism and only then the penalty is adapted online to ensure optimal constraint satisfaction. We call this controller the PAMPC, the details follow.

In the second step, the controller is initialized with the unconstrained solution to (4.1), which is:

$$\Delta U = (G^T G + \Lambda)^{-1} G^T (R - \bar{Y}) \tag{4.2}$$

i.e. the well known least squares solution with G, the step response matrix and $\bar{Y}$ the vector of base response.

In the third step, we check for constraint violation. Let us say v is the index of the constraint in the set $\Delta\mathbb{U}_c$ that is violated. Denote the error by $E = R - \bar{Y}$. Now rewrite (4.2) as:

$$\Lambda.\Delta U = -G^T G \cdot \Delta U + G^T \cdot E \tag{4.3}$$

The idea is to fix the violated constraint $\Delta u(t + v - 1|t)$ with its limit in $\Delta\mathbb{U}_c$ in the above system of equations and solve for the corresponding control penalty together with the rest of the control inputs, thus maintaining the solvability of the system. Let us denote the v^{th} column of matrix G by G_v and the matrix formed by rest of the columns other than v by $\tilde{G}_v$. Similarly, let the vector $\Delta\tilde{U}_v$ denote all the elements except v^{th} and Δu_v the v^{th} element. Finally Λ_v denotes the v^{th} diagonal element of matrix Λ and $\tilde{\Lambda}_v$ the matrix with v^{th} row and column removed.

Collecting all the Δu's other than the one which is violated, we have:

$$\tilde{\Lambda}_v \cdot \Delta\tilde{U}_v = -\tilde{G}_v^T \tilde{G}_v \cdot \Delta\tilde{U}_v - \Delta u_v \cdot \tilde{G}_v^T G_v + \tilde{G}_v^T \cdot E \tag{4.4}$$

Thus, a solution to the above set of equations can be found as:

$$\Delta\tilde{U}_v = (\tilde{G}_v^T \tilde{G}_v + \tilde{\Lambda}_v)^{-1} (-\Delta u_v \cdot \tilde{G}_v^T G_v + \tilde{G}_v^T \cdot E) \tag{4.5}$$

The fourth step is to form the optimal control move vector ΔU by inserting the fixed value Δu_v in the above computed control sequence $\Delta\tilde{U}_v$. Now, we are in a position to compute the control penalty such that the active constraint Δu_v is respected:

$$\lambda_v \cdot \Delta u_v = -G_v^T \cdot G \cdot \Delta U + G_v^T \cdot E \tag{4.6}$$

The steps three and four are repeated by sequentially checking for constraint violations in the future time steps until all constraints are satisfied. Once all constraints are satisfied, the absolute control applied to the plant is formed by: $u(t|t)^* = u(t-1) + \Delta u_1$. This forms the outer loop which runs within each sampling interval. In the next sampling time, this loop along with step two are repeated and so on.

Theorem 4.2.1. *Convergence of PAMPC control loop is always achieved.*

Proof. Consider a compact constraint set with origin in its interior. The PAMPC algorithm makes $|\Delta u(t+v-1|t)|$ smaller every time when it exceeds constraint $|\Delta u_v|$ by making $\Delta u(t+v-1|t) = \Delta u_v$. This in turn means that λ_v increases with each iteration. Finally, infinite penalty on $\Delta u(t+v-1|t)^2$ would mean $\Delta u(t+v-1|t) \to 0,\ \forall v$, refer (4.2), which satisfies the stop criterion of PAMPC. □

Remark 4.2.1. *The PAMPC algorithm can have two variations. In the first variation which is applicable to repetitive applications under nominal conditions, the optimal penalty matrix is computed only once offline for the MPC problem. This can be done by reinitializing the penalty matrix at the next iteration to the one computed before and in case there are no further adaptations, then the Λ of the first iteration would satisfy constraints every time. However, if this is not the case, then Λ can be chosen as the supremum amongst all the adaptations. Thus with Λ fixed, the closed-loop system remains linear and the closed loop characteristic polynomial can be derived to be:*

$$\phi_c = A \cdot I + q^{-d} \cdot B \cdot J \tag{4.7}$$

The expressions for the polynomials I, J have been derived in section 2.3 through Diophantine equations. This allows us to comment on the asymptotic stability of the adapted nominal system by checking if the polynomial ϕ_c is Hurwitzian (i.e. roots within unit circle).
In the second variation of the PAMPC, the penalty matrix is re-initialized to 0 or a very small constant at the beginning of each iteration, and runs online. A closed-form solution in this case would be more involved and may be given in the form of a bound over the set of closed-form solutions corresponding to the set of changes in the penalty matrix. This is effective in the presence of unknown disturbances, as the associated change in dynamics can necessitate a re-adaptation of the penalty matrix. However, this adaptation cannot be estimated beforehand and hence the closed-form solution is lost in this case.

Thus, PAMPC as well satisfies the stringent conditions $D1-5$ of chapter 3, for the input constrained case.

4.3 Robust Design of PAMPC

4.3.1 Robust Feasibility of PAMPC by Tunneling

Robustness is delivered by PAMPC through an appropriate design of the disturbance filter. For well damped processes, it is common to choose $C/D = 1/(1-q^{-1})$. However, for poorly damped systems, the 'integrator' disturbance filter gives oscillatory response.

Therefore, the first adaptation we make is to introduce the process denominator A as an additional factor of D which ensures that A cancels out of the closed loop transfer functions of (2.19). The second step is to choose $C = (1 - \rho.q^{-1})^{n_A}$ with $0 \leq \rho < 1$ in order to ensure that the closed loop is stable. In general, higher values of parameter ρ increases robustness but disturbances are rejected slower.

For constrained systems, however, feasibility should not be lost i.e. under possible acting disturbances, the controller output must lie within the constraints. As opposed to taking a conservative approach of computing the controller for the worst case scenario of disturbance sequence, we propose an online methodology to maintain feasibility by 'tunneling' i.e. creating tunnels through the input constraints. This approach is presented in the following two steps:

1. Compute and store the error at output as the difference between predicted model output and actual measured output i.e $n(t)$. From this, estimate the disturbance acting on the input: $u_d(t) = (A/B).n(t)$. Note that the plant must be inverse stable. In case of noise, filtered measurements must be stored.

2. Predict future input disturbance based on:
   $u_d(t+k) = f(u_d(t+k-1), u_d(t+k-2), \ldots)$ where $k \in [0, N_u - 1]$ and f is a dynamic system learning kernel like neural network, the complexity of which depends on the complexity of the disturbance signal. Next, update the input constraints $\mathbb{U}_c$ based on the tunnel: $\mathbb{U}_t = \mathbb{U}_c - U_d$, where $\mathbb{U}_t$ is the updated tunneled point-wise in time input constraint and U_d contains the predicted 'u_d's.

Theorem 4.3.1. *Consider a PAMPC controller with disturbance filter designed for robustness against a class of disturbances. Robust feasibility can then be guaranteed if the predicted input disturbance sequence U_d is subtracted from input constraints $\mathbb{U}_c$ through the control horizon N_u.*

Proof. Consider a robust PAMPC controller that computes an optimal, constraint admissible sequence U^*. Since, the optimization problem is solved for constraints $\mathbb{U}_c$, any or all of the future control values can lie on the constraint. In that case, the real control input to plant is: $U_p = \mathbb{U}_c + U_d$, clearly violating the constraints. But, if the constraint set is tunneled to $\mathbb{U}_t = \mathbb{U}_c - U_d$, we have: $\mathbb{U}_p = U_t + U_d = (\mathbb{U}_c - U_d) + U_d = \mathbb{U}_c$ which is feasible. □

The above approach would guarantee that the robust control inputs remain constraint admissible even under process disturbances.

4.3.2 An Alternate Tuning Procedure

Here we present the choice of tuning parameters except penalty Λ which are needed in the first step for PAMPC. Tuning of MPC controllers has drawn sig-

nificant attention in the literature, however the vast majority of analytical tuning methods are applicable only when the constraints are inactive [67]. The rest of the tuning methods are trial and error iterative approaches [68] or heuristic auto-tuners like particle swarm optimization [69]. It has been pointed in [10] that a manifold increase in computation speed occurs if the structure of the MPC problem at hand (e.g. warm starting for interior point methods) is exploited.

The first approach of obtaining the values for stabilizing control and prediction horizons is by following the guidelines which were developed in section 3.3.1. This gives the minimum horizons that make the feasible region control invariant, and subsequently persistently feasible. Recollect that, these rules are valid independent of the cost function and hence the penalty adaptation mechanism will not alter the sanctity of the tuning mechanism. As was noted earlier, this mechanism could be a little conservative and hence we present another alternative (not so rigorous) way here.

The computational complexity in each iteration scales with the square of control horizon i.e $O(N_u^2)$ for PAMPC and thus it is advisable to use short control horizons. However a control horizon of $N_u = 1$, which results in mean-level control is not capable of optimal performance in high order systems which have multiple modes [6]. A time-optimal performance can definitely be guaranteed by choosing the dead-beat settings for the unconstrained case.

Theorem 4.3.2. *If $N_2 \geq n_A + n_B + d$, $N_1 = n_B + d$, $N_u = n_A + 1$ with $d \geq 1$, then minimization of the cost function drives $y(t)$ in $n_B + d$ samples to the reference (n_A, n_B are the degrees of polynomials A, B respectively) [70].*

Proof. A regulation problem by minimization of the cost function (4.1) between the chosen $[N_1, N_2]$ with zero penalty ensures the only solution is when $y(t + n_B + d) = 0$. Further the equality constraints $\Delta u(t + k - 1|t) = 0$ for $k > N_u$ ensures the output remains at 0 as well. □

For the case with constraints, the control horizon N_u and minimum prediction horizon N_1 remain the same as in dead-beat settings as these are strongly related to the structure of the plant. One way of choosing the prediction horizon is by fixing it to $N_2 = \text{int}(\text{settling time})/T_s$ [71]. This choice of $N_u \ll N_2$ increases the stability of the closed loop, as this is equivalent to large terminal penalty. The time constant of the reference trajectory τ dictates the closed loop pole and thus must be fixed according to the desired speed of the closed loop. This leaves only the penalty term Λ which should be initialized to a very small value ≈ 0 and is then adapted online by PAMPC. Note that, in general Λ is fixed beforehand, and such a choice cannot be optimal under constraints.

4.4 Test case: Non-collocated mass-spring-damper

Non-collocation arises when the input force acts on the system at one point and the sensor measures the response at another. These are limitations posed by the design of the mechatronic systems. It may be seldom feasible to act and sense at the same location in reality. The flexible beam model is often used to analyze several characteristics of underdamped, non-collocated systems and their control. In [72], the cantilever beam model was used to study vibration suppression with non-collocated piezoelectric actuator and accelerometer.

Collocated systems, where the sensor and actuator are placed in the same position has the following property [63]: there is just one anti-resonance between two consecutive resonances. However, for *non-collocated* actuator-sensor systems, the above property is lost. It additionally poses the following problems:

1. As the sensor moves away from the actuator, the zeros migrate along the imaginary axis towards infinity and reappear from infinity on real axis.

2. If the resulting non-minimum phase zeros are within the system bandwidth, they can put severe restriction on the control system.

3. A pole-zero flipping might result due to modest variations in system parameters and can cause the corresponding branch in the root locus to become unstable!

As a direct consequence of the above, non-collocated control suffers from lack of robustness. We prove this over a mass-spring-damper set-up, that classical controller like PID lacks severely in terms of performance and stability.

4.4.1 Mass-spring-damper setup

The setup of Fig. 4.1 corresponds to a rectilinear electromechanical apparatus from ECP. The input of the plant is the voltage sent to the motor u and the outputs of the plant are the mass displacements (y_1 and y_2).

The electrical motor dynamics are fast compared to the mechanical dynamics, which means that the motor can be represented by a pure static gain:

$$F(t) = K \cdot u(t) \tag{4.8}$$

The parameters of the system are:

$$\begin{aligned} &m1 = 1.7\text{kg}, m2 = 1.2\text{kg}, k1 = k2 = 800\text{N/m}, \\ &k3 = 450\text{N/m}, c1 = 9\text{N/m/s}, K = 3.35\text{N/V} \end{aligned} \tag{4.9}$$

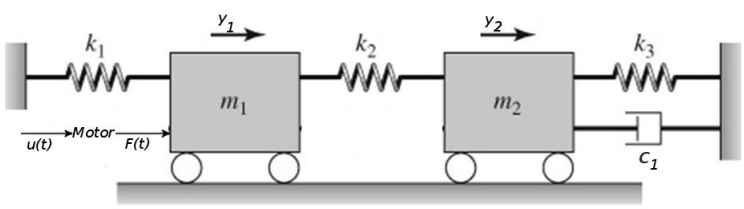


Figure 4.1: Mass spring damper setup

A mathematical representation of the system in Fig. 4.1 is derived from the free body diagram and the application of Newton's Second Law of motion.

$$\begin{aligned} F(t) &= m_1 \cdot \ddot{y_1} + k_2 \cdot (y_1 - y_2) + k_1 \cdot y_1 \\ 0 &= m_2 \cdot \ddot{y_2} + c_1 \cdot \dot{y_2} + k_3 \cdot y_2 - k_2 \cdot (y_1 - y_2) \end{aligned} \tag{4.10}$$

Assuming zero initial conditions, the Laplace transform of (4.10) results in :

$$\begin{aligned} F(s) &= m_1 \cdot s^2 \cdot Y_1(s) + (k_1 + k_2) \cdot Y_1(s) - k_2 \cdot Y_2(s) \\ 0 &= m_2 \cdot s^2 \cdot Y_2(s) + c_1 \cdot s \cdot Y_2(s) + (k_2 + k_3) \cdot Y_2(s) - k_2 \cdot Y_1(s) \end{aligned} \tag{4.11}$$

Further algebraic manipulations lead to:

$$\frac{Y_1}{F} = \frac{m_2 \cdot s^2 + c_1 \cdot s + (k_2 + k_3)}{CharPoly}, \quad \frac{Y_2}{F} = \frac{k_2}{CharPoly} \tag{4.12}$$

or, including also model of the motor:

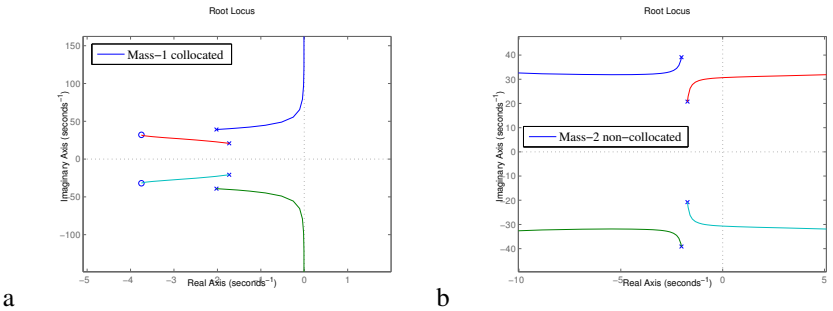
$$\frac{Y_1}{U} = K\frac{m_2 \cdot s^2 + c_1 \cdot s + (k_2 + k_3)}{CharPoly}, \quad \frac{Y_2}{U} = K\frac{k_2}{CharPoly} \tag{4.13}$$

where the characteristic polynomial $CharPoly$ is given by:

$$\begin{aligned} m_1 \cdot m_2 \cdot s^4 + m_1 \cdot c_1 \cdot s^3 + [m_1 \cdot (k_2 + k_3) + m_2 \cdot (k_1 + k_2)] \cdot s^2 + \\ c_1 \cdot (k_1 + k_2) \cdot s + k_1 \cdot k_2 + k_2 \cdot k_3 + k_3 \cdot k_1 \end{aligned} \tag{4.14}$$

In the first case, the position encoder measures the displacement of mass one y_1, at the same point as the input force from motor actuation, u, thus making the system collocated. In the second case, the displacement of mass two y_2 is measured at a different point than the input force u, thus making the system non-collocated.

The plot of Figs. 4.2(a), 4.2(b) demonstrates some of the key distinguishing features of collocated and non-collocated systems. The first is the alternating poles



a b

Figure 4.2: Root locus diagram of (a): mass-1 with all stable branches, (b): mass-2 with unstable branches.

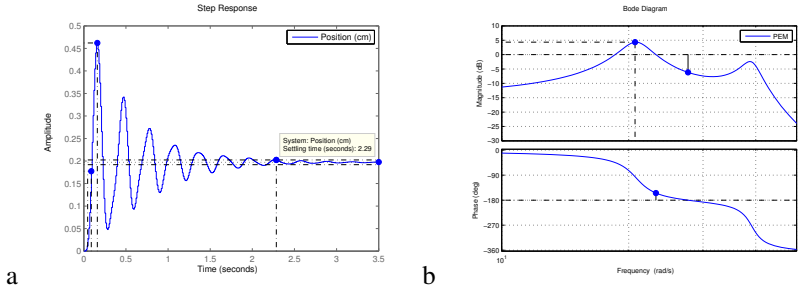
and zeros near the imaginary axis. This is the case for collocated systems i.e. mass-1 and does not exist for non-collocated systems like mass-2. Another principal feature is stability. For mass-1, the stable region is the negative real plane, and therefore this collocated system is and will be stable because the poles stay in the LH plane with increasing gain. However, for mass-2, since the complex conjugate zeros are no longer present, thus the system can very quickly become unstable as the poles travel to the RH plane. The step response of mass-2 is plotted in Fig. 4.3(a) which highlights the oscillatory response and long settling time.

A system identification using prediction error method is performed on the MSD system, and the corresponding frequency response function for the non-collocated case of mass-2 is plotted in Fig. 4.3(b). Multisine excitation signals covering the band of interest were used for the identification with 10 ms sampling time.

Recollect that, a feature of collocated systems like mass-1 is the presence of an anti-resonance between two consecutive resonance frequencies. This means the phase always oscillates within 0^o and -180^o. Furthermore, the zeros of the collocated system are in fact the natural frequencies of the same system with the additional restraint at the collocated sensor and actuator. Since the anti-resonant frequencies are based on the actuator-sensor location, the mass-2 bode plot of Fig. 4.3(b) depicts the absence of the anti-resonance between the same two resonant frequencies and thus there is no 180^o phase lead.

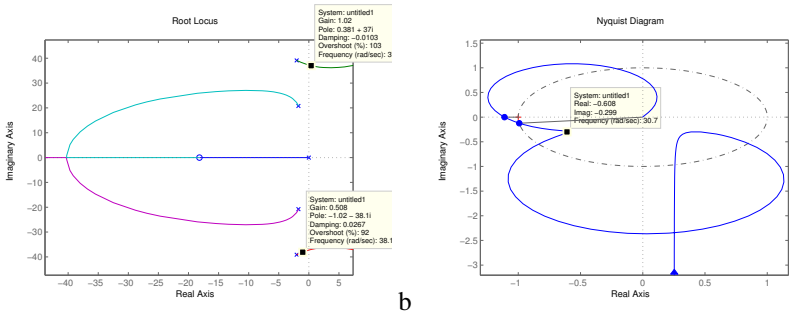
4.4.2 PID control

The objective is to control the position of the second mass which is a non-collocated scenario. A trivial but not at all suitable choice is PID control. The PID-controller possesses three tuning parameters: the proportional gain K_p, the integration time



a b

Figure 4.3: (a): Mass-2 response to step input of 1V, (b): Mass-2 frequency response function



a b

Figure 4.4: (a): Auto-tuner root locus with a pair of overlapped zeros, (b): Nyquist of AH-autotuner open loop

T_i and the differentiation time T_d. Towards tuning, a relay feedback test with relay amplitude r is applied to the process which makes the output oscillate around the set-point with a certain critical amplitude A_c and critical period T_c. Consequently, the critical gain can be computed as $K_c = \frac{4r}{\pi A_c}$. The experiment is performed on the mass-spring-damper and yields :

$$r = 0.5\text{V}, \quad T_c = 0.25\text{s}, \quad A_c = 0.35\text{cm} \tag{4.15}$$

Starting from (4.15), $\mathit{Åström}$ and $\mathit{Hägglund}$ have suggested several ways to compute the PID tuning parameters. We use a tuning method similar to Ziegler-Nichols [73]:

$$K_p = 0.6Kc; \quad T_d = 0.125Tc; \quad T_i = 4T_d \tag{4.16}$$

The resulting PID makes the closed loop unstable. This can be explained by the root locus analysis presented in Fig. 4.4(a).

In our case, the PID has a pole at origin and two overlapped real zeros. This clearly does not suffice to control a system with two pairs of underdamped poles. As marked on this plot, the closed loop corresponding to the tuning parameters (4.16) is clearly unstable, as the underdamped pole pair is already on the right half plane. This is a consequence of a non-collocated system with relative order 3, i.e. 3 zeros at infinity. The root locus of Fig. 4.4(a) has one asymptote along the negative real axis and two asymptotes at $\pm\frac{\pi}{3}$ with unstable branches.

This can be further explained with the Nyquist diagram of Fig. 4.4(b). The auto-tuning methods determine the critical frequency of the plant by the relay experiment and then enforce the open loop i.e. controller*plant to pass through the desired point on the complex plane at the plants critical frequency. However, in our special case with resonances, the open loop frequency response encircles the $(-1, 0)$ point at a frequency greater then the critical frequency, which leads to the instability.

The poorly damped fourth order system gives an unstable closed loop when PID controllers are tuned with auto-tuners or at best oscillatory response with settling time larger than open loop with other tuning techniques [74]. PIDs can neither control the oscillations, because of the poorly damped pole pair near the imaginary axis of the closed loop. A further inclusion of actuator limits can have disastrous consequences on the closed loop, as it is well known that clipping signals can make the system output unbounded [37]. Therefore, a more sophisticated model based control which cannot only counter the system dynamics but also deal with the constraints in a systematic manner, is necessary.

4.4.3 PAMPC applied to position control of MSD

MPC has been applied to vibrating systems, the most relevant for our study would be the work by [75] which deals with the predictive control of a mass spring damper system. However the study excludes analysis based on the structure of the system. The control system design based on the properties of underdamped non-collocated systems has been noted as a challenging problem in [76]. Moreover, the majority of research in vibration control has focused on unconstrained systems [77]. The PAMPC approach presented in 4.2 manages the constraints online by finding optimal penalties after tuning rest of the parameters as in 4.3. Let us demonstrate the efficiency of PAMPC on a simulation of the MSD benchmark system.

Recall that, the objective is to control the position of mass-2 with an input voltage to the motor for fast response with minimum oscillations and overshoot.

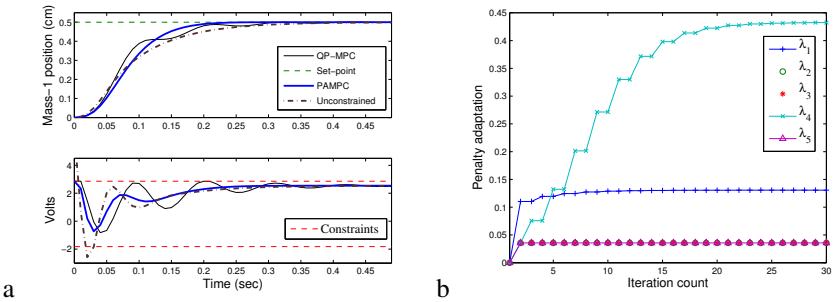


Figure 4.5: (a): MSD control with PAMPC and QP, (b): The penalty adaptations within the first sampling interval.

This system is subjected to the following asymmetric input constraints:

$$-1.82V \leq u \leq +2.86V \tag{4.17}$$

The process model of (4.13) is used with the pole structure shown in Fig. 4.2(b), and the sampling time is same as before i.e. 10ms. A discretization of the system yields $n_B = 3,\ n_A = 4,\ d = 0$. We detail the PAMPC design procedure:

- Use a quadratic cost $V(\Delta U)$ without terminal conditions.
- Fix $N_u = n_A + 1 = 5$, $N_1 = n_B + d + 1 = 4$.
- Compute rounded value of $N_2 = \text{int}(\text{settling time}/T_s) = 28$.
- Obtain a minimum positive integer τ which ensures no overshoot (through simulation), in this case $\tau = 6$.
- Initialize $\Lambda \approx 0 \cdot I_4$.

With these settings the first variant of the PAMPC controller is implemented on the MSD and the results are plotted in Fig. 4.5(a) vis-a-vis a corresponding unconstrained MPC controller. For PAMPC, the settling time is within 0.23s (10 times faster than open loop, see Fig. 4.3(a)) with no overshoot and the constraints are respected as well. The penalty matrix Λ which has $N_u = 5$ entries in its diagonal adapts considerably from initial value to $[0.13, 0.036, 0.036, 0.433, 0.036]^T$, to deliver the required performance with constraint satisfaction. The evolution of Λ within the first sampling interval is plotted in Fig. 4.5(b), which confirms its monotonic increasing property used in theorem 4.3.1. This adapted Λ matrix for the constrained problem does not change any further and thus enables the computation of the polynomial form of the PAMPC controller; the open loop frequency

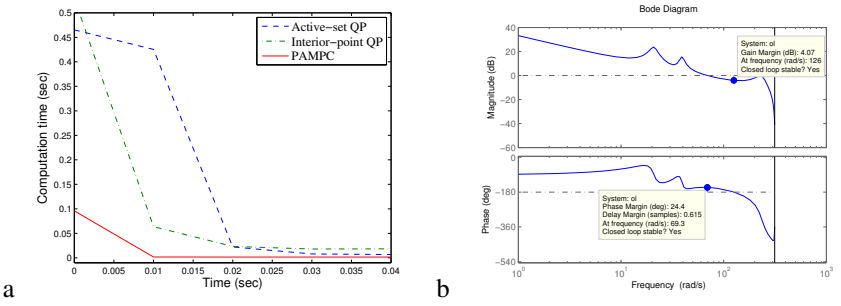


Figure 4.6: (a): Comparison of the computational costs, (b): The open loop frequency response of $PAMPC * plant$

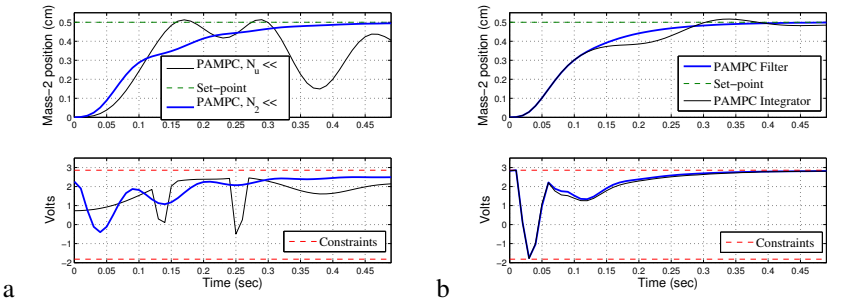


Figure 4.7: (a): The result of suboptimal tuning, (b): Robust control under model uncertainty

response of which is plotted in Fig. 4.6(b) with the phase and gain margins. The modulus margin i.e. the minimum distance between the open loop frequency response curve and the critical point is used as a measure of robustness and is 0.35 for the PAMPC. Further, the closed loop characteristic equation is a Hurwitzian, which guarantees asymptotic stability without terminal cost, terminal penalty, terminal control law, and the closed loop bandwidth of the system is 4.66 Hz.

For the sake of comparison, a MPC controller having more degrees of freedom is now designed with $N_u = 14, N_1 = 1, N_2 = 15, \Lambda = 0 \cdot I_{14}, \tau = 1$ and optimized by a QP solver instead of the penalty adaptation procedure. The constrained optimal solution in this case is plotted in the same Fig. 4.5(a) vis-a-vis PAMPC. PAMPC delivers much better performance for this underdamped non-collocated benchmark system. This is because the penalty matrix Λ which has $N_u = 5$ entries in its diagonal adapts considerably from initial value to deliver the required

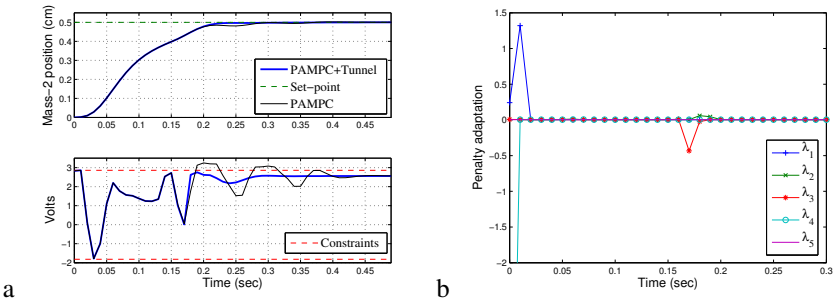


Figure 4.8: (a): Robust feasibility under process disturbances, (b): Evolution of Penalty adaptations till convergence

performance with constraint satisfaction. The QP based MPC however uses the conventional parameterization and can only guarantee constraint satisfaction, but has no means to find the correct penalty matrix and hence ends up with higher settling time and control effort. Computation time is a critical factor for the evaluation a control algorithm especially for fast systems, hence we compared the computation costs of active-set and interior-point QP solvers with PAMPC. The results are plotted in Fig. 4.6(a) for the first few iterations which are relevant due to active constraints. It is clear that PAMPC is at least 5 times faster than the QP solvers.

Further, in one case we deliberately detune the control horizon to unity leaving the rest of the parameters to nominal and in another detune the prediction horizon to half of its original value leaving the rest of the parameters to nominal, to show the effectiveness of our suggested tuning procedure. It can be noted from Fig. 4.7(a) that a mean-level control obtained from $N_u = 1$ induces oscillations and increases the settling time. Similarly, for $N_2 = 14$ case, an increase in oscillations and settling time can be noted from Fig. 4.7(a) compared to Fig. 4.5(a). Next, we consider the case where the model has $\pm 5\%$ uncertainty in terms of the gain, the two natural frequencies and damping ratios. Under these settings, the PAMPC with the same parameters as above but now with the disturbance filter designed as: $C/D = 1/(A.(1 - q^{-1}))$ is considered. The results are plotted in Fig. 4.7(b) which shows the settling time is now just over 0.3s and has zero overshoot. This is then compared to the one where a standard integrator filter is used. Notice, that in this case the oscillations persist in the controlled closed loop.

In a last test, an additive input step disturbance equal to one third of the input range is introduced at 0.15s. In this scenario we keep the above tuning with the improved filter for PAMPC and add the tunneling mechanism from the previous section. The learning function here is just a constant with no memory; the results are illustrated in Fig. 4.8(a). The PAMPC controller maintains the nominal per-

formance with no constraint violations. Notice that in Fig. 4.8(b), the penalties are adapted once again after 0.15s to cope with the step disturbance, before they converge again. This is compared to the controller without tunneling, and it can be clearly seen that it results in serious constraint violations.

4.5 Summary

First, a predictive control strategy PAMPC is introduced which manages constraints by adaptation of weights on control increment. The novelty is that in the case of input constrained nominal repetitive dynamics, the resulting closed-loop system can be made linear which allows for the use of linear system tools to guarantee asymptotic stability of the adapted MPC without terminal conditions through the closed-loop characteristic polynomial. Second, an online constraint tunneling approach is presented towards robust feasibility of the PAMPC controller under process disturbances. None of these schemes need any quadratic programming as the analytical solution is derived.

Next, the control challenges posed by constrained, underdamped, non-collocated mechatronic systems have been highlighted through a mass-spring-damper representative system. The PAMPC is shown to be effective for control of the MSD system, which poses all the mentioned challenges. Extensive simulations are performed which validate the controller under nominal and perturbed settings.

5

Switched Nonlinear Predictive Control

A logical extension to MPC over linear models (which has been largely dealt with in the previous chapters) is to use nonlinear dynamical systems, given the ubiquitousness of nonlinear control problems and the lack of an universally accepted nonlinear control solution [44]. Extending the MPC formulation for constrained linear systems to nonlinear systems is conceptually straightforward but met with practical difficulties. Most of the stability results for the constrained linear systems apply to nonlinear systems without modification (the Lyapunov and invariance techniques considered a general nonlinear state-space representation).

However the implementation is greatly complicated by the computational complexity in finding a globally optimal solution to a non-convex optimization problem. Despite the progress made in the area of nonlinear programming, computational complexity remains a major obstacle for designing a practically implementable nonlinear MPC algorithm [44]. Naturally, the researchers focused on finding a formulation that does not require a globally optimal solution to be found, just a feasible solution, for example in [78]. Once a feasible solution is found, the subsequent solution preserves the feasibility and tries to merely improve the cost. Recently, some progress has been made in developing tools in $C++$ for fast optimization in control [79]. Chen and Allgower [80] presented an approach called quasi-infinite-horizon MPC, where a quadratic terminal penalty corresponding to the infinite horizon cost of linearized system is imposed. Because a terminal constraint is used to force the state to lie within a terminal region, within which the system is stabilized by linear feedback, feasibility alone implies asymptotic stability. However, our focus here, once again, will be on nonlinear MPC without

terminal conditions due to the same reasons as presented in chapter 3.

5.1 Introduction

In fact, the theory presented in chapter 3 on robust persistent feasibility to guarantee practical stability made use of a very general nonlinear system model, and hence all the results which were obtained are generally valid for nonlinear MPC without terminal conditions. The paper [81] provides an exposé to the computation of control invariant sets for nonlinear systems. More so, since the results on robust persistent feasibility are based on set invariance and are independent of the cost function or the optimality of the solution, therefore the practical stability proofs derived in chapter 3 are valid for nonlinear MPC where optimality of the solution cannot be guaranteed. Therefore the conditions $D2 - D5$ of chapter 3 are satisfied for nonlinear MPC. If the nonlinear models are already in state-space form, then condition $D1$ is fulfilled by default, and if not a nonlinear state-space can be derived from the input-output representation by using similar rules as in the linear systems case (refer 2.4.1).

Remark 5.1.1. *The input-output model in the nominal form is represented as:* $y(t) = f_y(y(t-1), y(t-2), \ldots, u(t-1), u(t-2), \ldots)$. *An equivalent state-space representation would be with the state vector as:* $x(t) = [y(t), y(t-1), \ldots u(t-1), u(t-2), \ldots]^T$. *Correspondingly there would be one big equation for the first state update* $x_1(t+1) = f(x(t), u(t))$ *with the rest updated through a shift register (i.e. appear as equalities to their previous values). The input, output constraints* $\mathbb{U}, \mathbb{Y}$ *are mapped to the state constraint* $[\mathbb{Y} \times \mathbb{Y} \times \ldots \times \mathbb{U} \times \mathbb{U} \times \ldots]$.

The reverse direction is a hard problem, i.e. to come up with a nonlinear input-output representation from a nonlinear state-space model. A few ideas in this direction would be to use state elimination technique (not guaranteed to work in all cases) [82] or to perform a nonlinear auto-regressive exogenous system identification over the state-space model [83].

Amongst the NMPC techniques, the nonlinear EPSAC (NEPSAC) [34] is a pragmatic choice for the industry as it is generally valid for any nonlinear system and is computationally tractable. However, the convergence of the NEPSAC algorithm remained an open problem for around two decades. One of the fundamental contributions of the thesis is to formally present the proof of convergence.

Many nonlinear systems in the mechatronic domain appear as piecewise affine systems or systems with switching dynamics [44]. Moreover, any nonlinear system, in principle can be expressed as a piecewise affine (PWA) system with bounded uncertainty [84].

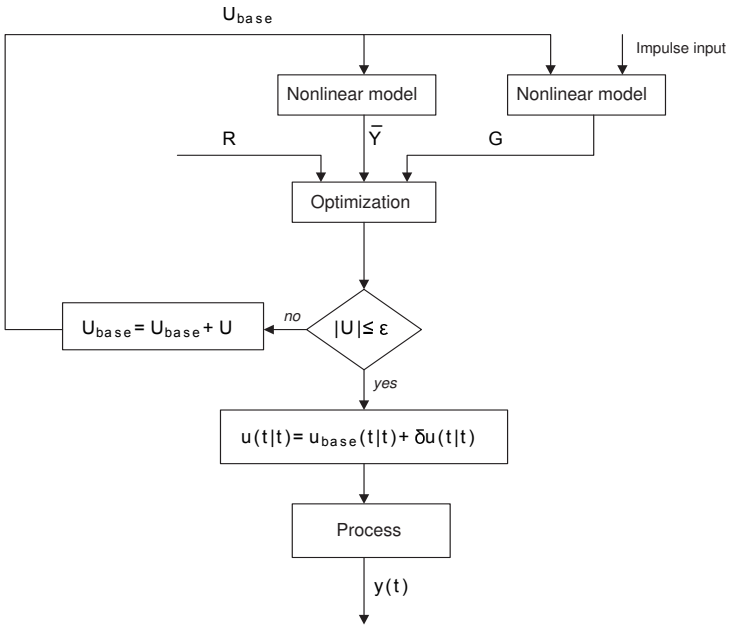


Figure 5.1: NEPSAC flowchart (The base input trajectory U_{base} is used to compute the base response $\bar{Y}$ which is added to the convolution of impulse response matrix G with optimal U (computed by optimizer) to realize target reference R. Then, U_{base} is incremented with U and the steps repeated until U_{base} converges to a local optimal, following which the first element of the update $U_{base} + U$ giving $u(t|t) = u_{base}(t|t) + \delta u(t|t)$ is applied to the process and output $y(t)$ is measured.

Hence, we extend our stability certification mechanisms to MPC of PWA systems without terminal conditions which could then be used to certify the corresponding nonlinear MPC without terminal conditions. Next to it, a two-level NMPC architecture is developed to control switched dynamical systems.

5.2 Nonlinear MPC (NEPSAC) with guaranteed convergence

The nonlinear EPSAC i.e NEPSAC is an iterative execution of the EPSAC algorithm until convergence to the locally optimal control effort within each sampling period [34]. The advantages of NEPSAC over classical NMPC strategies are (1) direct use of the nonlinear prediction model, avoiding local linearization; (2) intrinsic capability to deal with repetitive disturbances; and (3) ease of implementation. All these are feasible because NEPSAC does not demand significant modification

of the basic linear EPSAC concept which is computationally simple and fast, making it suitable for real-time online control.

When a nonlinear system $f[.]$ is used for $\hat{y}(t)$, the superposition of (2.4) is still valid only if the term $y_{optimize}(t+k|t)$ is small enough compared to $\bar{Y}(t+k|t)$, which is true if $\delta u(t+k|t)$ is small, with $k \in [1, \ldots, N_2]$. This can in turn only happen if $u_{base}(t+k|t)$ is close to the optimal $u^*(t+k|t)$. Notice that, in the nonlinear case the step responses are different for each operating point, and hence unlike the linear controller, NEPSAC requires re-computation of the G matrix at every sampling instant. However, the exact values of the matrix are not critical, since the effect of G would gradually disappear. This can be realized iteratively by going over the following steps for each sampling interval:

1. Initialize $u_{base}(t+k|t)$ to the previous optimal control sequence i.e. $u^*(t+k|t-1)$.
2. Compute $\delta u(t+k|t)$ using the linear EPSAC procedure of section 2.3.
3. Calculate the corresponding $y_{optimize}(t+k|t)$ (same as the linear case, refer 2.3) and compare it to $\bar{Y}(t+k|t)$.
   - In case the difference is not small enough, redefine $u_{base}(t+k|t)$ as $u_{base}(t+k|t)+\delta u(t+k|t)$ and return to step 2. The underlying concept is that this optimal input can act as a second estimate for the nonlinear system.
   - In case the difference is within certain tolerance, $u^*(t) = u_{base}(t|t) + \delta u(t|t)$ is the locally optimal control for the current sampling instant.

This algorithm is graphically illustrated in Fig. 5.1, where $R, \bar{Y}, U_{base}$ are now in the vector form of the signals $r(.|.), \bar{Y}(.|.), u_{base}(.|.)$ introduced before. The algorithm after convergence results in the locally optimal control for a given nonlinear system. The number of iterations required depends on how far the optimal control is from its prior value.

Theorem 5.2.1. *The NEPSAC nonlinear MPC controller without terminal conditions converges to a local optimum, provided the value of the control penalty forces a monotonic decrease of the cost function through every iteration.*

Proof. The NEPSAC algorithm recursively computes the locally optimal base response through the locally optimal base input vector. Therefore, the algorithm starts by initializing the base input vector, to U_{base} resulting in output vector $\bar{Y}$. Then the corresponding quadratic cost function takes the form (without terminal conditions):

$$V(U_{base}) = (R - \bar{Y})^T \cdot (R - \bar{Y}) \tag{5.1}$$

Now, the problem is to find the locally optimal U^*_{base} which minimizes the above cost function. The standard way to tackle such a nonlinear optimization problem is by gradient descent. Hence the next approximation of the optimal U^*_{base} should be the initial U_{base} plus a perturbation vector δU. To approximate δU, we need an expansion of the nonlinear system around $\bar{Y}$. This can be done by Taylor's series approximation to the first derivative term:

$$Y \approx \bar{Y} + J \cdot \delta U \tag{5.2}$$

where J is the Jacobian of the nonlinear system $f[.]$. It is here, that NEPSAC uses the clever trick, that the Jacobian is nothing other than the matrix G of impulse and step responses. Hence the Taylor expansion can equivalently be written as:

$$Y \approx \bar{Y} + G \cdot \delta U \tag{5.3}$$

At the minimum of the sum of squares, the gradient of V with respect to δU will be zero. The above first order approximation gives:

$$V(U_{base} + \delta U) \approx (R - \bar{Y} - J \cdot \delta U)^T \cdot (R - \bar{Y} - J \cdot \delta U) \tag{5.4}$$

Taking the derivative with respect to δU and setting the result to zero gives:

$$\delta U = (J^T \cdot J)^{-1} J^T (R - \bar{Y}) = H^{-1} J^T (R - \bar{Y}) \tag{5.5}$$

Note that the Hessian i.e. the second derivative is in fact, $H = J^T \cdot J$ and this is called the Newton method of optimization. Here is the second trick of NEPSAC as the Hessian can simply be re-written as $H = G^T \cdot G$, as we established through Taylor series that $J = G$. Therefore the Hessian computation comes as free again from the G matrix. This exactly leads to the NEPSAC iterate:

$$\delta U = (G^T \cdot G)^{-1} G^T (R - \bar{Y}) \tag{5.6}$$

This formulation however has the disadvantage that convergence cannot be guaranteed, partly because a requirement that $G^T \cdot G \succ 0$ may not be always true. Therefore Levenberg-Marquardt algorithm [85] proposed a solution to this problem, by adding a so called damping term to the Hessian matrix: $\Lambda = \lambda \cdot \mathcal{I}$ where $\mathcal{I}$ is the identity matrix and $\lambda > 0$ is a scalar called the damping factor, which is to be adjusted to guarantee a cost decrease. Here comes the third and final trick of NEPSAC: the damping factor λ turns out to be exactly identical to the control penalty by re-writing the cost as:

$$V(U_{base} + \delta U) \approx (R - \bar{Y} - J \cdot \delta U)^T \cdot (R - \bar{Y} - J \cdot \delta U) + \delta U^T \cdot \Lambda \cdot \delta U \tag{5.7}$$

Now the iterate, which is obtained by equating the derivative to zero, is exactly the same as Levenberg-Marquardt algorithm, with $J = G$:

$$\delta U = (G^T \cdot G + \Lambda)^{-1} G^T (R - \bar{Y}) \tag{5.8}$$

In the next step, of course the initial solution is then updated to $U_{base} = U_{base} + \delta U$ and this NEPSAC algorithm which we have just shown is equivalent to the Levenberg-Marquardt optimization is guaranteed to converge to a locally optimal solution because of the damping λ that forces a monotonic cost decrease through every iteration. □

Remark 5.2.1. *As the above theorem suggests the choice of λ is critical to guarantee convergence, a heuristic way of finding the right λ is, to first initialize it and then:*

1. *Compute a NEPSAC iterate.*
2. *Evaluate the cost function as a result of the optimization.*
3. *If the cost has increased, increase λ by a factor of* 10.
4. *Go to step 1.*

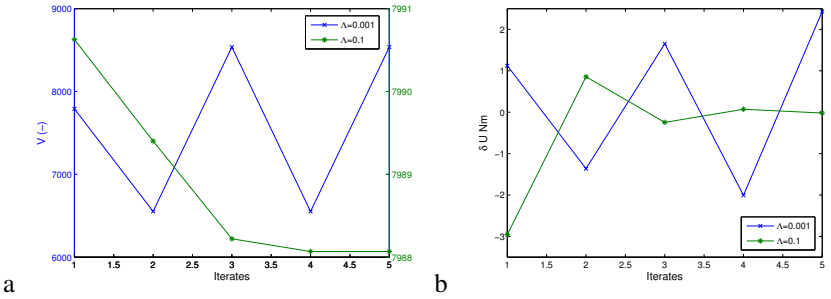
Lemma 5.2.1. *A more robust way of getting a correct λ is to replace it with:* $\Lambda = \lambda \cdot diag(H) = \lambda \cdot diag(J^T \cdot J) = \lambda \cdot diag(G^T \cdot G)$.

Proof. The scaling of each component of the gradient $J^T = G^T$ by the Hessian matrix $H = J^T \cdot J = G^T \cdot G$ results in larger movement along the directions where the gradient is smaller. Since the Hessian is proportional to the curvature of V, (5.8) implies a large step in the direction with low curvature (i.e., an almost flat terrain) and a small step in the direction with high curvature (i.e., a steep incline). □

Remark 5.2.2. *Note that, with control horizon $N_u = 1$, the constrained solution equals to the unconstrained one, passed through a saturation, therefore no modification is necessary to the theorem 5.2.1. However, for longer control horizons, and in the presence of input, output constraints, the step where the unconstrained optimal solution is found i.e. (5.8) has to be replaced by the active sets quadratic programming constrained optimal solution. The rest of the steps in the proof remains the same.*

5.3 Example: Longitudinal Flight Control

The complete set of equations describing the longitudinal flight dynamics were introduced in (3.31) and a linear MPC pitch controller was presented during automatic take-off. Now, we are interested in controlling the height of the aircraft i.e. z_e to a target $z_r = 60\text{m}$ using the constrained moment $||M||_\infty \leq 100\text{Nm}$ as the control input. The associated equations of (3.31) are non-linear with state multiplications, saturation and trigonometric non-linearity.



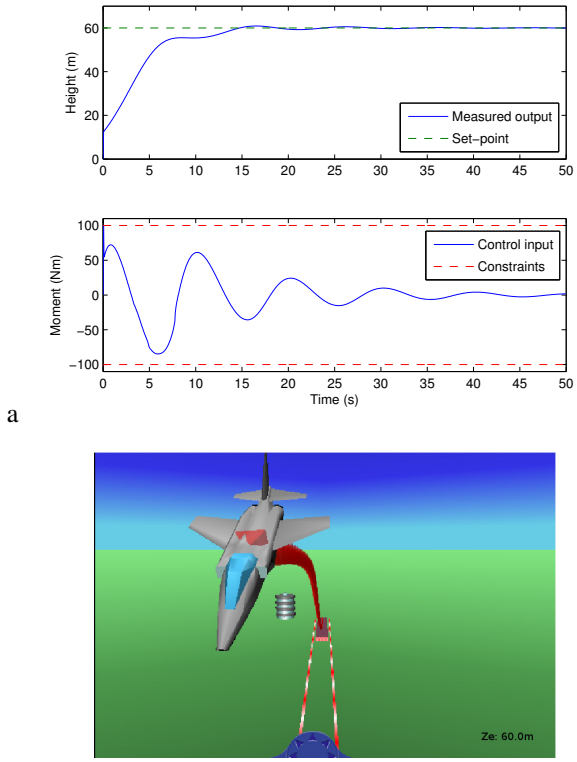
a b

Figure 5.2: Illustration of the impact of control penalty on the convergence of (a): cost function, (b): NEPSAC iterates.

Therefore, a NEPSAC nonlinear MPC controller is now designed with $N_u = 1, N_2 = 100$ in order to obtain stabilizing response. The closed-loop as usual is sampled at $T_s = 0.05$s and all the model parameters have the same values as in section 3.5.2. The impulse/step matrix should be re-computed at certain operating points which gives improved computational cost. An automatic take-off is performed with full throttle $F_x = 3000$N and after 8s when the aircraft is airborne, the controller is switched on. The main purpose of this example is to validate theorem 5.2.1 i.e. to illustrate that adapting the penalty is necessary to obtain a monotonic decrease in the cost (V) as well as a reduction in NEPSAC iterates (δU) within every sampling interval. So, at first the control penalty is initialized to an arbitrary value of $\lambda = 0.001$. It turns out that the associated costs of the NEPSAC iterates within the same sampling time do not decrease monotonically, refer Fig. 5.2(a). The resulting NEPSAC iterates as one would expect do not converge either and one such case is plotted in Fig. 5.2(b).

Now, we apply the algorithm of remark 5.2.1 and increase the penalty by a factor of 10 to $\lambda = 0.01$ and recompute the controller. Even this time, the in iteration cost does not decrease which prompts a further ten-fold increase of penalty to $\lambda = 0.1$. The NEPSAC controller with $\lambda = 0.1$ does enforce a monotonic decrease of the cost function with every NEPSAC iteration for most of the sampling intervals. In those few instances, when the cost still increases, an infinite penalty is imposed or in other words the previous control is maintained. This penalty adaptation strategy also results in the convergence of the NEPSAC control iterates. The iterations within a representative sampling time illustrating the monotonic decrease of the cost and convergence of NEPSAC iterates are plotted in Figs. 5.2(a), 5.2(b) respectively.

The performance of the NEPSAC controller is satisfactory with a settling time of 20s and the input constraints are obeyed. These results are plotted in Fig. 5.3(a).



a

b

Figure 5.3: (a): Nonlinear MPC control of aircraft height, (b): A height controlled automatic take-off scenario.

A snapshot (not to scale) of the corresponding animation can be seen in Fig. 5.3(b).

5.4 Piecewise Affine Systems

PWA systems can model a large number of physical processes, such as systems with static nonlinearities, and can approximate nonlinear dynamics via multiple linearizations at different operating points [86]. Piecewise affine systems in the

nominal case are described by the state-space equations:

$$x(t+1) = \mathcal{A}_i \cdot x(t) + \mathcal{B}_i \cdot u(t) + f_i, \tag{5.9a}$$
$$y(t) = \mathcal{C}_i \cdot x(t) + g_i, \tag{5.9b}$$
$$M_{x_i} \cdot x(t) + M_{u_i} \cdot u(t) \leq M_i, \quad i \in \mathcal{I} \tag{5.9c}$$
$$if \quad [x^T(t), u^T]^T \in \chi_i \tag{5.9d}$$

whereby the dynamics (5.9a) with associated constraints (5.9c) are valid in the polyhedral set defined in (5.9d). The set $\mathcal{I}$ represents all possible dynamics. If all the $f_i = 0$, then the system is said to be piecewise linear (PWL) and if all $\mathcal{A}_i = 0, \mathcal{B}_i = 0$, then the system is piecewise constant. Henceforth, we will abbreviate (5.9a) and (5.9c) with $x(t+1) = f_{PWA}(x(t), u(t))$. Note that there is no general description of PWA systems in the input-output space and can be on a case-by-case basis be transformed to the above state-space form by considering the state vector $x(t) = [y(t), y(t-1), \ldots u(t-1), u(t-2), \ldots]$ and the constraints mapped through $\chi_i = [\mathbb{Y}_i \times \mathbb{Y}_i \times \ldots \mathbb{U}_i \times \mathbb{U}_i \times \ldots]$ for each respective region i. An illustrative example is given in the next section.

The MPC problem 3.1.2 is restated below for the PWA case (without terminal conditions):

$$\begin{aligned} V^*_{N_u}(x(t)) = \min_U \sum_{k=N_1}^{N_2} (x^T(t+k|t) \cdot C^T C \cdot x(t+k|t)) \\ + \lambda \sum_{k=0}^{N_u-1} (u^T(t+k|t) \cdot u(t+k|t)) \\ subj.\ to \quad M_{x_i} \cdot x(t) + M_{u_i} \cdot u(t) \leq M_i, \quad if \quad [x^T(t), u(t)^T]^T \in \chi_i, \\ i \in \mathcal{I}, \quad \forall k \in [N_1, N_2] \end{aligned} \tag{5.10}$$

For the invariant set computations for the PWA systems, note the basic difference in the definition of the pre set:

Definition 5.4.1. *The **pre set for PWA systems** is stated as:*

$$\begin{aligned} Q^{PWA}(\mathbb{X}) \triangleq \{x(t) \in \mathbb{R}^n | \exists u(t) \in \mathbb{U}, x(t+1) \in \mathbb{X} : \\ x(t+1) = f_{PWA}(x(t), u(t))\} \end{aligned} \tag{5.11}$$

The i-step **controllable set for PWA systems**, $K_i^{PWA}(\mathbb{X}, \mathbb{T})$ and i-step **tunnel set for PWA systems**, $L_i^{PWA}(\mathbb{X})$ have exactly the same definition as before, but use the PWA dynamics and their computation is based on $Q^{PWA}(\mathbb{X})$ through the algorithm 1.

Corollary 5.4.1. *Now, the feasibility set X_F^{PWA} is not necessarily convex, but can still be described by the union of convex polyhedra:*

$$X_F^{PWA}(\mathbb{X}, N_u, N_2) = K_{N_u}^{PWA}(\mathbb{X}, L_{N_2-N_u}^{PWA}(\mathbb{X})) \tag{5.12}$$

Proof. The feasibility set is the same as that derived in theorem 3.3.1 and hence the same proof applies. The fact that it can be an union of convex polyhedra is a direct consequence of the state constraints now being an union of convex polyhedra. □

Remark 5.4.1. *The guidelines for stabilizing horizons without terminal conditions derived in 3.3.1 are all valid for the PWA case, except that we use the respective controllable set and its determinedness index $i^*_{K^{PWA}}$ and the tunnel set and its determinedness index $i^*_{L^{PWA}}$ and $N^*_{u^{PWA}}, N^*_{2^{PWA}}$ as the horizons which deem the feasible set invariant. Then the three algorithms presented in section 3.3.1 apply to give the stabilizing horizons N_u, N_2.*

A posteriori certification of stability is through the following test:

Corollary 5.4.2. *The MPC regulator that solves PWA problem 5.10 is persistently feasible iff $\forall i \in \mathcal{I}$:*

$$\begin{aligned} &\mathcal{A}_i((\chi_i \cap X_F^{PWA}(\mathbb{X}, N_u, N_2)) \oplus \mathcal{B}_i \cdot \mathbb{U} \oplus f_i) \\ &\quad \cap X_F^{PWA}(\mathbb{X}, N_u - 1, N_2 - 1) \subseteq X_F^{PWA}(\mathbb{X}, N_u, N_2) \end{aligned} \tag{5.13}$$

Proof. The proof is the same as that of 3.3.6, except that the reach set of the feasibility set has been explicitly computed by mapping it through the PWA dynamics in the LHS to: $\mathcal{A}_i((\chi_i \cap X_F^{PWA}(\mathbb{X}, N_u, N_2))$. □

Remark 5.4.2. *Note that, from corollary 3.3.2, for the special case with $N_u = 1$, $X_F^{PWA}(\mathbb{X}, N_u - 1, N_2 - 1) = L^{PWA}_{N_2 - N_u}(\mathbb{X})$.*

Stability in the practical Lyapunov sense follows, from persistent feasibility, which is independent on the optimality of the solution. A less conservative test is checking for recursive feasibility which though requires the knowledge of the optimal control law. Therefore, an explicit solution to the problem 5.10 is to be computed, with the only difference this time being it is over a PWA partition of the feasibility region. Therefore, intuitively, a mp-QP problem needs to be solved for each PWA partition, followed by merging the partitions, which would clearly still produce a PWA controller. This is done using dynamic programming i.e. backwards through the control horizon, the algorithm is briefly presented at a high level here:

Note that the algorithm is initialized with feasibility set equal to the union of all the χ_i over which the problem is defined as there is no terminal constraint, with $|X^{PWA}_{F_{k+1}}|$ denoting its cardinality and s the number of modes. The resulting PWA controller with feasibility set X_F^{PWA} may have multiplicity i.e. one partition with more than one control laws, which can be resolved by picking the one which minimizes the associated stage cost. This algorithm is used as a mere tool to check recursive feasibility, hence for more details in to the algorithm itself, please refer [86].

For $k = N_u - 1, \ldots, 0$
For $j = 1, \ldots, |X^{PWA}_{F_{k+1}}|$
For $i = 1, \ldots, s$
Solve the mp-QP (using theorem 2.4.3),
$S(k,j,i) = \min_{u(t+k)} \; x^T(t+k|t) \cdot C^T C \cdot x(t+k|t) + u^T(t+k|t) \cdot u(t+k|t)$
$+V^*_{k+1}(\mathcal{A}_i \cdot x(t) + \mathcal{B}_i \cdot u(t) + f_i)$
Subj. to: $\mathcal{A}_i \cdot x(t) + \mathcal{B}_i \cdot u(t) + f_i \in X^{PWA}_{F_{k+1}}, M_{x_i} \cdot x(t) + M_{u_i} \cdot u(t) \leq M_i,$
$[x^T(t+k|t), u(t+k|t)^T]^T \in \chi_i$
End
End
End

algorithm 7: The explicit solution to PWA MPC

Corollary 5.4.3. *A **positively invariant subset for PWA systems**, $O^{PWA}(X^{PWA}_F)$ of the feasibility set X^{PWA}_F induces recursive feasibility and practical Lyapunov stability, when the above derived explicit controller is applied in receding horizon sense.*

Proof. The proof follows from infinite time boundedness of the trajectories, by definition of invariance. □

The positive invariant set (not necessarily convex in the PWA case) can be found, by following the algorithm 6. The online implementation involves identifying the partition which contains the current state $x(t)$, and invoking the corresponding control law, which are available as a result of the above algorithm. One problem of the approach is that the number of regions grows quickly with the control horizon size and number of partitions of the model. This is another motivation for us to consider very short control horizons.

5.5 Example: Car

Assume a frictionless car moving horizontally on a hill with different slopes, as illustrated in Fig. 5.4. Dynamics of the car is driven by Newton's laws of motion (in the input-output form):

$$m \cdot \ddot{y}(t) = u(t) - m \cdot g \cdot \sin\alpha \tag{5.14}$$

where y denotes horizontal position in m, α is the slope of the road and $g = 9.81\mathrm{m/s^2}$ is the acceleration due to gravity and u the force applied in N. Assume the mass $m = 1\mathrm{kg}$ and discretize the system with sampling time $T_s = 0.5\mathrm{s}$, we obtain the following affine system in the state-space form using the nominal

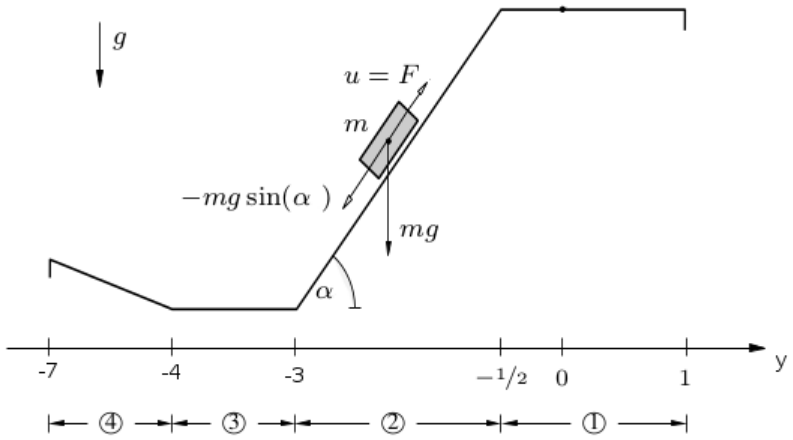


Figure 5.4: Car moving on a PWA hill [1].

transformations introduced in chapter 2:

$$x(t+1) = \begin{bmatrix} 2 & -1 & 0.25 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(t) + \begin{bmatrix} -2.45 \cdot \sin\alpha \\ 0 \\ 0 \end{bmatrix} \tag{5.15}$$

$$y(t) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} x(t) \tag{5.16}$$

where the state vector is $x(t) = [y(t), y(t-1), u(t-1)]^T$. It can be seen that speed of the car depends only on the force applied to the car (manipulated variable u) and slope of the road α, which is different in different sectors of the road. In particular we have:

$$\begin{aligned} &\text{Sector 1}: \quad y(t) \geq -0.5\text{m} \Rightarrow \alpha = 0^o \\ &\text{Sector 2}: \quad -3\text{m} \leq y(t) \leq -0.5\text{m} \Rightarrow \alpha = 10^o \\ &\text{Sector 3}: \quad -4\text{m} \leq y(t) \leq -3\text{m} \Rightarrow \alpha = 0^o \\ &\text{Sector 4}: \quad y(t) \leq -4\text{m} \Rightarrow \alpha = -5^o \end{aligned} \tag{5.17}$$

Substituting slopes α from (5.17) to (5.15), we obtain 4 tuples $[\mathcal{A}_i, \mathcal{B}_i, f_i, \mathcal{C}_i]$ for $i \in 1, \ldots, 4$. Furthermore, we need to define parts of the state-input space where

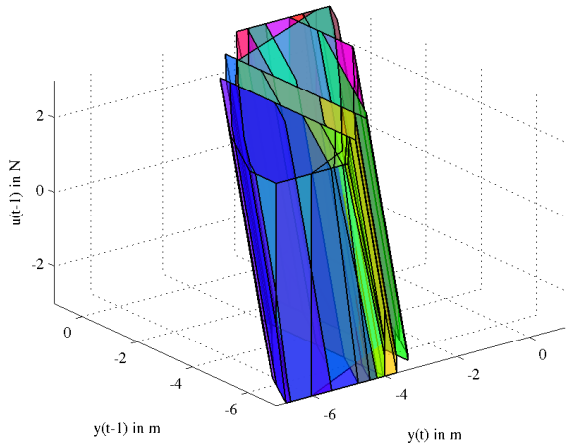


Figure 5.5: Persistent feasibility test: LHS of (5.20) [lighter box] ⊆ RHS of (5.20) [darker box].

each dynamics is active. We do this using the corresponding sectors as follows:

$$\begin{aligned}
\text{Sector 1}: &\quad \begin{bmatrix}-1 & 0 & 0\end{bmatrix} x(t) \leq 0.5 \\
\text{Sector 2}: &\quad \begin{bmatrix}1 & 0 & 0\\ -1 & 0 & 0\end{bmatrix} x(t) \leq \begin{bmatrix}-0.5\\ 3\end{bmatrix} \\
\text{Sector 3}: &\quad \begin{bmatrix}1 & 0 & 0\\ -1 & 0 & 0\end{bmatrix} x(t) \leq \begin{bmatrix}-3\\ 4\end{bmatrix} \\
\text{Sector 4}: &\quad \begin{bmatrix}1 & 0 & 0\end{bmatrix} x(t) \leq -4
\end{aligned} \tag{5.18}$$

Further, the following global constraints are imposed: $-7\text{m} \leq y(t) \leq 1\text{m}$ on output displacement and $-3\text{N} \leq u(t) \leq 3\text{N}$ as limit on input force, which can be transformed to the state constraint: $[-7,-7,-3]^T \leq x(t) \leq [1,1,3]^T$.

The PWA MPC controller to be verified is designed with control horizon $N_u = 1$ and prediction horizon $N_2 = 10$ and no terminal conditions. The nominal feasibility set for this PWA system is computed as per corollary 5.4.1 to:

$$X_F^{PWA}(\mathbb{X}, 1, 10) = K_1^{PWA}(\mathbb{X}, L_9^{PWA}(\mathbb{X})) \tag{5.19}$$

after the computation of the 9-steps tunnel set $L_7^{PWA}(\mathbb{X})$ and the 1-step controllable set to it i.e. $K_1^{PWA}(\mathbb{X}, L_7(\mathbb{X}))$. Next, the reach set of the feasibility set and tunnel set for the PWA system are computed and to check nominal persistent feasibility of the MPC problem, we make use of corollary 5.4.2 and remark 5.4.2:

$$\mathcal{A}_i((\chi_i \cap X_F^{PWA}(\mathbb{X}, 1, 10)) \oplus \mathcal{B}_i \cdot \mathbb{U} \oplus f_i) \cap L_9^{PWA}(\mathbb{X}) \subseteq X_F^{PWA}(\mathbb{X}, 1, 10) \tag{5.20}$$

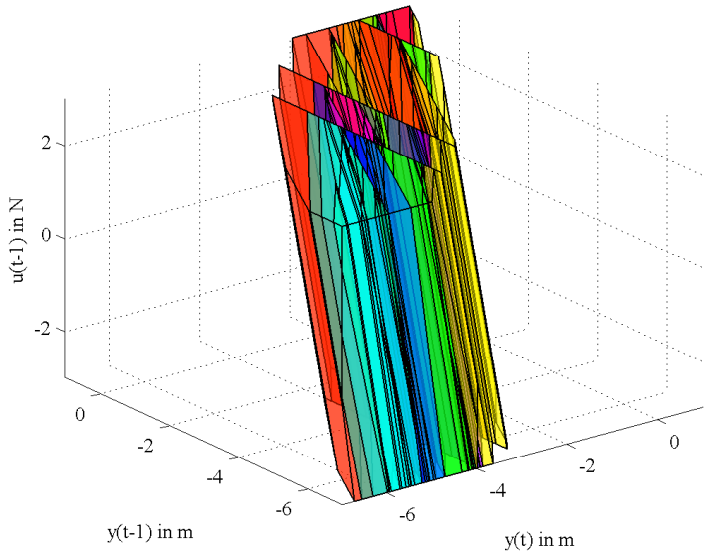


Figure 5.6: Recursive feasibility test through construction of $O^{PWA}(X_F^{PWA})$.

Indeed, this test is fulfilled as can be seen graphically in Fig. 5.5 and thus a certificate of practical nominal stability can now be issued to this PWA MPC controller.

Now, we want to compare the above test with the PWA test of recursive feasibility. Since, we have an intuition of it being less conservative, an PWA MPC controller is designed with much smaller prediction horizon $N_2 = 4$ and same control horizon $N_u = 1$ and no terminal conditions with $\Lambda = \mathcal{I}$. So, an explicit solution to the PWA MPC is constructed through algorithm 7 and then a positively invariant subset $O^{PWA}(X_F^{PWA})$ of the feasibility set is computed by using corollary 5.4.3, thereby guaranteeing practical stability. The positive invariant set has 161 partitions and is plotted in Fig. 5.6.

5.6 Switched Nonlinear Systems

The switched nonlinear dynamical systems are characterized (and hence different from PWA systems) in the following ways: (1) the entire system can be broken down into a series of linear or nonlinear interconnected systems (as against only linear for PWA) (2) and the state or input-output mapping is not preserved i.e. the meaning of the state or input-output vectors may change, in order and dynamics (as opposed to PWA where it is preserved).

For repetitive mechatronic applications with hard nonlinear transitions, a single nonlinear control algorithm is complex and difficult to develop. For an optimal control of switched systems, the optimal switching sequence as well as the optimal input are to be found and solving such a problem is generally very difficult [87]. To reduce the complexity, we propose to use separate controllers for each phase. This makes the identification easier as it now suffices to develop a model for each phase separately instead of a global model. As a further simplification, we only consider tracking controllers, as these are typically easier to develop and implement than controllers with more complex cost functions. These tracking controllers are included in the low level of the proposed two-level control scheme.

On the high level iterative learning algorithms are then included that learn the parameters of the parameterized references for the low level. These are added to ensure the following:

1. Transition between the different phases happens in a smooth manner.

2. Since the specifications cannot readily be translated into references, these learning algorithms also learn references corresponding to the optimal performance.

3. The optimal reference trajectories may vary with changes in the operating conditions, to which these algorithms can automatically compensate for these changes as well.

At the low level, if the constituent process is linear, then a linear MPC controller suffices, however if the constituent process is nonlinear, a nonlinear MPC controller may be necessary. The theory of these constituent MPC and NMPC controllers including stability and convergence properties have been adequately dealth with in chapters 2 to 4 and section 5.2. The high level control mechanism is presented next. Note that, the two-level NMPC design philosophy is strongly inspired from solving the engagement control problem of wet-clutches which is a typical example of a switched nonlinear system. Hence, section 5.7 gives a very comprehensive account of its application.

Two-level Control

For the high level reference adaptation, process knowledge is used to select a profile or procedure that allows to construct the reference trajectories from a few discrete parameters, say $w_i(k)$, with i the variable index and k the iteration number. After a set of parameters and corresponding references are selected, the low level tracking controllers are allowed to operate. Next, the obtained performance is assessed and used by the high level controller to update the parameters using iterative learning algorithms, and new references are calculated. The parameter update is

based on the difference between the desired and the measured performance indices PI_i^d, PI_i^{est} respectively with gain η_i as follows:

$$w_i(k+1) = w_i(k) + \eta_i \cdot (PI_i^d(k) - PI_i^{est}(k)) \tag{5.21}$$

This process is repeated until the parameters of the references converge. At the low-level either MPC, NMPC controllers are deployed and the references are adapted after every iteration after computation of the performance index. The algorithm 8 summarizes the learning process (variables in capitals represent vectors in time of the corresponding signals over each iteration of the closed-loop control).

Data: (Non)linear models, Tuned (N)MPCs
Data: Initialize parameters W, reference R
Result: Adapted R
while *true* **do**
  apply input $U(k)$ to system and measure output $Y(k)$ NMPC controller computes U(k+1) Evaluate performance, update parameters $W(k+1)$ Compute new reference $R(k+1)$ from $W(k+1)$ $k++$
end

algorithm 8: Automated two-level learning control

The two-level control scheme can also be applied to other mechatronic applications performing repetitive operations, where reference trajectories cannot easily be obtained. This can be due to specifications that can not easily be translated into reference trajectories and the optimal references changing with operating conditions. Other potential applications are control problems where different controllers need to be coordinated for different phases of a process.

5.7 Test case: Clutch Engagement

Note that, this section is a result of collaborative work done jointly with FMTC (Flanders' Mechatronics Technology Centre), KUL (Catholic University of Leuven), VUB (Free University of Brussels) under the framework of the IWT (agency for Innovation by Science and Technology) SBO (Strategic Basic Research) funded LeCoPro (Learning Control for Production machines) project. In particular, high-level learning algorithms developed in 5.7.2, system identification in 5.7.3 are due to KUL and VUB respectively. The experimental setup including hardware and interfacing in 5.7.4 is due to FMTC.

Wet-clutches are commonly used in automatic transmissions for off-highway vehicles and agricultural machines to transfer torque from the engine to the load. By dis-engaging one clutch and engaging another, different transmission ratios can be obtained. These machines are operated through several years and under varying

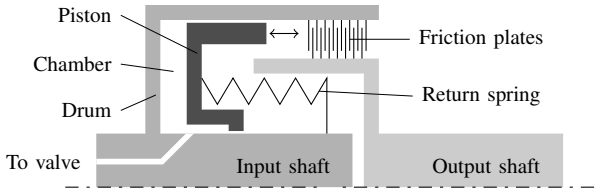


Figure 5.7: Schematic overview of a wet-clutch and its components.

environmental conditions such that clutches undergo significant amount of wear and tear. Operators however always expect a fast and smooth response without system oscillations induced by poor engagement performance. These expectations, combined with the intrinsic time varying and nonlinear dynamics, make the wet-clutch control a challenging industrial problem [88].

Contrary to wet-clutches, the modeling and control of dry-clutches, has received considerable attention in research but only considering a stick-slip hybrid model for analysis. A slip control using linear quadratic regulator with force on clutch piston as input is developed in [89]. Explicit Model Predictive Control (MPC) is used in [90] over a piece-wise affine model for slip control in dry-clutches, and [91] has concluded that an online MPC scheme for clutch control is not practically implementable due to high computation costs.

However, research in advanced identification and control of wet-clutches has not reached the necessary maturity for production objectives. The electro-hydraulic wet-clutches, as opposed to stick-slip in dry-clutches, have dynamics in both fill and slip phases. A fill phase control based on piston position measurement feedback is carried out in [92] and through pressure control, requiring a pressure sensor, in [93].

5.7.1 Wet-clutch

Typically, a clutch contains two sets of friction plates, one that can slide in grooves on the inside of the drum, and another that can slide in grooves on the outgoing shaft. As illustrated in Fig. 5.7, the torque can be transferred between the shafts by pressing both sets together with a hydraulic piston, i.e. by sending a current signal to the servo-valve in the line to the clutch (i.e. the manipulated variable is current).

The objective is to obtain a good clutch engagement i.e. a fast response without significant drivetrain vibrations that can be uncomfortable to the operator. This can be realized by first quickly moving the piston towards the plates, as there is no significant torque transfer before the piston makes contact, after which the piston is gently pressed into the plates and the force is built up gradually, such that any

torque spikes and vibrations of the drivetrain are avoided. This is a challenging control task as the two objectives are conflicting and compounded by the absence of sensors measuring the piston position.

For this task, the measurements that are available are the pressure in the clutch and the rotational speeds of the shaft. From these speeds, the slip can also be calculated, which is defined as the difference between the speeds of the clutchs' input shaft (engine) and output shaft (vehicle), relative to the speed of the input shaft. The complexity of the control task is increased further since optimal references for pressure, slip variables cannot readily be derived from the specifications i.e. to achieve a fast and smooth engagement.

Since operator comfort is one of the specifications for a good clutch engagement, a measure has to be used to quantify the comfort. In the remainder of this paper, the absolute value of the second derivative of the slip, i.e. the jerk, will be used for that purpose, since it is strongly related to the experienced operator discomfort [94]. For a fixed duration, the smoothest engagement is then obtained when the absolute value of jerk is either minimized or kept within (relatively small) bounds.

When engaging, the chamber of the clutch first has to fill up with oil, after which the pressure builds up until it is high enough to compress the return spring and move the piston towards the friction plates. This is called the fill phase, and it ends once the piston advances far enough and presses the plates together such that the transfer of the torque is initiated.

When the piston comes into contact with the plates, it triggers a sudden change in system dynamics, entering the slip phase. The slip value starts to decrease until both the input and output shafts rotate with the same speed (i.e. zero slip), at which point the clutch is considered engaged. Therefore, the wet-clutch is clearly composed of two sequential dynamical subsystems and optimal control of such switched systems is considered a hard problem [87].

To validate the practical applicability of this control scheme of algorithm 8, it is applied to the control of wet-clutch engagements. Since two phases can be distinguished during an engagement, two low-level controllers are used consecutively. The first aims to track a pressure reference in the filling phase, and is deactivated once the slip phase begins, at which point a second controller is activated to track a slip reference. These low level tracking controllers are discussed in the following sections.

Assuming well-performing tracking controllers for the pressure and slip at the low level, the remaining challenge is the selection and learning of appropriate reference trajectories at the high level. Even though an approximate shape can be defined based on knowledge of the clutch dynamics, the exact quantitative expressions are difficult to obtain. To this end, both reference trajectories are parameterized, each having two parameters that can be learned by the high-level learning

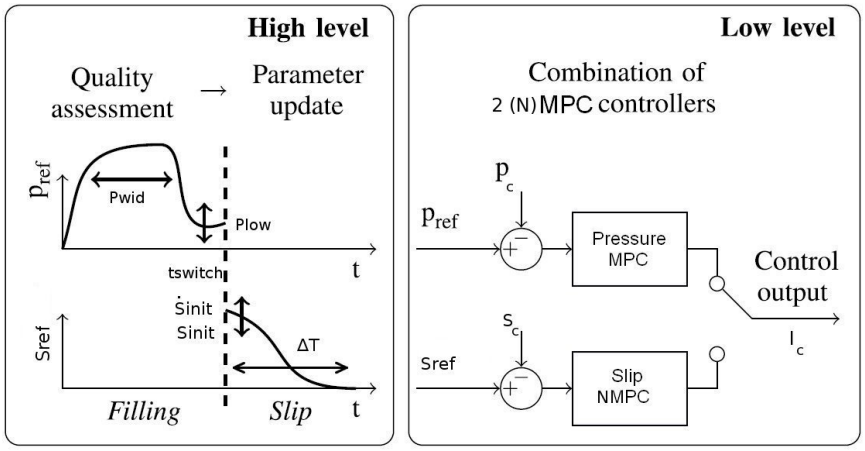


Figure 5.8: A schematic illustration of the proposed two-level control scheme. At high level, the pressure reference p_{ref} is parameterized by duration of high pressure p_{wid} and the low pressure value p_{low} at $t_{switch} = p_{wid} + 100$ms beyond which the slip reference s_{ref} is parameterized using the initial slip value and its derivative $s_{init}, \dot{s}_{init}$ respectively and the duration $\Delta T$to go to zero slip. At low level the pressure and slip references are tracked by MPC and NMPC controllers respectively. The measured pressure, slip and servovalve current to the clutch are denoted by P_c, S_c, I_c respectively.

laws to ensure an optimal performance is achieved and smooth transitions are obtained between the separate phases, as well as enabling adapting for variations in the operating conditions. The resulting two-level control architecture is illustrated in Fig. 5.8. The references and their update laws are discussed in the remainder of this section.

5.7.2 High-level learning algorithms for clutch control

5.7.2.1 Pressure reference profile and updating laws

The reference profile that has to be tracked by the pressure controller in the first part of the engagement, consists of two distinct parts. In the first part, the goal is to move the piston towards the plates such that it can press the pack of friction plates together. Since there is no significant amount of torque being transferred before this moment, and to reduce the delay before an operator can feel the reaction to a requested engagement, it is desired to move the piston as quickly as possible, so the pressure is maintained at a high level during this first part. When the piston approaches the friction plates however, it is needed to slow down the piston to avoid it hitting the plates brusquely causing drivetrain oscillations. To achieve this, the pressure reference is reduced to a low level and then gradually increased

again. Ideally, contact with the plates should occur just after the pressure dip, where the piston velocity will be lowest. Once the contact occurs, the gradual pressure increase then serves to build up the torque transfer needed to accelerate the output shaft and the attached load.

To parameterize this pressure reference, a piecewise expression is used containing two unknown parameters which are learned at the high level. The first of these two parameters is the duration of the high-pressure part at the beginning, indicated by the time p_{wid} in Fig. 5.8. The second parameter is the end-pressure to which the pressure is maintained at the end of the pressure control phase, denoted by p_{low}. The other values are all fixed in advance, with the duration of the pressure dip and the gradual increase both being 50 ms, and the low pressure value during the dip being 2 bar. A cosine function is used to ensure a smooth transition from the high to the low pressure. With these values, the overall pressure reference can now be expressed as: $p_{ref}(t) =$

$$\begin{cases} 12\text{bars}, & t \leq p_{\text{wid}}, \\ 12 - 9.5(\frac{1}{2} - \frac{1}{2}\cos\left(\frac{t-p_{\text{wid}}}{0.05}\pi\right))\text{bars}, & p_{\text{wid}} \leq t \leq p_{\text{wid}} + 0.05s, \\ 2.5 + (p_{\text{low}} - 2)(\frac{1}{2} - \frac{1}{2}\cos\left(\frac{t-0.05-p_{\text{wid}}}{0.05}\pi\right))\text{bars}, & p_{\text{wid}}+0.05 \leq t \leq p_{\text{wid}}+0.1s. \end{cases} \tag{5.22}$$

To learn the optimal parameter values at the high level, we use the knowledge that there is already some torque transfer happening prior to the piston making contact with the plates. This happens when the piston is close to the plates due to fluid coupling in the clutch. The absolute value of these torques is quite low, but it can still be detected by observing the speed changes of the in- and output shafts of the clutch. As a result, an estimate can be made of the time instant $t_{\text{drag,est}}$ the drag torque reaches a certain threshold value, by observing when the slip speed has also changed with a corresponding amount. At that time the piston is then close to, but not yet in contact with, the friction plates. Ideally, we want the piston to make contact with the plates after the dip in the reference, so we will try to get the instant $t_{\text{drag,ideal}}$ at which the drag reaches the threshold to occur just at the end of the dip, just before contact should happen. Since the drag is mainly influenced by the value of p_{wid}, the value of p_{wid} is then updated according to

$$p_{\text{wid}}(k+1) \;=\; p_{\text{wid}}(k) \;+\; \eta_1(t_{\text{drag,ideal}}(k) - t_{\text{drag,est}}(k)), \tag{5.23}$$

where the learning gain η_1 now needs to be selected. In this case, $\eta_1 < 0$. As a result, if the drag then starts too soon, the value of p_{wid} is reduced, making it likely that the drag will start at a later time during the next iteration. Similarly, if the drag starts too late, the value of p_{wid} is increased and the drag should start sooner during the next iteration. To select the absolute value of η, the typical errors $(t_{\text{drag,ideal}} - t_{\text{drag,est}})$ and the desired changes of p_{wid} are taken into account, and the

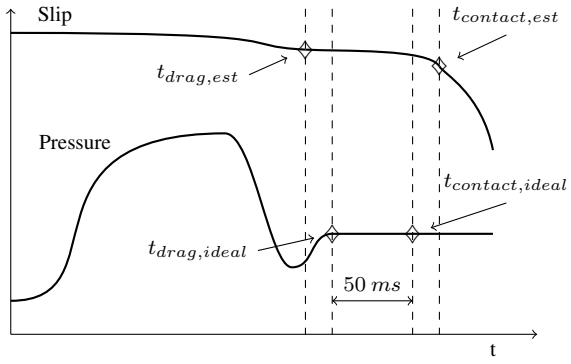


Figure 5.9: Updating mechanism for the pressure reference trajectory.

learning speed is selected low enough to avoid excessive changes. An illustration of this updating mechanism is given in Fig. 5.9.

A similar updating mechanism is used for the second pressure parameter, p_{low}. Since the pressure will be controlled to this value at the end of the pressure control phase, it should be chosen such that it ensures contact takes place and the slip phase commences. The moment contact occurs, $t_{\text{contact,est}}$, is again estimated by observing the speed changes of the in- and output shafts of the clutch. This value is then compared to that of the ideal contact time, $t_{\text{contact,ideal}}$, which we define as 50 ms after the pressure dip, and the difference is used to update the value of p_{low} according to

$$p_{\text{low}}(k+1) \;=\; p_{\text{low}}(k) \;+\; \eta_2(t_{\text{contact,ideal}}(k) - t_{\text{contact,est}}(k)), \tag{5.24}$$

where η_2 is again the learning gain, chosen in a similar manner as described for η_1. Since its value is negative, it will cause the value of p_{low} to be reduced if contact happens sooner than desired, and vice versa. This updating mechanism is also illustrated in Fig. 5.9.

5.7.2.2 Slip reference profile and updating laws

To ensure a smooth transition is obtained between the filling and slip phases, we will impose that the slip reference has to commence with certain initial conditions, which are updated by the high level learning laws. These initial conditions are the initial slip value and its derivative, denoted as s_{init} and $\dot{s}_{init}$. Once these are specified, the reference profile s_{ref} that has to be tracked by the slip controller in the second phase can be derived analytically.

Besides these initial conditions

$$s_{ref}(0) \;=\; s_{init} \quad \wedge \quad \dot{s}_{ref}(0) \;=\; \dot{s}_{init}, \tag{5.25}$$

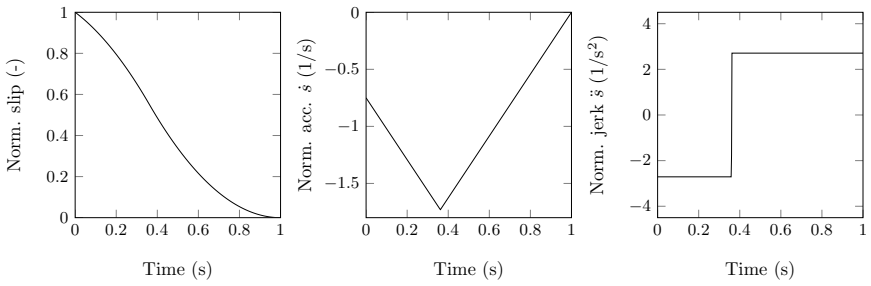


Figure 5.10: Evolution of bang bang jerk profile with Δt fixed to 1 s

the clutch also has to fully engage after the predefined duration ΔT, so that

$$s_{ref}(\Delta T) \;=\; 0 \tag{5.26}$$

Taking into account these specifications, the goal of the slip controller is to reduce the slip to zero while keeping the values of the jerk $\ddot{s}$ low, since the jerk is an important factor in the comfort experienced by the operator during a clutch engagement [94, 95]. For a given allowable duration ΔT, the best neutral-to-drive shift will thus be found when the highest absolute value of the jerk, $\|\ddot{s}_{ref}\|_\infty$, is minimized. The resulting slip reference profile corresponding to these specifications and its derivatives are shown in Fig. 5.10, where the jerk shown on the right clearly displays a bang-bang profile, with the absolute value of the jerk always being equal to the same value, $\ddot{s}_{\text{max}}$. Initially, there is a period where $\ddot{s} = -\ddot{s}_{\text{max}}$ during which the slip is reduced as quickly as possible. In the next period, $\ddot{s} = \ddot{s}_{\text{max}}$ is maintained up to the time $t = \Delta T$. Even though this leads to a slower reduction of the slip, this is needed to ensure that the slip derivative at synchronization equals 0, since otherwise an infinite jerk would be obtained at that time. If the moment where the value of $\ddot{s}$ changes is denoted by T_1, the values of $\dot{s}$ can be found by integrating this bang-bang profile as:

$$\dot{s}(t) \;=\; \begin{cases} \dot{s}_{init} \;-\; \ddot{s}_{\text{max}} t, & t \le T_1, \\ \dot{s}(T_1) \;+\; \ddot{s}_{\text{max}}(t - T_1), & t > T_1. \end{cases} \tag{5.27}$$

With this set of equations, it then becomes possible to integrate again to find the value of s in a similar manner

$$s(t) \;=\; \begin{cases} s_{init} \;+\; \dot{s}_{init} t \;-\; \ddot{s}_{\text{max}} t^2/2, & t \le T_1 \\ s(T_1) \;+\; \dot{s}(T_1)(t - T_1) \;+\; \ddot{s}_{\text{max}}(t - T_1)^2/2, & t > T_1 \end{cases} \tag{5.28}$$

The first lines of equations (5.27) and (5.28) can be used to find values for $s(T_1)$ and $\dot{s}(T_1)$, which can be inserted in the second lines of these equations.

When the resulting equations are combined with equation (5.26), a system of two equations is obtained, where all values are known except for T_1 and,

$$\ddot{s}_{\max} = \begin{cases} 0 = & \dot{s}_{init} - \ddot{s}_{\max} T_1 + \ddot{s}_{\max}(\Delta T - T_1) \\ 0 = & s_{\text{init}} + \dot{s}_{init} T_1 - \ddot{s}_{\max} T_1^2/2 + \\ & (\dot{s}_{init} - \ddot{s}_{\max} T_1)(\Delta T - T_1) + \ddot{s}_{\max}(\Delta T - T_1)^2/2 \end{cases} \tag{5.29}$$

After some manipulation of these equations, a solution can be found by solving

$$0 \;=\; T_1^2 \;(2\dot{s}_{init}) \;+\; T_1 \;(4 s_{init}) \;+\; (-2 s_{init}\Delta T - \dot{s}_{init}\Delta T^2). \tag{5.30}$$

This yields two solutions for T_1, from which the value needs to be selected such that

$$T_1 \;\geq\; 0 \quad \wedge \quad T_1 \;\leq\; \Delta T, \tag{5.31}$$

after which this value for T_1 can be used to find $\ddot{s}_{\max}$ by inserting it into equation (5.29) as

$$\ddot{s}_{\max} \;=\; \frac{\dot{s}_{init}}{2T_1 - \Delta T}. \tag{5.32}$$

Once the values of T_1 and $\ddot{s}_{max}$ are found, the profile for $s_{ref} = s(t)$ can be found by evaluating equation (5.28), and this profile can be used for the slip controller during the second phase of the overall control strategy.

At the time the slip controller is activated, the piston is already in contact with the plates and the slip is already changing. The main difficulty with generating good slip references is then simply the selection of the initial values s_{init} and $\dot{s}_{init}$. These are set equal to the measured values of the slip and its derivative during the last trial. This ensures that the initial tracking error is close to zero, making the control task easier, and avoids any transitional effects that could cause operator discomfort.

5.7.3 Low-level stabilizing (N)MPC controllers for clutch control

For the filling phase, a model predicting the generated oil pressure P_c as a function of the valve current I_c is needed. Since the system is quasi-linear (i.e. mild static nonlinearity) in the filling phase itself, a linear model suffices. To obtain this, a frequency response function (FRF) is first estimated by averaging its response to a number of multisine periods and realizations. Based on this FRF, a finite order rational transfer function is then fitted in the z-domain. For the open clutch i.e. before the piston touches the friction plates, the resulting model is given by [12]:

$$\frac{P_c(q^{-1})[Output \;\; in \;\; Bars]}{I_c(q^{-1})[Input \;\; in \;\; Amps]} = \frac{0.234 \cdot q^{-2}}{1 - 2.736 \cdot q^{-1} + 2.515 \cdot q^{-2} - 0.7764 \cdot q^{-3}} \tag{5.33}$$

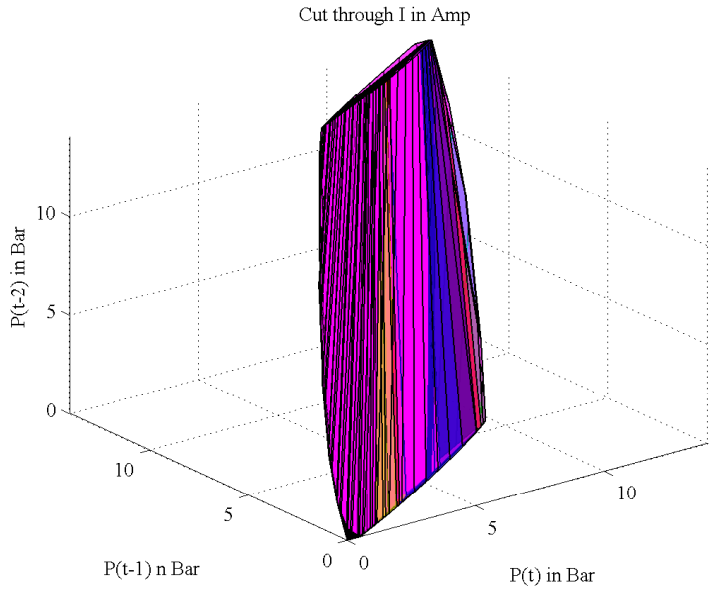


Figure 5.11: The positively invariant subset of feasibility set demonstrating recursive feasibility of the EPSAC controller without terminal conditions.

Since the pressure system has a relatively high bandwidth, the model is sampled at $T_s = 1$ ms, and is used to develop a linear EPSAC-MPC controller with the same sampling time. This controller takes into account the following constraints, which come from the physical limits on the actuation of the system:

$$\begin{aligned} 0 \leq I_c(Input \;\; in \;\; Amps) \leq 0.8 \\ 0 \leq P_c(Output \;\; in \;\; Bars) \leq 14 \end{aligned} \tag{5.34}$$

The various control parameters are chosen as $N_1 = 2, N_u = 1, N_2 = 10, \lambda = 10^{-3}, \frac{C}{D} = \frac{1}{1-q^{-1}}$ for nominal performance. This combination of extended prediction horizon and short control horizon achieves mean level control and ensures robustness towards varying parameters. The default integrator filter is to obtain zero steady-state error. The constrained optimization problem is feasible in the nominal conditions and the application being repetitive would remain recursively feasible. Moreover the optimal solution is computed well within 1 ms by the active-set strategy implemented in real-time over a dSPACE 1103 control board.

Now we analyze the a posteriori practical stability of the given input-output EPSAC MPC controller without terminal conditions. Recollect from chapter 2, that the first step is to transform the input-output model to state-space. Since the

model from input current to output pressure is of 3^{rd} order with a unit delay, the state vector takes the form: $x(t) = [P_c(t), P_c(t-1), P_c(t-2), I_c(t-1)]^T$ and the corresponding state constraints in the nominal case: $[0,0,0,0]^T \leq x(t) \leq [14,14,14,0.8]^T$. The state-space maps $\mathcal{A}, \mathcal{B}, \mathcal{C}$ can be computed by using corollary 2.4.2. Now, as per the a posteriori feasibility testing methodology developed in chapter 3, there are two options for proving practical stability: one is the conservative test of persistent feasibility and the other is the less conservative test of recursive feasibility. It turns out that the persistent feasibility test fails to certify the EPSAC controller for the given model using the specified horizons. Therefore, the explicit solution to the EPSAC controller is found using theorem 2.4.3 along with its feasibility set X_F. Next a positively invariant region $O(X_F)$ with 37 regions is computed within the feasibility set by using algorithm 2, as shown in Fig. 5.11. The presence of such a set certifies recursive feasibility and induces practical stability of the EPSAC clutch controller.

For the slip phase, a model capturing the dynamics between the current I_c to the servo-valve and the slip S_c is needed. An MPC using a linear slip model has not been considered here since it has been shown to be suboptimal in our previous work [19]. Given that a linear MPC controller is inadequate for slip control, other linear control methods like PID would neither work. A nonlinear model is therefore needed, mainly due to variation in the values of the friction coefficient depending on the slip value. To find such a non-linear model, a frequency response function (FRF) is first estimated as before. Using the linear model parameters as an initial guess, a polynomial nonlinear state space model is then estimated by solving a nonlinear least squares optimization problem with the Levenberg-Marquardt (LM) algorithm [96]. This results in a polynomial nonlinear state space model from input $u(t) = I_c$ Amps to normalized output $y(t) = S_c$ (the states $x(t)$ have no physical meaning):

$$\begin{aligned} x(t+1) &= A_s \cdot x(t) + B_s \cdot u(t) + E_s \cdot v(t) \\ y(t) &= C_s \cdot x(t) + D_s \cdot u(t) + F_s \cdot v(t) \end{aligned} \tag{5.35}$$

with $A_s \in \mathbb{R}^{4\times4}$, $B_s \in \mathbb{R}^{4\times1}$, $C_s \in \mathbb{R}^{1\times4}$, $D_s \in \mathbb{R}$, $E_s \in \mathbb{R}^{4\times15}$ and $F_s \in \mathbb{R}^{1\times15}$. The vector $v(t)$ contains the state and input monomials and is
$v(t) = [x_1^2, x_2^2, x_3^2, x_4^2, u^2, x_1^3, x_2^3, x_3^3, x_4^3, u^3, x_1^4, x_2^4, x_3^4, x_4^4, u^4]'$.

A NEPSAC controller is designed taking into account he following constraints:

$$\begin{aligned} 0.15 \leq I_c(Input \;\; in \;\; Amps) \leq 0.25 \\ 0 \leq S_c(Output \;\; normalized) \leq 1 \end{aligned} \tag{5.36}$$

The constraint on current has been tightened as the nonlinear slip model is bounded input bounded output stable only within this region. The constraints on the slip are trivial. The control parameters are tuned to $N_1 = 1, N_u = 4, N_2 = 5, \lambda =$

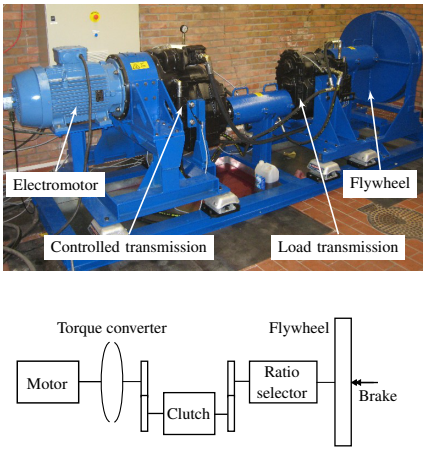


Figure 5.12: Testbench consisting of an electromotor driving a flywheel via two mechanical transmissions.

$O(10^2)$, $\frac{C}{D} = \frac{1}{1-q^{-1}}$. As stated earlier, the controller can be helped by selecting a good base input vector, which is achieved here by setting it to p_{low}. This also helps to avoid running into issues with the model at the edges of the stability region and getting into local minima, as the second term in the cost function (5.7) then penalizes deviations from this value, thereby increasing the robust feasibility of the NMPC. The chosen combination of control horizon and control penalty gives the controller enough degrees of freedom (equal to the model order) and ensures a low controller activity such that the constraints are satisfied and the output does not deviate from the reference. Since a high value of λ is chosen the NEPSAC algorithm by theorem 5.2.1 is almost sure to converge to a locally optimal solution. Note that, in this nonlinear case, the active-set based primal-dual solver embedded in the NEPSAC controller is called multiple times within the same sampling instant till convergence. For the slip phase, convergence is achieved within 4 iterations, but a sampling time of 1 ms is no longer sufficient for the associated computations. Moreover, since the nonlinear dynamics of slip phase has a lower bandwidth, this model is sampled at $T_s = 10$ ms, thus allowing sufficient time for the 4 iterations to be completed each time.

5.7.4 Experimental results

The developed methodology is validated on the experimental test setup shown in Fig. 5.12, where an electromotor (of power 30kW) drives a flywheel (of inertia 2.5kgm^2) via a torque converter and two mechanical transmissions. The transmis-

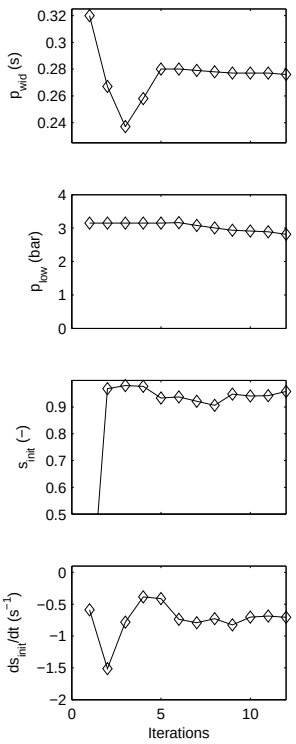


Figure 5.13: Evolution of the reference signal parameters for the 2L-NMPC, with from top to bottom: pressure peak duration, pressure level at the end of the filling phase, the initial slip speed, initial slip derivative.

sion to be controlled comes equipped with incremental encoders measuring the rotational speeds of the in- and output shafts, and also comes with a sensor measuring the pressure in the line to the clutch. An additional sensor is used to measure the transferred torque, but this is used only for illustrative purposes and is not typically available in production units. A dSPACE 1103 control board is used to run the controller and drive the servo-valve current. A cooling system is installed that allows to maintain the oil temperature at a constant level, so that during the tests the temperature is either $40°\mathrm{C}$ or $70°\mathrm{C}$. To reiterate, the control objective is to minimize the maximum jerk, with $\Delta T = 1\mathrm{s}$, refer 5.7.2.2.

In a first test at nominal conditions, the oil is maintained at $40°\mathrm{C}$ and the second transmission is set such that the observed inertia at the clutch becomes

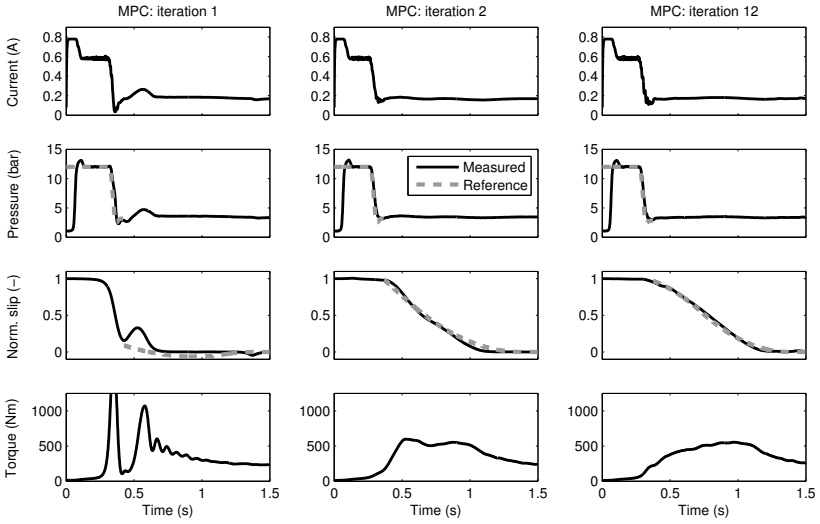


Figure 5.14: Engagements achieved by the 2L-NMPC at nominal conditions.

8.4kgm^2. For this test, the parameter values for the reference trajectories are intentionally initialized poorly to illustrate the convergence process. Then the learning starts and the parameters are adapted after each engagement for 2L-NMPC (two-level NMPC) with their evolution shown in Fig. 5.13. The corresponding input current and measured pressure, slip and torque at several points during the convergence process is shown in Fig. 5.14 for the 2L-NMPC.

The initial performance is expectedly poor with a high torque peak due to an initial overfilling, resulting in an uncomfortable engagement for an operator. As a result, during the first parameter update after an engagement by 2L-NMPC, the high-level controller reacts by reducing t_{switch} as shown in Fig. 5.13. The corresponding engagement in Fig. 5.14 already reports significant improvement in performance. The slip reference is however not yet adapted to the new conditions at the end of the pressure control phase which causes some tracking error in the slip control loop. Another issue is the jerking of the torque that appears when synchronization occurs with a steep slope of the slip profile. Over the course of next engagements, the parameters of the pressure reference converge further such that they eventually remain almost unaltered, see Fig. 5.13. Consequently, the initial conditions for slip control remain fixed, allowing the slip parameters to converge as well. As a result, an appropriate start of the slip reference trajectory is found by the 12^{th} iteration of the 2L-NMPC, ensuring that the transitional effects have completely disappeared, and jerk free smooth engagements are obtained.

To test the robustness, two additional test cases are considered. Firstly, the load

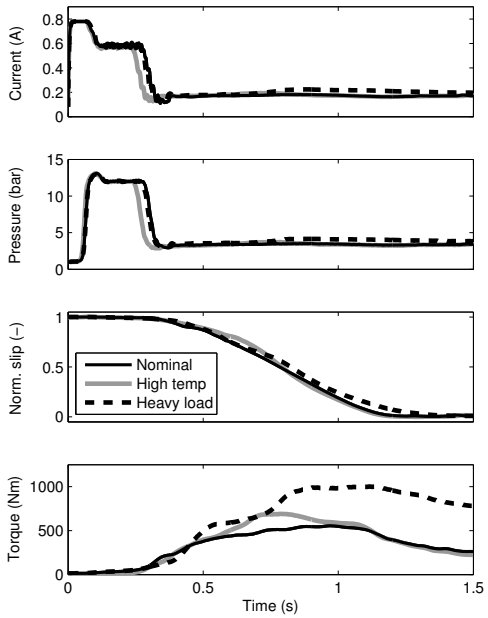


Figure 5.15: Demonstration of the robustness against variations in temperature and load by the 2L-NMPC (after 10 *iterations)*

is kept at the same value but the oil temperature is increased to $70°\mathrm{C}$. Secondly, the oil is again at $40°\mathrm{C}$, but the second transmission is set such that the observed inertia is increased to $28.6\mathrm{kgm}^2$.

Fig. 5.15 shows the results obtained by the 2L-NMPC controller. Since the oil viscosity decreases strongly when the temperature increases, the filling can be completed sooner and with less effort. The controller has mainly compensated for this by reducing p_{wid} of the pressure reference trajectory. As a result, the performance is very close to the nominal case, but the engagements start a little bit sooner. For the test where the observed load is increased, higher torques and pressures are required to accelerate the larger load, but the slip signals remain very similar to the nominal case. Since the NMPC controllers are designed using models obtained at the nominal settings, the fact that a good performance can still be achieved during these tests at different operating conditions demonstrates the robustness of the developed controllers.

The two-level control technique can be seamlessly extended to be used for a vast majority of switching systems. For example, the immediate foreseeable ex-

tension is controlled gear shifting in automatic transmissions. This leads to various phases of operation and depending on the number of phases, appropriate high and low level controllers may be designed. Next, control of flexible manipulators is an exciting field of research. The system transits between two types of output feedback: (i)only the base angle of the manipulator is measured, (ii) the tip angle is also measured (when it is greater than some threshold). Thus two controllers are designed optimized for feedback from (i)base angle only, (ii)base and tip angle. High level control algorithm is then used for switching. The two-level NMPC can also be used in adaptive supervisory control where the supervisor must try to determine which is the correct process model by observing inputs and outputs and then select the appropriate controller.

The 2L-NMPC scheme is compared with other model-based and model-free learning control technologies both at qualitative and experimental levels in appendix B and significant conclusions are drawn on their applicability.

5.8 Summary

In this chapter, the entire philosophy of designing and proving stability for linear MPC controllers in the input-output formulation without terminal conditions has been extended to nonlinear systems and NMPC controllers. The NEPSAC control algorithm, which has been widely used in the industry due to its simplicity has been adopted in the thesis as the baseline NMPC strategy. However, the proof of convergence of NEPSAC iterations remained an open problem for long. A major contribution of this thesis is in tackling this problem and giving a formal proof of convergence to locally optimal solution by using Taylor's series expansion and using control penalty to guarantee a monotonic decrease in cost. This is illustrated on an example of nonlinear longitudinal flight control.

Industrial systems especially production machines exhibit switching nonlinearity. The switching dynamics has been categorized into piecewise affine systems and switched nonlinear systems. Many nonlinear systems can be identified as a PWA system i.e. a collection of linear systems with associated regions with bounded uncertainty. Since the explicit MPC law is also PWA, the application of explicit MPC to PWA system would also lead to PWA controller and should be rather simple. The invariant set theory is now extended to PWA systems and a test of persistent feasibility for MPC of PWA systems is derived in the nominal case. Further, the skeleton of the algorithm to derive the explicit solution to the PWA MPC is given, to be able to derive a positively invariant subset of the feasibility set to prove recursive feasibility.

A representative example of a car riding over a PWA hill is presented in the input-output domain. A state-space is then derived and persistent feasibility is shown for $N_u = 1, N_2 = 10$ and recursive feasibility for $N_u = 1, N_2 = 4$ thus

giving less conservative certificate of practical stability.

Next, switched nonlinear systems are introduced as the class of systems where the dynamics and the very meaning of states change between the constituent systems, thus differentiating from PWA systems.

A two-level control scheme is developed and presented in the context of clutch control where a high level controller generates reference trajectories and switching instances for the low level tracking controllers of the switched nonlinear systems. The operation of the wet-clutch engagement has been broken down into linear fill switching to nonlinear slip phase. A high level iterative learning controller has been designed to generate the references as well as switching times. At low level EPSAC controller for fill phase has been designed without terminal conditions and has been a posteriori certified stable by our developed techniques of feasibility in chapters 2 and 3. Further, a NEPSAC controller for fill phase has been designed with high damping or control penalty λ that would result in almost guaranteed convergence of the NEPSAC algorithm. The engagement control results on a real wet-clutch test bench under nominal and perturbed conditions are shown to be superior in terms of obtaining low jerks and engagement times. The two-level NMPC can be applied to a wide range of switching mechatronic systems requiring a supervisory control.

6

Distributed Nonlinear Predictive Control

We have thus far considered the design and stability analysis of linear and nonlinear MPC without terminal conditions of (multivariable) monolithic systems, however global systems are composed of many such interacting monolithic subsystems and the associated NMPC(s) throw up new challenges. The ever increasing complexity of large scale systems found nowadays in process industry, manufacturing systems and traffic networks urged the control community to revise old concepts of distributed control and develop novel, pragmatic approaches. An excellent review of the current techniques used in practice is given in [97]. The challenge for control is that these large scale systems are composed of many interacting subsystems. They can be difficult to control with a linear centralized control structure due to nonlinearity, computational complexity and limitations on communication [98,99].

Many industrial systems can be described by a hierarchical structure where an algorithm at the higher level coordinates the actions of local regulators placed at a lower level. However, often the high level algorithm becomes so complex that it becomes hard to justify its advantages over a centralized controller [97]. For all these reasons, in the last decade, many distributed control structures have been developed and the nonlinear model predictive control (MPC) approach was recognized as one of the most suitable candidates [100, 101]. This is not surprising, since MPC has a great potential to play a crucial role in distributed control due to its intrinsic forecasting properties which can be exchanged (in part) to neighbouring MPC units [37, 97, 102–104].

6.1 Introduction

In traditional decentralized control, the interaction between the subsystems is ignored, resulting in suboptimal and at times unstable closed-loop. Centralized control on the other hand, requires all the information to be available to one controller that results in heavy computation costs, neither of which are feasible. In this chapter, we introduce a pragmatic approach to distributed NMPC by taking into account the interaction amongst the constituent sub-systems with minimum amount of information exchange such that the improvement in overall cost is guaranteed. We make use of the in-house developed NEPSAC (Nonlinear Extended Prediction Self-Adaptive Control) algorithm introduced and analyzed in 5.2, without terminal conditions as a basis for stemming the proposed approach [34]. The novelty of our approach is the ease of implementation preserving the guarantee of improvement in cost, pragmatism and ability to tackle constraints without significant computational complexity. In order to test these claims, a hydrostat drivetrain system is used, consisting of two highly nonlinear, time-varying dynamic, interacting sub-systems. This is a representative global mechatronic system (system composed of interacting sub-systems) widely used in industry, e.g. in mobile vehicles such as ground moving machines, agricultural machines, forest machines, industrial and mining lifters. The use of hydrostatic transmission as the vehicle drives is primarily motivated by its large range of continuously variable speed, high maneuverability and a possibility to increase the overall efficiency [105].

In such machines and other large scale systems including industrial production lines, wind mills, smart grids etc., centralized control design is not feasible because of the associated high costs on computation and communication. More so, a failure would lead to total shut down of the system. This motivates us to develop a distributed nonlinear model predictive control (DNMPC) framework which can guarantee an improvement in the overall cost with every cycle of control computation by each of the distributed controllers. This technique is applied for the distributed nonlinear control of the hydrostat and to the best of our knowledge this opens up a new way of viewing and controlling such global production machines.

6.2 Distributed Nonlinear MPC

Most large scale industrial systems are still controlled in decentralized way where the input and output are grouped in disjoint sets. Once this structure has been fixed the local regulators (R1 and R2) are designed in completely independent fashion, which is trivial when the interactions (input or state) are weak. However, strong interactions can cause the system to become unstable [106].

The centralized controller, on the other hand assumes knowledge of all the information to be used in control computation. This centralized concept is often

used in the form of a coordinator at a higher level that coordinates the actions of local regulators placed at a lower level, resulting in a two-level hierarchical control structure. The immediate drawbacks are the need for centralized communication and high associated computational costs [97].

The distributed control mechanism lies somewhere in between centralized and decentralized control. Within the distributed control framework, a limited amount of information is transmitted between the local regulators, such that each of them have some knowledge about the behavior of its interacting neighbours. For simplicity, a schematic overview is given in Fig. 6.1, depicting the concept of distributed MPC with interacting sub-systems. The information exchange can be performed either:

- noniterative, i.e. information is transmitted/received only once each sampling period; or
- iterative, i.e. information is transmitted/received many times to reach global consensus within each sampling period.

The cost function can be formulated as either:

- independent, i.e. each regulator minimizes a local performance index; or
- cooperating, i.e. all local regulators minimize a global cost function.

The clear advantages of the distributed NMPC approach are reduction in computation and communication costs, improvement in robustness with respect to failures in information transmission, improvement in the modularity and flexibility of the global system and may synchronize subsystems working at different time scales. Agent based distributed model predictive control scheme is another approach that is based on cooperative game theory and has been reported for constrained linear systems in [107]. In this chapter, we would explore the cooperative approach due to its established stability properties [108] along with the noniterative scheme to be able to perform fast computation, and give an extension for nonlinear systems with a guarantee on monotonic decrease in global cost.

6.2.1 Proposed DNMPC algorithm

In this section, we elaborate on the DNMPC controller design. To facilitate the exposition, we assume the plant comprises only two subsystems. For a process with number of inputs $n_u = 2$ and number of outputs $n_y = 2$, the structure of the generic process model becomes [34]:

$$y_1(t) = \hat{y}_1(t) + n_1(t) \quad \text{and} \quad y_2(t) = \hat{y}_2(t) + n_2(t) \tag{6.1}$$

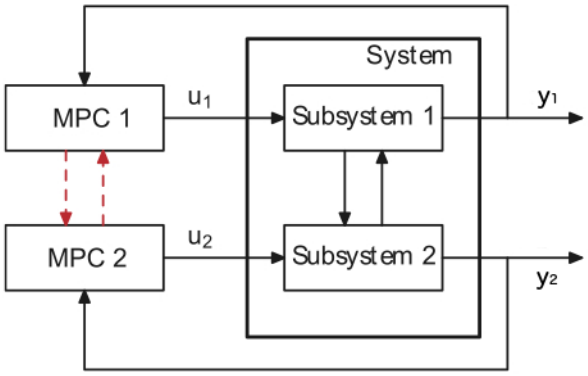


Figure 6.1: Distributed MPC of two interacting subsystems with information exchange

with $y_i(t), \hat{y}_i(t), n_i(t)$ as the i^{th} process output, model output, disturbance respectively, where,

$$\begin{aligned}\hat{y}_1(t)&=f_1[\hat{y}_1(t-1),\hat{y}_1(t-2)\dots u_1(t-1),u_1(t-2)\dots u_2(t-1),u_2(t-2)\dots]\\ \hat{y}_2(t)&=f_2[\hat{y}_2(t-1),\hat{y}_2(t-2)\dots u_1(t-1),u_1(t-2)\dots u_2(t-1),u_2(t-2)\dots]\end{aligned} \tag{6.2}$$

in the case where the system is available in the input-output formulation. In the case of a state-space realization:

$$\begin{aligned}x_1(t+1) &= f_1^x(x_1(t), x_2(t), u_1(t), u_2(t)),\\ x_2(t+1) &= f_2^x(x_1(t), x_2(t), u_1(t), u_2(t)) \qquad (6.3)\\ y_1(t) &= g_1(x_1(t), x_2(t)) + n_1(t), \quad y_2(t) = g_2(x_1(t), x_2(t)) + n_2(t) \qquad (6.4)\end{aligned}$$

The disturbances are modeled by colored noise processes

$$n_1(t) = (C_1(q^{-1})/D_1(q^{-1})) \cdot e_1(t) \quad \text{and} \quad n_2(t) = (C_2(q^{-1})/D_2(q^{-1})) \cdot e_2(t) \tag{6.5}$$

where $e_1(t), e_2(t)$ are zero mean white noise sequences. In the linear case, the future response can then be expressed as:

$$\begin{aligned}y_1(t+k|t) &= y_{1base}(t+k|t) + y_{1opt}(t+k|t) \quad \text{and}\\ y_2(t+k|t) &= y_{2base}(t+k|t) + y_{2opt}(t+k|t) \qquad (6.6)\end{aligned}$$

where the predictions are made at time t over the prediction horizon $k \in [N_1, N_2]$. In vector notation:

$$Y_1 = \bar{Y}_1 + G_{11} \cdot U_1 + G_{12} \cdot U_2 \quad \text{and} \quad Y_2 = \bar{Y}_2 + G_{21} \cdot U_1 + G_{22} \cdot U_2 \tag{6.7}$$

where $\bar{Y}_1, \bar{Y}_2$ are the base responses computed as the cumulative effect of both the past control inputs/outputs (states), the apriori defined future control actions U_{1base}, U_{2base} and the predicted disturbances. The rest of the optimizing terms are the discrete time convolution of the predicted optimal inputs U_1, U_2 (defined as the increments to U_{1base}, U_{2base}) with the corresponding impulse response coefficients i.e. the respective G_{ij} matrices defined from input j to output i, where $i, j \in 1, 2$. Thus, $U_1 = [\delta u_1(t|t) \dots \delta u_1(t+N_{u1}-1|t)]^T$ where N_{u1} is the control horizon.

Next, consider the following global cooperating cost function (without terminal conditions):

$$\begin{aligned} &V = V_1(U_1, U_2) + V_2(U_1, U_2), \\ &\text{where, } V_i = (R_i - Y_i)^T \cdot (R_i - Y_i) + U_i^T . \Lambda_i . U_i \end{aligned} \tag{6.8}$$

where R_i, Λ_i are the respective reference trajectories and control penalty matrices for $i = 1, 2$. It follows that the optimization problem for MPC-1 is:

$$U_1^* = \min_{U_1} V, \text{ subject to } U_1 \in \mathbb{U}_1^c \text{ and } U_2 = U_{2base}^{*-1} \tag{6.9}$$

where U_2^{*-1} is the optimal input trajectory communicated by MPC-2 delayed by one sample (the last control value is repeated to form the full vector), and $\mathbb{U}_1^c$ is the polytopic constraint set for MPC-1, which arise from the input and output constraints of the model. An analytical solution can be obtained in the unconstrained case:

$$U_1^* = (G_{11}^T \cdot G_{11} + G_{21}^T \cdot G_{21} + \Lambda_1 \cdot I)^{-1} \cdot (G_{11}^T \cdot (R_1 - \bar{Y}_1) + G_{21}^T \cdot (R_2 - \bar{Y}_2)) \tag{6.10}$$

Similarly, an explicit solution can be derived for U_2^*. Note that, in many cases a control horizon of $N_u = 1$ sample suffices and then we can still use the explicit solution followed by clipping. A short control horizon is widely used in industry, at least in the case of stable plants when only input constraints are active. When the underlying process model is nonlinear, for controller-1, the superposition of (6.6) is still valid only if the term $y_{1opt}(t+k|t)$ is small enough compared to $y_{1base}(t+k|t)$. This is true when $\delta u_1(t+k|t)$ is small, which is the case if $u_{1base}(t+k|t)$ is close to the optimal $u_1^*(t+k|t)$. To address this issue, the idea is to recursively compute $\delta u_1(t+k|t)$ using (6.10), within the same sampling period, until $\delta u_1(t+k|t)$ converges to 0. Inside the recursion, $u_{1base}(t+k|t)$ is updated each time to $u_{1base}(t+k|t) + \delta u_1(t+k|t)$, i.e. the extension of EPSAC algorithm for nonlinear systems [34, 109]. Notice that linearization of the process is not necessary in this case, which is a significant advantage over other NMPC strategies. The procedure is similar for controller-2 and both controllers are further denoted in the remainder of this paper as NMPC-1 and NMPC-2.

The steps comprising non-iterative sequential DNMPC for the 2×2 process is summarized in algorithm 9 (after suitable initialization of controls $U_{ibase}, i =\in \{1, 2\}$).

1. NMPC-1:
   (a) Compute $U_1^* = \arg\min_{U_1} V$ subj. to $U_1 \in \mathbb{U}_1^c$, $U_{2base} = U_{2base}^{*-1}$.
   (b) Update $U_{1base} = U_{1base} + U_1^*$.
   (c) Communicate U_{1base} to NMPC-2 together with $n_1(t)$.
   (d) Apply the first input of U_{1base} to subsystem 1.
2. NMPC-2:
   (a) Compute $U_2^* = \arg\min_{U_2} V$ subj. to $U_2 \in \mathbb{U}_2^c$, $U_{1base} = U_{1base}^*$.
   (b) Update $U_{2base} = U_{2base} + U_2^*$.
   (c) Communicate U_{2base} to NMPC-1 together with $n_2(t)$.
   (d) Apply the first input of U_{2base} to subsystem 2.
3. Go to step 1 at the next sampling period.

algorithm 9: Sequential cooperative DNMPC

There are two main advantages of the proposed strategy. Firstly, if a temporary drop in communication occurs, NMPC-1 can compute both U_1^* and estimate of U_2^* (and similarly for NMPC-2) while only the disturbance estimates are exchanged until communication is re-established. Secondly, for large scale systems with for instance $n_u = 50$ decision variables, even by using $N_u = 1$ for each input, the resulting MPC problem involves an optimization over $n_u * N_u = 50$ variables, which requires large matrix inversion and complex quadratic programming. However, the DNMPC algorithm simplifies the computation by considering 50 subsystems, each optimizing over the control horizon $N_u = 1$. In this case, the analytical solution can be used, which reduces the computational burden to a scalar division.

A centralized NMPC (CNMPC) i.e multivariable NEPSAC would minimize the same cost function of (6.8) but now with respect to the multivariable vector $[U_1, U_2]^T$ at once subject to the constraints on $[U_1 \times U_2]$. This, even in the limiting case i.e. when both the inputs have individual control horizons of 1, adds up to a net of 2 optimizing variables and hence can only be solved by quadratic programming, which has an exponential cost. Comparatively, in this case the DNMPC with control horizons 1 will have a polynomial time complexity.

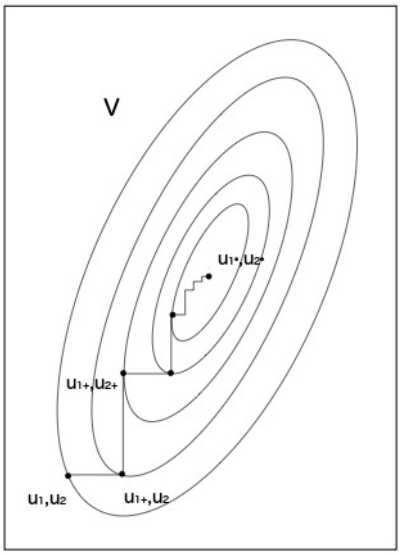


Figure 6.2: The sequential distributed NMPC algorithm approaching locally optimal solution of centralized NMPC (u_1, u_2 is an initial point following which direction 1 is updated to u_{1+}, then direction 2 to u_{2+} and so on till local optimum u_{1*}, u_{2*} is attained shown by the level sets of the cost V).

Assumption 6.2.1. *The cost function of* (6.8) *is smooth and twice differentiable.*

Theorem 6.2.1. *The presented sequential non-iterative DNMPC algorithm 9 without terminal conditions achieves a monotonic decrease in global cost with every iteration, in the nominal case.*

Proof. A multivariate nonlinear system usually leads to a non-convex cost function, which is convex around a multivariate neighbourhood where a local minimum exists as a function of the constituent variables. An initial cost can be written down as:

$$V(U_{1base}, U_{2base}) = \sum_{i=1}^{2} (R_i - Y_i)^T \cdot (R_i - Y_i) \tag{6.11}$$

Around this point, a Taylor's series expansion in the first direction leads to:

$$\begin{aligned} V(U_{1base} + U_1, U_{2base}) = {} & (R_1 - \bar{Y}_1 - G_{11} \cdot U_1)^T \cdot (R_1 - \bar{Y}_1 - G_{11} \cdot U_1) \\ & + (R_2 - \bar{Y}_2 - G_{21} \cdot U_1)^T \cdot (R_2 - \bar{Y}_2 - G_{21} \cdot U_1) + \sum_{i=1}^{2} U_i^T \cdot \Lambda_i \cdot U_i \end{aligned} \tag{6.12}$$

where the G's as usual are the respective Jacobians. Now, the presented DNMPC algorithm freezes the second direction and takes a gradient descent step in the first direction, leading to the perturbation U_1^* given by (6.10). An appropriate choice of the corresponding penalty matrix Λ_1 (Levenberg-Marquardt damping) ensures a guaranteed decrease in the overall cost.
Next, the first direction is updated and fixed and based on which a gradient descent step is taken in the second direction leading to an exactly similar expression for U_2^* with a guaranteed decrease in this direction ensured by Λ_2.
A combination of the above two steps ensures a monotonic cost decrease in both the directions is achieved. □

Lemma 6.2.1. *If the steps* $2, 3$ *of algorithm 9 are iterated over and over, convergence is guaranteed to the solution of the centralized NMPC.*

Proof. The theorem 6.2.1 established that a monotonic decrease is achieved by the sequential DNMPC algorithm 9 by computing U_1^* followed by U_2^*. Now, instead of stopping here, the second direction is frozen and a gradient descent step is taken in the first direction leading to a new U_1^* and so on and so forth till no further improvement is possible. At this point, the decrease in the global cost is 0 on both directions which is only possible if a stationary point is attained leading to the locally optimal cost:

$$V^*(U_{1base}^*, U_{2base}^*) \tag{6.13}$$

where $U_{ibase}^*, i \in \{1, 2\}$ are the locally optimal input trajectories. The satisfaction of assumption 6.2.1, guarantees that the algorithm does not get stuck in corners of the level sets of the global cost function and hence the coordinate descent would stop decrementing only at a locally optimal solution, same as the one obtained when the optimization is performed in a multivariable fashion by CNMPC. This is called Coordinate-Descent and is shown in Fig. 6.2. □

6.2.2 Distributed Adaptation

The production machines have intrinsic time-varying dynamics (e.g. oil temperature, density, leakage, etc). They are also intensively operated under varying environmental conditions (e.g. process properties, toxic gas, seasonal variations, etc). Consequently, these factors imply the necessity of an adaptation mechanism for updating the model parameters in a distributed sense. In this section we propose a simple, yet effective learning method.

The model equation for the i^{th} subsystem can be written as:

$$y_i(t) = \phi_i^T(t) \cdot \theta_i(t) + \phi_{i-}^T(t) \cdot \theta_{i-}(t) + \nu_i(t) \tag{6.14}$$

where the nonlinear system is assumed to be linear in the parameter vector θ. The vector ϕ contains all the past inputs and measurements (i.e. state), ν denotes the

error and subscript $i-$ denotes all other parameters except the parameters of i^{th} subsystem. We employ the classic recursive least squares (RLS) algorithm for the learning step. The distributed learning mechanism for a 2×2 system is proposed as follows:

1. Initialize $\theta_1(t-1)$ and $\theta_2(t-1)$;
2. RLS-1: Compute,
   $\theta_1(t) = \theta_1(t-1) + K_1 \cdot (y_1(t) - \phi_1^T(t) \cdot \theta_1(t-1) - \phi_2^T(t) \cdot \theta_2(t-1))$;
   in the least squares sense and communicate to RLS-2.
3. RLS-2: Compute,
   $\theta_2(t) = \theta_2(t-1) + K_2 \cdot (y_2(t) - \phi_2^T(t) \cdot \theta_2(t-1) - \phi_1^T(t) \cdot \theta_1(t))$;
   in the least squares sense and communicate to RLS-1.
4. Go to step 2 at the next sampling period.

The gain K_i can be computed recursively, refer [110]. Further the error term multiplied with the gain can be weighted with an exponential forgetting factor (improves sensitivity) [110]. Since the distributed RLS algorithm has the same structure as that of DNMPC, it was combined to add the $learning$ feature to the DNMPC.

The DRLS algorithm itself has mild requirements such as (i) the measurement noise is assumed to be white (ii) the parameters vary slowly and continuously which are generally true for RLS even. In such cases, the forgetting factor is prescribed to be in between 0.98 and 1. For the distributed RLS case, if the incoming information is not uniformly distributed in the parameter space, a directional forgetting factor may be used [111]. Therefore it is desirable to assign different forgetting factors to different parameters (however this is to be determined based on simulation).

Remark 6.2.1. *The arguments for the controller have been given for the case of two subsystems only, but same arguments apply for any finite $M > 0$ number of interconnected subsystems, where each subsystem has a copy of the plantwide model and can evaluate the objective function independently (by definition of cooperative control).*

6.3 Benchmark: Hydrostatic Drivetrain

Note that, this section is a result of collaborative work done jointly with FMTC and KUL under the framework of the IWT SBO funded LeCoPro project. In particular, the PID controller developed in 6.3.4 is due to FMTC and the experimental setup including interfacing and maintenance in 6.3.4 is also due to FMTC.

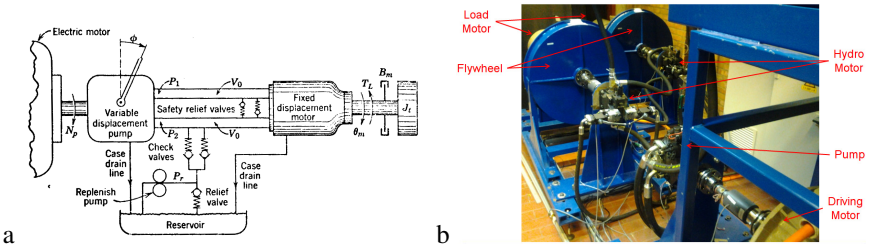


Figure 6.3: (a): A schematic of the pump controlled motor [2], (b): Hydrostat benchmark consisting of two hydromotors driven by a pump.

Generically, the hydrostatic drive uses fluid under pressure to transmit engine power in order to drive wheels or tracks (as opposed to hydrodynamic drive where power is transmitted by kinetic energy of the fluid). Mechanical power is converted to hydraulic power and back to mechanical power by a pump-motor synergy. The pump and the motor are joined in a closed hydraulic loop as shown in Fig. 6.3(a), which is good for power transmission when variable output speed is required. Hydrostatic transmissions outperform electrical, gear-type transmissions as they can offer fast response, maintain precise speed under variable load, allows infinitely variable speed control and can increase torque without changing gears. In a closed hydrostatic transmission, the torque can be transmitted in both directions, thus allowing hydrostatic breaking. However, this property implies the existence of a precise control of the traction effort and speed. Another important advantage for hydrostatic drives is the high efficiency and thereby low fuel consumption, when compared to hydrodynamic drives [105].

Typically, hydraulic systems are highly nonlinear, complex dynamic plants. For this reason, linear model-based controllers that are used in practice often fail to maintain good performance. Although nonlinear differential equations can be used to describe in detail a hydraulic system, it is difficult to find suitable model-based controllers without loss of implementability [105].

6.3.1 Modeling

To understand the hydrostatic assembly, we would briefly describe the working principles of a motor followed by a pump controlled motor. A hydraulic motor consists of a swash plate connected to a rotating barrel with pistons sitting on the plate via connectors. Valve plate ports inlet fluid to half of the cylinder barrel and pistons receiving this are forced against the swash plate. This causes the barrel attached to the drive shaft to rotate. Variable displacement can be achieved by varying the angle of the swash plate. A hydraulic motor can be made to work in

an opposite fashion i.e. the drive shaft rotation (due to a connected engine) now causes the pistons to generate a pressurized flow. This arrangement is called a hydraulic pump.

For an ideal hydraulic motor/pump, the mechanical power output is given by [2]:

$$hp|_{out} = T_g \cdot \omega_m \tag{6.15}$$

where T_g is the torque generated by the motor and ω_m the angular speed of the motor shaft. The hydraulic power supplied to the motor is:

$$hp|_{in} = (P_1 - P_2) \cdot Q_m \tag{6.16}$$

where P_1, P_2 are pressures in high, low pressure lines respectively (refer to Fig. 6.3(a)) and Q_m is the oil flow through the motor. Assuming 100% motor efficiency, we have that:

$$T_g = S_m \cdot (P_1 - P_2) \text{ where,} \tag{6.17}$$

$$S_m = Q_m/\omega_m \text{ (by definition)} \tag{6.18}$$

where S_m is the volumetric displacement (stroke) of the motor. However in practice, leakage flows and friction are important sources of losses. There are two types of leakage: internal and external. The internal leakage is given by:

$$Q_{im} = C_{im} \cdot (P_1 - P_2) \tag{6.19}$$

where C_{im} is the internal leakage coefficient. The external leakage is given by:

$$Q_{emi} = C_{em} \cdot P_i, \quad i \in \{1, 2\} \tag{6.20}$$

where C_{em} is the external leakage coefficient. Further, there are two major sources of torque losses:

1. the damping torque (due to shearing the fluid): $T_d = B_m \cdot \omega_m$ where, B_m is the viscous damping coefficient; and
2. the friction force opposing the motion of piston: $T_f \propto \text{Sgn}(\omega_m) \cdot (P_1 + P_2)$

Hence the resultant torque delivered to the load can be written as:

$$T_l = S_m \cdot (P_1 - P_2) - T_d - T_f \tag{6.21}$$

From the continuity equation for high (modulated) pressure forward chamber we have that (refer to Fig. 6.3(a)):

$$\begin{aligned} S_p \cdot \omega_p - C_{ip} \cdot (P_1 - P_2) - C_{ep} \cdot P_1 - C_{im} \cdot (P_1 - P_2) \\ -C_{em} \cdot P_1 - S_m \cdot \omega_m = \frac{V_0}{\beta} \frac{dP_1}{dt} \end{aligned} \tag{6.22}$$

where subscript p denotes 'pump', V_0 is the volume of forward chamber and β is the bulk modulus of system. The torque balance equation gives:

$$T_g = S_m \cdot (P_1 - P_2) = J_t \cdot \dot{\omega}_m + B_m \cdot \omega_m + T_l \tag{6.23}$$

where J_t is the total inertia of the motor and the load.

In the setup from Fig. 6.3(b), we have one pump with variable displacement driving two motors with variable displacement. The pump is driven by an engine which is speed controlled. Thus, the above analysis can be directly extended now to account for the two motors. It follows from (6.22) that the combined continuity equation for the high pressure line, P_1 becomes:

$$\begin{aligned} S_p \cdot \omega_p - C_{ip} \cdot (P_1 - P_2) - C_{ep} \cdot P_1 - 2 \cdot C_{im} \cdot (P_1 - P_2) - 2 \cdot C_{em} \cdot P_1 \\ - S_{m1} \cdot \omega_{m1} - S_{m2} \cdot \omega_{m2} = \frac{V_0}{\beta} \frac{dP_1}{dt} \end{aligned} \tag{6.24}$$

Subsequently, for the low pressure line, P_2 we have:

$$\begin{aligned} S_{m1} \cdot \omega_{m1} + S_{m2} \cdot \omega_{m2} - 2 \cdot C_{im} \cdot (P_1 - P_2) - 2 \cdot C_{em} \cdot P_1 \\ - C_{ip} \cdot (P_1 - P_2) - C_{ep} \cdot P_1 - S_p \cdot \omega_p = \frac{V_0}{\beta} \frac{dP_2}{dt} \end{aligned} \tag{6.25}$$

The torque balance equations at the two hydromotors are:

$$\begin{aligned} S_{m1} \cdot (P_1 - P_2) &= J_{t1} \cdot \dot{\omega}_{m1} + B_{m1} \cdot \omega_{m1} + T_{l1} \\ S_{m2} \cdot (P_1 - P_2) &= J_{t2} \cdot \dot{\omega}_{m2} + B_{m2} \cdot \omega_{m2} + T_{l2} \end{aligned} \tag{6.26}$$

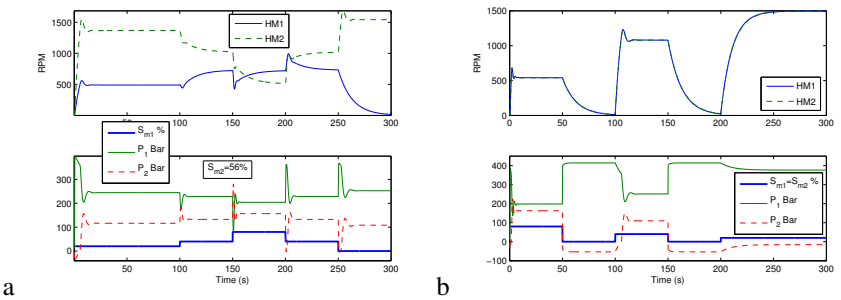
where subscripts $1, 2$ denote the first and second hydromotors respectively. Finally, the torque balance equation for the driving electric motor is:

$$T_{DrEM} = (J_{Dr} + J_p) \cdot \dot{\omega}_p + S_p \cdot (P_1 - P_2) \tag{6.27}$$

The equations (6.24), (6.25), (6.26) and (6.27) define the model of the hydrostatic drivetrain.

6.3.2 Open loop tests

The experimental setup as shown in Fig. 6.3(b) consists of: i) a speed controlled driving motor, ii) two torque controlled load motors, and iii) a hydraulic pump attached to the engine which is connected to the two hydromotors via flywheels. The stroke of the pump is fixed to 40% and the speed of the engine to 1200 rpm. The objective is to regulate the hydromotor speeds, reject load disturbances and adapt to effects of varying load. Input constraints are set on the actuators i.e. the stroke of the two hydromotors must be between 12% to 100%.



a b

Figure 6.4: (a): Open loop test on the hydrostat model demonstrating the coupled dynamics and (b): the presence of significant nonlinearity.

In order to access the characteristics of this system, we use the above derived system equations together with the parameters obtained from the real machine to demonstrate the presence of coupled and nonlinear dynamics. In the first test, the displacement volume of the second hydromotor is fixed and that of the first hydromotor is varied stepwise. As shown in Fig. 6.4(a), changing the displacement volume of the first hydromotor has almost equal, but opposite influence on the speeds of both hydromotors. This observation suggests that the hydrostat is composed of highly coupled subsystems.

In the second test, both hydromotor displacement volumes are changed stepwise. Since both hydromotors are assumed to have the same physical behavior, the effects of nonlinearity will be more pronounced if the directions of the simultaneous variation in the inputs are the same. As depicted in Fig. 6.4(b), the step changes in the hydromotor displacement volumes produce significantly different dynamics in the hydromotor speeds. These differences are in terms of varying gain, damping coefficients and time constants, all depending on the operating point. This observation suggests the presence of significant nonlinearities in the global system.

6.3.3 Closed loop tests

In order to explicitly demonstrate the superiority of the algorithm 9 in terms of performance and computation, a centralized NEPSAC controller is designed with $N_1 = 1, N_u = 1, N_2 = 5, \Lambda = 10 \cdot I$. Note that, a $N_u = 1$ for each of the $n_u = 2$ control inputs, results in an optimization over $n_u * N_u = 2$ variables. Our target is now to show that two distributed NEPSAC controllers with the same parameters $N_1 = 1, N_u = 1, N_2 = 5, \Lambda = 10$ approaches the centralized performance which is the best that can be achieved as all information is available. The

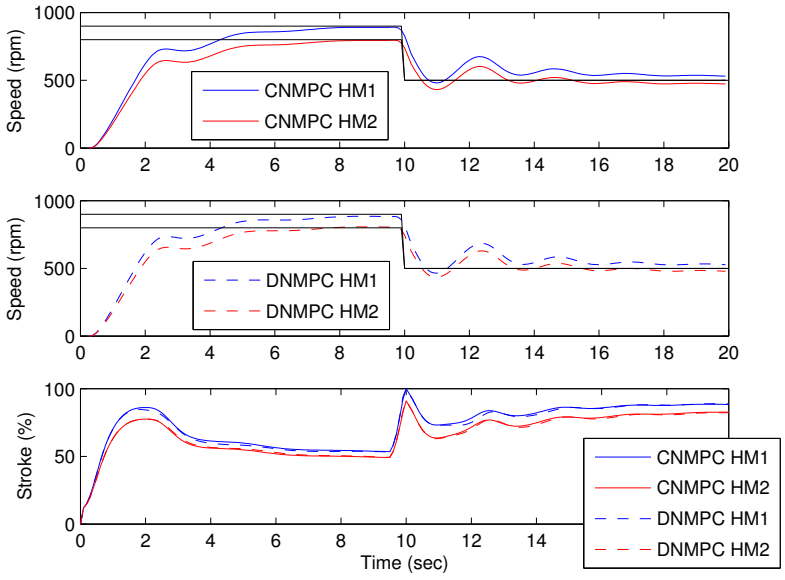
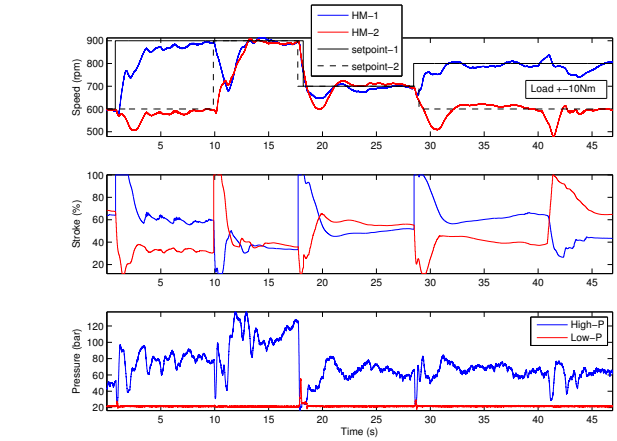


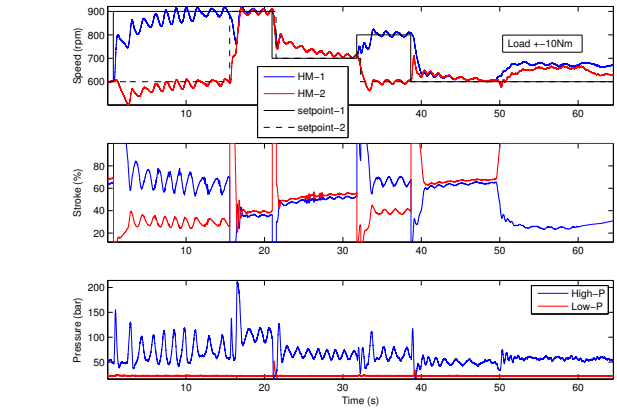
Figure 6.5: A comparison of centralized and distributed NMPC for tracking

DNMPC controllers communicate sequentially and only once as in algorithm 9. The control penalty is adapted to a very high value whenever the cost does not decrease monotonically, essentially freezing the new control to the past value. The closed loop is sampled at 100ms and the results are plotted in Fig. 6.5. As can be inferred, there is no difference to the naked eye between the performance of the CNMPC and DNMPC tracking controllers with the controls almost overlapping each other. This is a validation of theorem 6.2.1 and lemma 6.2.1 as the distributed locally optimal solution approaches the centralized one. The graphical verification is given in Fig. 6.2, where it can be seen that the first steps are enough to guarantee sufficient decrease in the gradient, which is in fact the case here.

Next, the computational costs for both the methods are compared. The maximum time, average time required to perform all the computations within one sampling time are 500ms, 20ms for CNMPC and 50ms, 2ms for DNMPC on embedded Matlab for real-time target. This clearly shows that as the peak computation time for CNMPC 500ms is greater than the sampling time of 100ms, it cannot be used in practice. However, the DNMPC controller is well within the limits even in the worst case.



a



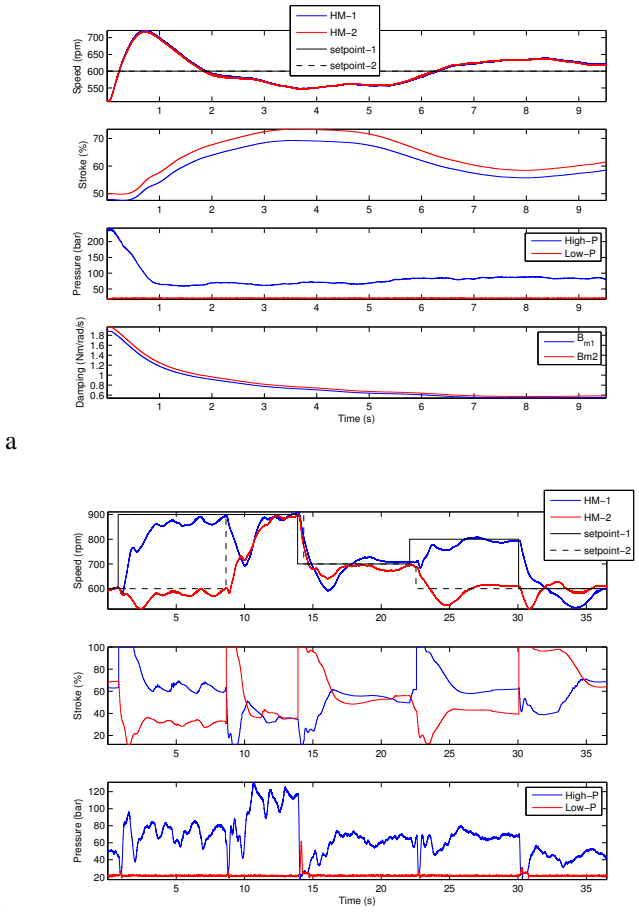
b

Figure 6.6: (a): Tracking performance and disturbance rejection by distributed NMPC and (b): PID control.

6.3.4 Experimental Results

The hydrostatic drive test setup is equipped with 150kW power hydromotors and pump. The controllers are developed in matlab and run in real-time on a dSPACE setup which interfaces with the setup. The nominal settings correspond to temperature of $50°\text{C}$, load torques $T_{l1}, T_{l2} = 20\text{Nm}$, dampings $B_{m1} = B_{m2} = 0.55\text{Nm/rad/s}$, intertias $J_p + J_{DR} = 0.7\text{kgm}^2, J_{t1} = J_{t1} = 6.5\text{kgm}^2$, and $\beta/V_0 = 1.2 \cdot 10^{11}\text{Pa/m}^3$.

First, we present the design of two PID loops, which would be used for com-



a

b

Figure 6.7: (a): Learning the true motor damping coefficients by the distributed RLS method and (b): Robust performance of the distributed NMPC after learning the correct damping coefficients.

parison purposes. It is clear from the analysis in section 6.3.2 that there exists no unique relation between the stroke of one hydraulic motor and its speed. Therefore, it is impossible to determine steady-state gains necessary for classic PID tuning methods. A transformation is thus required that decouples significantly the hydromotor interaction and is discussed hereafter.

One can rewrite the flow equation (6.24) (assuming incompressible fluid and negligible leakage) as:

$$S_p \cdot \omega_p = (S_{m1} + S_{m2}) \cdot (\omega_{m1} + \omega_{m2}) + (S_{m1} - S_{m2}) \cdot (\omega_{m1} - \omega_{m2}) \quad (6.28)$$

Ignoring the second term yields:

$$d(\omega_{m1} + \omega_{m2})/d(S_{m1} + S_{m2}) < 0 \tag{6.29}$$

Since the pressures across the pump and the two motors are equal, we have:

$$d(S_{m1}/S_{m2}) = d(T_{g1}/T_{g2}) = (T_{g2} \cdot dT_{g1} - T_{g1} \cdot dT_{g2})/T_{g2}^2 \tag{6.30}$$

where T_{g1}, T_{g2} are the torques of the two hydromotors. Assuming positive torques in steady state, if S_{m1}/S_{m2} increases, then either T_{g1} increases or T_{g2} decreases. Further at the flywheels, $dT_{l1}/d\omega_{m1} > 0$ and $dT_{l2}/d\omega_{m2} > 0$. This leads to the resulting steady state relation:

$$d(\omega_{m1}/\omega_{m2})/d(S_{m1}/S_{m2}) > 0 \tag{6.31}$$

The relations (6.29) and (6.31) concur that two independent PID controllers can be designed with one controlling the sum of hydromotor speeds with sum of hydromotor strokes and the other controlling the ratio of hydromotor speeds with ratio of hydromotor strokes. The PID controllers are defined on the sum and the ratio error signals. The controller outputs are then re-transformed to the hydromotor strokes through negative exponentials (as it more or less neutralizes the nonlinear stroke-speed relation). This is indeed by far the most effective PID design, as the others which fail to induce any decoupling make the system unstable and are not safe for real tests.

Two identical DNMPC controllers, without terminal conditions have been designed with prediction horizon $N_1 = 1$ to $N_2 = 5$, control horizon $N_u = 1$, control penalty $\Lambda = 10$, an integrator as disturbance filter i.e. $C/D = 1/(1 - q^{-1})$. The embedded DRLS uses a forgetting factor of 0.99 and the closed loop is sampled at $T_s = 100$ms. The DNMPC uses a cooperative cost function and exchange the optimal trajectories once every sampling period. The number of iterations in the nonlinear EPSAC algorithm within the DNMPCs have been restricted to 1 iteration per sampling period.

During this first closed loop test, variations in the speed set-points are tested and, as a disturbance, the load torque values are varied within $\pm 10\mathrm{Nm}$. The performance of the DNMPC is given in Fig. 6.6(a). For comparison purpose, the same experiment has been evaluated with the PID controllers and the results given in Fig. 6.6(b). It can be observed that the DNMPC outperforms the PID control, generating a much smoother control action compared to the rather oscillatory response of the PID control strategy. The net oil flow, which is the product of stroke and angular velocity, must remain constant. This implies that if both speeds go up, the strokes go down (similarly for reverse sign). The load change is nicely compensated by the DNMPC, while significant offsets are present in case of PID control. Both in nominal and perturbed load settings, the DNMPC controller manages a settling time < 10s with a rise time < 5s whereas the PID though has the

same rise time of < 5s, the settling time $\rightarrow \infty$ because of sustained oscillations in the nominal case and the PID produces a big steady state error in the perturbed case i.e. after 50s.

In the next experiment, the controllers have been initialized with erroneous motor damping coefficients, i.e. $1.9\mathrm{Nm/rad/s}$. Given that damping varies exponentially with temperature, the ability to continuously learn the right damping is of high practical importance. As shown in Fig. 6.7(a), the distributed RLS algorithm converges to the true value of $0.55\mathrm{Nm/rad/s}$ in a reasonable period of time. Consequently, the performance of the controller in terms of settling time and rise time (plotted in Fig. 6.7(b)) remains comparable to the nominal case, demonstrating robustness of the proposed control strategy.

6.4 Summary

The global control of large scale production machines composed of interacting subsystems is a challenging problem due to the intrinsic presence of high coupling, constraints, nonlinearity and communication limitations. In this work a pragmatic approach to distributed nonlinear model predictive control (DNMPC) is presented. The main contribution has been to ensure guaranteed improvement in the cost function by the DNMPC without terminal conditions, even in the limiting non-iterative case and a 10 fold reduction in computation time for nonlinear non-convex problems. Furthermore, in order to tackle time-varying process dynamics, a learning algorithm is developed, thereby improving the performance of the global control.

The proposed control framework is experimentally validated on a hydrostatic drivetrain which exhibits nonlinear dynamics through strongly interacting subsystems. The experimental results indicate that good tracking performance and disturbance rejection can be obtained by the proposed DNMPC. This indicates that the DNMPC technique has great potential in controlling fast, coupled, nonlinear dynamical systems like agricultural and industrial machines, power grids, production lines, wind mills etc.

7 Concluding Remarks and Perspectives

In conclusion, the main contributions of this thesis are summarized and suggestions for possible future directions are outlined.

7.1 Contributions

The central idea behind this thesis was (i) to develop predictive controllers (linear, nonlinear and distributed) without terminal conditions and with very short control horizons (different from prediction horizons) for industrial applications and subsequently (ii) certify practical stability through infinite time constraint satisfaction (feasibility), (iii) guarantee improvement in cost. The main contributions of this thesis are summarized below.

7.1.1 Linear MPC without Terminal Conditions

- An equivalence between input-output industrial MPC formulation (EPSAC) and Diophantine analysis based MPC synthesis and state-space based MPC formulation and its explicit solution is established. The three great techniques are brought together for obvious reasons: EPSAC algorithm is easy to implement, it is then analyzed with Diophantine equations to get closed-form solution in the unconstrained case and transformed to the state-space MPC in the constrained case to synthesize stability certificates.

- The powerful theory of set invariance is presented in a consolidated fashion

and the important tools like robust controllable sets and reach sets have been highlighted. A new set which we call the robust tunnel set is introduced to construct feasibility set for the EPSAC MPC controller with very short control horizon compared to prediction horizon.

- The a posteriori certification of practical stability of the EPSAC controller without terminal conditions has been approached in two ways: (1) by guaranteeing persistent feasibility i.e. infinite time constraint satisfaction without optimality of the solution i.e. through control invariance of the feasibility set (2) by guaranteeing recursive feasibility i.e. infinite time constraint satisfaction based on optimality and thereby requiring the explicit solution i.e. through positive invariance of the feasibility set. Subsequently new rules which specify the control and prediction horizons for guaranteeing practical stability of the closed-loop have been derived. A longitudinal flight control scenario is used for demonstration.

- A more strict condition of nominal asymptotic stability without terminal conditions has been achieved through penalty adaptation in the input constrained case for repetitive systems. This we introduced as PAMPC and subsequently robust feasibility was improved through tunneling of the constraint sets. PAMPC was shown to be effective on a benchmark mass-spring-damper system.

7.1.2 Nonlinear MPC without Terminal Conditions

- The NEPSAC algorithm is used as the NMPC controller without terminal conditions and a proof of guaranteed convergence has been derived for the NEPSAC controller based on Levenberg-Marquardt algorithm.

- The predominant form of nonlinearity in the industry occurs as switching either between linear well-defined partitions (PWA) or between completely different dynamics (switched nonlinearity). The results on persistent feasibility and recursive feasibility have been extended for the PWA case without terminal conditions and demonstrated over position control of a car.

- A two-level NMPC has been introduced to handle the switched nonlinearity which consists of a high-level learning controller that generates references to be tracked by low-level (N)MPC controllers depending on (non)linear dynamics. This has been demonstrated for the engagement control over a real wet-clutch test-bench in nominal and perturbed settings.

7.1.3 Distributed NMPC without Terminal Conditions

- A distributed NMPC controller based on multiple NEPSAC controllers minimizing a cooperative cost function (i.e. the sum of the individual costs) without terminal conditions is proposed for fast, highly coupled, nonlinear dynamical systems. A proof of convergence together with a guarantee on cost decrease with every iteration is given based on coordinate descent algorithm. A distributed RLS algorithm is used together with the DNMPC for systems with changing parameters. The scheme has been shown to be effective in controlling a test setup consisting of a highly coupled nonlinear hydrostatic drivetrain.

7.2 Directions for future research

Some possible directions for future research are outlined below.

7.2.1 Linear MPC without Terminal Conditions

- The stability guarantee which has been developed is in terms of boundedness as a result of persistent feasibility. It would be interesting to see if the certification without terminal conditions could be extended to asymptotic stability.

- Robust stability has been explored without terminal conditions, hence a direct step forward is to guarantee robust performance together with stability.

7.2.2 Nonlinear MPC without Terminal Conditions

- The nonlinear MPC has been explored in terms of switching systems. So, there is room to make contributions for other classes of nonlinear systems in terms of proving theoretical properties of the associated NMPC without terminal conditions.

- A two-level NMPC setup has been proposed to handle plantwide objectives, however, the two-levels could be merged to a single layer giving rise to the fascinating subject of economic NMPC. A posteriori feasibility and stability guarantee in economic NMPC without terminal conditions is still an open question.

7.2.3 Distributed NMPC without Terminal Conditions

- Strong convergence proof has been given in this thesis for DNMPC. This prepares the groundwork to extend the presented set invariance techniques

to distributed NMPC without terminal conditions for rigorous certification of practical and/or asymptotic stability.

- Finally, economic DNMPC without terminal conditions includes all the possible ingredients foreseeable at this moment, which we believe is a framework that would remain relevant with enormous potential for the industry. Therefore future research should be directed towards certifying stability and performance for economic DNMPC.

A

MPC Certification Algorithm

The three building blocks for the MPC certification procedure presented in chapter 3 are the pre set $Q(\mathbb{X})$, reach set $R(\mathbb{X})$ and the i-step positively invariant set $O_i(\mathbb{X})$. Now, consider a linear state-space system in its nominal form with dynamics:

$$x(t+1) = \mathcal{A} \cdot x(t) + \mathcal{B} \cdot u(t) \tag{A.1}$$

subject to the constraints:

$$u(t) \in \mathbb{U} \quad x(t) \in \mathbb{X}. \tag{A.2}$$

Then the pre set as defined in 5.12(b) can be computed as follows:

$$Q(\mathbb{X}) = \mathcal{A}^{-1}(\mathbb{X} \oplus -\mathcal{B} \cdot \mathbb{U}) \tag{A.3}$$

The reach set as defined in 5.12(b) can be computed as follows:

$$R(\mathbb{X}) = \mathcal{A} \cdot \mathbb{X} \oplus \mathcal{B} \cdot \mathbb{U} \tag{A.4}$$

The positively invariant set is defined in 5.12(b). The i-step positively invariant set is the set of states that remain positively invariant for i-steps. In other words, the state trajectories must respect the constraints for i-steps. Let the constraint set be polytopic i.e.

$$\mathbb{X} \equiv Z_l \cdot x(t) \leq z_r. \tag{A.5}$$

Note that now we consider an autonomous system, hence there is no input. Then the i-step positively invariant set deduces to:

$$O_i(\mathbb{X}) \equiv \begin{bmatrix} Z_l \\ Z_l \cdot \mathcal{A} \\ Z_l \cdot \mathcal{A}^2 \\ .. \\ Z_l \cdot \mathcal{A}^i \end{bmatrix} \cdot x(t) \leq \begin{bmatrix} z_r \\ z_r \\ z_r \\ .. \\ z_r \end{bmatrix} \tag{A.6}$$

Now, in order to certify a MPC controlled constrained linear system without terminal conditions, the following need to be known:

1. A nominal input-output model ($n(t) = 0$):

$$\hat{y}(t) = \frac{q^{-d}B(q^{-1})}{A(q^{-1})}u(t). \tag{A.7}$$

2. The associated input/output constraints:

$$u(t) \in \mathbb{U} \quad y(t) \in \mathbb{Y}. \tag{A.8}$$

3. The control and prediction horizons N_u, N_2 respectively.

The steps involved in the certification procedure is presented as algorithm 10. Note that, the methodology is independent of the exact cost function being used in the MPC problem.

1. Convert the input-output model of (A.7) to state-space form of (A.1) using corollary 2.4.2.
2. Map the input/output constraints of (A.8) to state constraints of (A.2) using corollary 2.4.4.
3. Compute the tunnel set $L_{N_2-N_u}(\mathbb{X})$ as follows:
   (a) Construct the augmented autonomous system using lemma 3.2.1.
   (b) Compute the $i = N_2 - N_u$-step positively invariant set $O_{N_2-N_u}(\mathbb{X})$ for the augmented system using (A.6).
4. Compute the pre set $Q(\mathbb{X})$ using (A.4) as preparation for computing the i-step controllable sets $K_i(\mathbb{X})$.
5. Construct the feasibility set $X_F(\mathbb{X}, N_u, N_2)$ as $K_{N_u}(\mathbb{X}, L_{N_2-N_u}(\mathbb{X}))$ by plugging in $Q(\mathbb{X})$ into algorithm 1.
6. Construct the next feasibility set $X_F(\mathbb{X}, N_u - 1, N_2 - 1)$ as $K_{N_u-1}(\mathbb{X}, L_{N_2-N_u}(\mathbb{X}))$ in the same way.
7. Compute the reach set of next feasibility set i.e. $R(X_F(\mathbb{X}, N_u - 1, N_2 - 1))$ using (A.5).
8. Evaluate the subset test: $R(X_F(\mathbb{X}, N_u, N_2)) \cap X_F(\mathbb{X}, N_u - 1, N_2 - 1) \subseteq X_F(\mathbb{X}, N_u, N_2)$.
9. A positive result certifies the MPC controlled loop as persistently feasible and stable.

algorithm 10: Certification algorithm for a MPC controlled constrained linear system

B

Model-based and Model-free Learning Control Strategies

This appendix is a result of collaborative work done jointly with FMTC, KUL and VUB under the framework of the IWT SBO funded LeCoPro project. In particular, the controllers developed in B.2, B.3 are due to KUL, VUB respectively and the development and maintenance of the experimental setup in B.4 is credited to FMTC.

B.1 Introduction

Next to NMPC, other model-based: Iterative Learning Control (ILC), Iterative Optimization (IO) and model-free: Reinforcement Learning (RL), Genetic Algorithm (GA) learning strategies have great potential in the control of repetitive systems where reference trajectories cannot be readily generated from control specifications and the trajectories themselves change with operating conditions.

The model-based approaches rely on a model of the system dynamics to update the control signals after each trial, while in contrast, the model-free ones omit this model and directly explore the input space of possible control signals using a guided trial-and-error procedure, attempting to maximize the reward/fitness. The clutch control problem is considered as the benchmark and the objective remains the same as in 5.6 i.e. to find the control yielding the lowest absolute jerk for a given engagement duration.

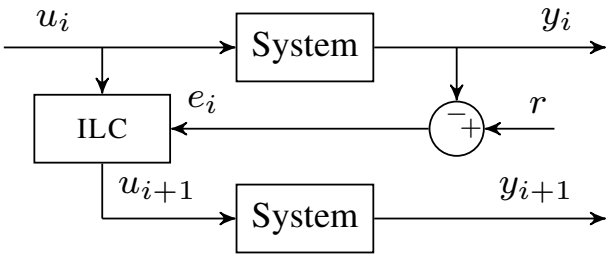


Figure B.1: Schematic representation of an ILC controller; first, during trial i, the system to be controlled is excited using u_i, then after completion of this trial, u_i is used along with the measured tracking error $e_i = r - y_i$ to find the excitation u_{i+1} to be used during trial $i + 1$.

B.2 Model-Based Learning Control

This section discusses two model-based learning techniques for wet clutch control. Next to the two-level learning control scheme presented based on NMPC in 5.6, a similar two-level control scheme using ILC is developed instead of NMPC. Afterwards, the IO technique is presented as an alternative.

B.2.1 Two-Level ILC (2L-ILC)

In this section, a similar approach is used as in the 2L-NMPC (refer Fig. 5.8, but the low-level controllers are now ILC instead of NMPC controllers. The reason for doing so it that ILC is itself a learning control technique, unlike NMPC, which uses experience gained during previous iterations to improve the tracking performance for repetitive systems [112, 113]. An NMPC thus requires an accurate model to be able to obtain a good tracking performance, whereas an ILC algorithm can, due to its learning strategy, realize a good tracking performance even when there is a large model uncertainty due to its learning behaviour. The downside of this is that we will not be able to update the reference profile parameters after each trial, since we cannot yet judge the parameter's quality since the ILC controller has not yet learned to track the profile closely. Instead, we wait for 5 trials now each time the parameters are updated, allowing the ILC to converge before we calculate the performance indices and update the reference parameters.

Fig. B.1 shows a first order ILC control scheme, as is used in this paper. Here, y is the output of the plant and r is a reference trajectory. The ILC control signal for the $(i + 1)^{th}$ iteration, u_{i+1}, is calculated based on the previous ILC control signal, u_i and the previous tracking error e_i. We use a linear update law such that

$$u_{i+1}(k) = Q(q^{-1})\Big(u_i(k) + L(q^{-1})e_i(k)\Big), \tag{B.1}$$

with linear operators Q and L that can be chosen during the design of the ILC controller. For this update law, a convenient frequency domain criterion for monotonic convergence can be derived [112, 113]. For a plant with FRF $P(\omega)$, a monotonically decreasing tracking error is obtained with controller (B.1) if

$$|Q(j\omega)\Big(1 - L(j\omega)P(j\omega)\Big)| < 1, \tag{B.2}$$

with $Q(j\omega)$ and $L(j\omega)$ the FRF's of the operators Q and L. It is also possible to derive an expression for the remaining error after convergence, $E_\infty(j\omega)$, which becomes

$$E_\infty(j\omega) = \frac{1 - Q(j\omega)}{1 - Q(j\omega)\Big(1 - L(j\omega)P(j\omega)\Big)} R(j\omega), \tag{B.3}$$

where $R(\omega)$ is the Fourier transform of the reference r.

Based on these expressions, [112] and [113] show that by selecting $L(j\omega) = P(j\omega)^{-1}$ and $Q(j\omega) = 1$, perfect tracking would be obtained after only one iteration. However when there is uncertainty about $P(j\omega)$, this choice of $L(j\omega)$ becomes impossible. It is then needed to select an estimate $\hat{P}(j\omega)$ of the plant and use $L(j\omega) = \alpha\hat{P}(j\omega)^{-1}$ with $0 < \alpha < 1$. This way, the robustness increases while the learning slows down, but a good performance is still achieved. This is possible for all frequencies where the angular deviation between the system and the nominal model does not exceed $90°$. Once this deviation becomes larger, the value of $|Q(j\omega)|$ has to be decreased in order to satisfy (B.2). It then follows from (B.3) that perfect tracking can no longer be achieved, not even by learning more slowly. As the uncertainty typically increases with the frequency, $Q(j\omega)$ is often chosen as a low pass filter, effectively deactivating the ILC controller for high frequencies with much uncertainty, while obtaining good tracking in the less uncertain, lower frequency range.

Since an accurate plant model is not required to achieve a good tracking performance, ILC is well suited to the control of wet-clutch engagements, where the plant dynamics are non-linear and vary significantly over time. For each of the two ILC controllers, a single, linearized model, approximating the plant dynamics in all conditions suffices, keeping the required modeling effort small. With these choices it becomes possible to design ILC controllers that achieve bandwidths of > 10Hz, but in practise the controllers are detuned intentionally. Especially in the slip phase this is needed, as it is preferable to keep the jerk low instead of aggressively tracking the reference. A more detailed description of the implementation applied to the wet clutch can be found in [114].

B.2.2 Iterative Optimization

Another technique based on learning control, denoted iterative optimization, has been developed as an alternative to the two-level ILC approach. It is aimed at (i) reducing the amount of time needed before a good performance is obtained, at (ii) allowing an easier adaptation mechanism to deal with changes in the oil temperature and the load, and at (iii) removing the dependence on the parameterization of the reference profile. The method should however not require accurate models, as was the case for two-level NMPC scheme, but should instead operate using simplified linear models. To achieve this result, it is useful to note that the two-level ILC scheme learns in an indirect manner. At the low level learning takes place, but there the only goal is to accurately track a reference so this reference's quality can be evaluated, even though this reference's parameters are likely to change at the high level so that the low learning will have to be restarted. This is a consequence of the fact that the task itself does not deal with tracking at all, but was formulated in such a manner to be able to use classical tracking-based ILC techniques. The idea in iterative optimization is to omit this indirect approach, and to directly learn based on the specifications themselves. Solving the problem more directly will reduce the convergence period and reduce the effort needed to adjust to varying conditions, and since this effectively removes the parameterized reference, this will also remove the dependence on the chosen parameterization.

The resulting IO technique again uses a two-level control scheme, but learning is now only included at the high level. At the low level, a numerical optimal control problem is solved, formulated directly from the specifications. Since it is in general very difficult to accurately solve such a problem without a large amount of prior information, some constraints can be included whose exact value in order to reach optimality is initially unknown, and afterwards learning laws can be added at the high level to find appropriate values for these constraints, based on the results observed using the control signals calculated with the current values. Besides these laws, the high level also contains algorithms for a recursive model estimation to describe the system dynamics, combining previously estimated models with the newly measured data. A schematic overview of this control scheme is presented in Fig. B.2, where it can be seen that an optimal control problem is essentially solved before each trial, using models and constraints that are adapted after completion of each trial based on the observed performance.

When applying this method to the clutch a numerical optimization problem has to be formulated and solved, and we will again try to separate the two phases, although a single large optimization is solved. First, in the filling, the goal is to advance into to the slip phase as soon as possible, without causing unwanted torque spikes that could cause operator discomfort. Afterwards, once the slip phase begins, the goal becomes to further engage the clutch while keeping the jerk as low as possible. To optimize the control signals accordingly, a piece-wise linear model

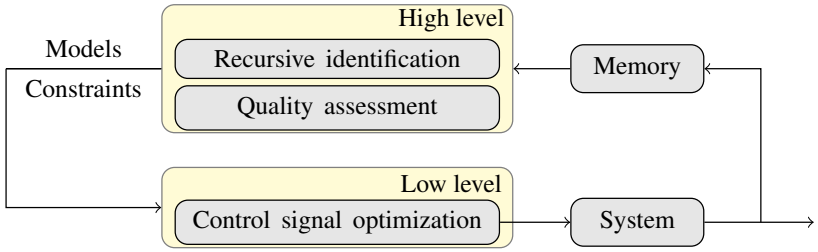


Figure B.2: Two-level iterative optimization control scheme: At the high level, the models and constraints for the optimization problem are updated after each engagement, which are then used at the low level to optimize the control signal for the next engagement.

structure is selected, with one model to predict the pressure and piston position in the filling phase and one to predict the pressure and the slip in the slip phase, while recursive estimation techniques are added to learn these online, so that these models are tuned to the observed behaviour. Transition constraints are also added to ensure a smooth transition occurs between both phases, but since the optimal conditions in which to go from the filling to the slip are unknown, values for these are chosen and afterwards their optimal values are found using learning laws. Using the notation that $\dot{z}(k)$ denote the discrete time finite difference $(z(k+1)-z(k))/T_s$ with T_s the sampling time, the problem to be solved at the low level is then:

$$\min_{\substack{u(:),\\ x(:),p(:),s(:),\tilde{z}(:),\\ j_{\max},K_1,K_2}} \quad K_1 + \gamma * j_{\max}, \tag{B.4a}$$

s.t.

filling phase: $k = 1 : K_1$

$$\mathbf{x}(k+1) = \mathbf{A_1}\mathbf{x}(k) + \mathbf{B_1}u(k), \tag{B.4b}$$

$$\begin{pmatrix} p(k) \\ \tilde{z}(k) \end{pmatrix} = \mathbf{C_1}\mathbf{x}(k) + \mathbf{D_1}u(k), \tag{B.4c}$$

$$u_{\min} \leq u(k) \leq u_{\max}, \tag{B.4d}$$

$$p_{\min} \leq p(k) \leq p_{\max}, \tag{B.4e}$$

slip phase: $k = K_1 + 1 : K_1 + K_2$

$$\mathbf{x}(k+1) = \mathbf{A_2}\mathbf{x}(k) + \mathbf{B_2}u(k), \tag{B.4f}$$

$$\begin{pmatrix} p(k) \\ s(k) \end{pmatrix} = \mathbf{C_2}\mathbf{x}(k) + \mathbf{D_2}u(k), \tag{B.4g}$$

$$0 \leq s(k) \leq s_{\text{trans}}, \tag{B.4h}$$

$$-j_{\max} \leq \ddot{s}(k) \leq j_{\max}, \tag{B.4i}$$

transition and terminal constraints:

$$\mathbf{x}(K_1+1) = \mathbf{x}_{\text{trans}}, \tag{B.4j}$$

$$p(K_1) = p_1, \tag{B.4k}$$

$$\tilde{z}(K_1) = z_{\text{final}}, \qquad \dot{\tilde{z}}(K_1) \leq \epsilon, \tag{B.4l}$$

$$s(K_1+K_2) = 0, \quad \dot{s}(K_1+K+2) = 0 \tag{B.4m}$$

In this problem, the piecewise structure can clearly be seen, as the problem is split into two parts with K_1 and K_2 samples for each phase respectively (with $K_1 + K_2 = T/T_s$), and a set of constraints which need to be respected during the transition. In order for the solutions of this problem to yield good engagements, the high-level learning laws recursively identify the matrices $\mathbf{A_i}$, $\mathbf{B_i}$, $\mathbf{C_i}$ and $\mathbf{D_i}$. Since the piston position is not measured, its model $\tilde{z}$ can however not be estimated so easily, so here we use a simple first principles model and rescale it using a rule similar to (5.21). Similar rules are included for $\mathbf{x}_{\text{trans}}$, p_1 and z_{final}. A more detailed description of the implementation can be found in [115].

The control techniques presented so far are based on certain characterization of the system dynamics. In very complex systems, however, another approach could be to focus entirely on improving the performance without the intermediate step of modeling the system. Two representative techniques which fall in this category are discussed next.

B.3 Model-free Learning Control

Next to the model-based algorithms described so far in section B.2, the potential of model-free algorithms has also been investigated. To date, most complex mechatronic systems are controlled using either a model-based technique, or using controllers tuned during an experimental calibration. Even though these latter are often tuned without the use of a model, this tuning is usually done in an ad-hoc manner derived from system knowledge or insight, and a systematic model-free machine learning (ML) strategy is rarely applied. These strategies would however make it possible to also learn controllers for more complex situations, where insight would not be sufficient to yield the desired behavior. This can improve the current controllers by being able to use more complex control laws, or allowing to optimize a cost criterion and taking into account constraints. This can further also make it possible to develop controllers for more complex applications, for which now no good controllers can be tuned automatically.

Nowadays, wet clutches in industrial transmissions are filled using a feed forward controller of the current (with a set of tunable parameters) to the electro-hydraulic valve. These are now tuned using some heuristic rules, but now we will use model-free learning control methods instead, while still looking for optimal parameterized control signals.

B.3.1 Genetic Algorithm

Genetic Algorithm (GA) is a stochastic search algorithm that mimics the mechanism of natural selection and natural genetics, and belong to the larger class of evolutionary algorithms (EA). They are routinely applied to generate useful solutions for optimization and search problems, often for complex non-convex problem where gradient-based methods fails to find the correct solution. One of the main strengths of GA is that multi-objective optimization problems [116, 117] can be studied.

Unlike conventional optimization techniques, GA starts with an initial set of random solutions (satisfying the boundary and/or system constraints though), called the population. Each individual in the population is called a chromosome, which represents a possible solution to the implementation. Usually, a chromosome is a string of symbols, but not necessarily is a binary bit string. The idea of a GA is that the chromosomes evolve through successive iterations called generations, and converge towards the solution. To achieve this, the chromosomes are evaluated throughout their evolution by a function to obtain a fitness value. Once a complete generation is evaluated, the next generation, with new chromosomes called offspring, are formed by i) copying from the parents using a reproduction operator; ii) merging two chromosomes from current generation using a crossover operator; iii) modifying a chromosome using a mutation operator [118]. The selection of

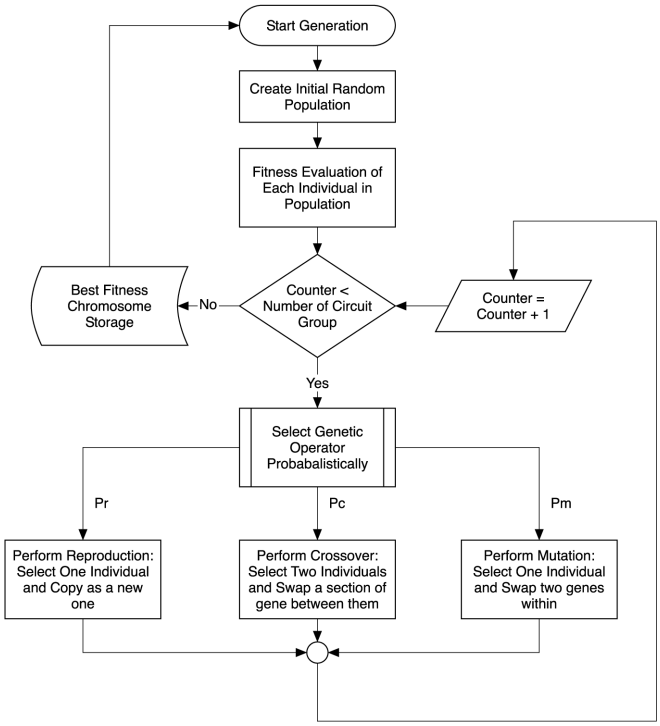


Figure B.3: General structure of a genetic algorithm

which parents' chromosomes will be used is based on the fitness values, with fitter chromosomes having a higher probability of being selected. Fig. B.3 illustrates how a generation is used to define the next one in simple genetic algorithm [119].

For the application to the clutch, each chromosome contains values of the parameters of the parameterized control signal that is applied to engage the clutch. It contains five variables for tuning as shown in Fig. B.4. First, a step signal with maximum height and width d_1 is sent to the valve to generate a high pressure level in the clutch. With this pressure, the piston will overcome the force from the return spring, and start to get closer to the clutch disks. After this pulse, the signal will give some lower current with fixed height and width to decelerate the piston and try to position it close to the clutch disks. Once the piston is close to the clutch disks and with very low velocity, a force is needed to push the piston forward further, so that the clutch disks are compressed together. Since, the change between the fill phase and slip phase would happen within this period, by providing more freedom to the signal, better engagement performance can be achieved. As a result, two slopes are used to cover this critical period, defined by combination of h_1, d_2, h_2, h_3. Then a ramp current signal with fixed slope and the end height is

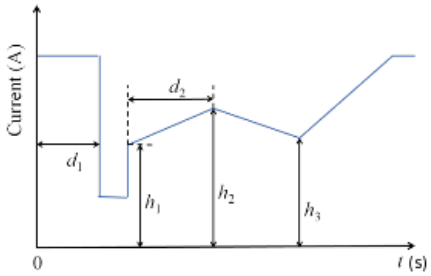


Figure B.4: Parameterized signal with five tunable parameters d_1, h_1, d_2, h_2, h_3, optimized by both GA and RL to obtain fast and smooth engagement

sent to the valve so that the pressure inside the clutch will increase again gradually. In order to secure the full closing of the clutch, the current signal will be kept constant at the end.

In each generation, each of the chromosomes is evaluated, which is done by applying the corresponding control signal experimentally to the clutch, and afterwards calculating a scalar reward to express the engagement quality. For the reward, we want a function that is monotonically decreasing with the maximum jerk. We could therefore choose it as $r(jerk) = e^{k_1(1-jerk/k_2)}$, with $k_2 = 5000$, corresponding to what can be considered a typical value for the jerk during an engagement. Regardless of the value of k_1, this will give a reward $r = 1$ for a jerk of $jerk = k_2 = 5000$, and rewards higher and lower than 1 for lower and higher jerk values. The constant k_1 controls the steepness of the reward, and we choose it as $k_1 = 5$. Next, to take into account engagement time, we discount the reward with a discount factor γ, so the overall reward is given by

$$r = \gamma^{ent}.e^{(5-jerk/1000)} \tag{B.5}$$

where the ent is the engagement time. Since γ is chosen as $\gamma = 0.8$, longer engagements will yield lower rewards than shorter engagements, so that to find the highest reward it is needed to do an engagement that is both smooth and fast, similar to our control objectives. A more detailed description of the implementation applied to the wet clutch can be found in [120].

B.3.2 Reinforcement Learning

RL problems [121] are a class of machine learning problems, where an agent must learn to interact with an unknown environment, using a trial and error approach. At a given timestep t, the agent may execute one of a set of actions, possibly causing the environment to change its state and generate a (scalar) reward. Both state and action spaces can be multidimensional, continuous or discrete. An agent

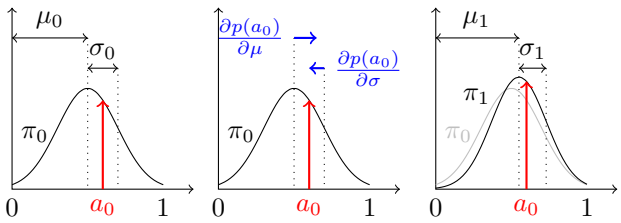


Figure B.5: A simple example illustrating the effect of one step of PGPE, with no state information and single stage epochs ($T = 1$). A single policy parameter $\mathcal{A} = [0, 1]$ is sampled from a Gaussian prior π, with $\theta = (\mu, \sigma)$. Left: *the first epoch is executed, drawing a parameter value $a_0 \sim \pi_0(a)$, and observing a return R_0.* Center: *as $R_0 > b$, following the gradient (B.6) increases $\pi(a_0)$.* Right: *updated prior π_1, ready for the next epoch.*

is represented by a policy, mapping states to actions. The aim of a RL algorithm is to optimize the policy, maximizing the reward accumulated by the agent.

In this work, we apply an existing variant of the basic Policy Gradient method [122], called Policy Gradients with Parameter-based Exploration (PGPE) [123]. In this approach, the parameters of a controller are adapted based on the return collected during the whole epoch, regardless of the trajectory in the state space. The advantage of using a direct policy search method is that it easily allows to use a policy that has been optimized on a simulated plant as a good starting point for learning to control the real plant. In the remainder of this section we briefly describe PGPE, referring the reader to [123, 124] for further details.

In Policy Gradients (PG) methods, the policy is represented as a parametric probability distribution over the action space, conditioned by the current state of the environment. Epochs are subdivided into discrete time steps: at every step, an action is randomly drawn from the distribution, conditioned by the current state, and executed on the environment, which updates its state accordingly. After an epoch has been completed, the parameters of the policy are updated, following a Monte Carlo estimate of the expected cumulative (discounted) reward.

A major disadvantage of PG methods is that drawing a random action at every timestep may result in noisy control signals, as well as noisy gradient estimates. Moreover, the policy is required to be differentiable w.r.t. its parameters. To overcome these issues, PG with Parameter-based Exploration (PGPE) was introduced [123, 125]. In this method, the random sampling and policy evaluation steps are, in a sense, 'inverted': the policy is a parametric function, not necessarily differentiable, therefore it can be an arbitrary parametric controller; the parame-

ter value to be used is sampled at the beginning of each epoch from a Gaussian distribution, whose parameters are in turn updated at the end of the epoch, again following a Monte Carlo estimate of the gradient of the expected return. In other words, rather than searching the parametric policy space directly, PGPE performs a search in a 'meta-parameter' space, whose points correspond to probability distributions over the (parametric) policy space.

To simplify notation, we consider a parametric policy f_a with a scalar parameter a. Be $\alpha = (\mu, \sigma)$ the meta-parameter defining the Gaussian distribution $p_\alpha(a)$ over parameter values. The index we intend to maximize is the expected value of the return R given a, $J = E\{R|a\}$. The gradient of this expected return J with respect to the metaparameter α is then estimated as follows (see [123] for details):

$$\nabla_\alpha J \approx \frac{1}{N} \sum_{n=1}^{N} \nabla_\alpha \log p_\alpha(a^n)(R^n - b), \tag{B.6}$$

where θ^n is the parameter used at the n-th of the N epochs considered (typically $N = 1$), and b is a *baseline* return, which, in the simplest case, is the average return observed so far. Based on this estimated gradient, the policy is then updated, as illustrated in Fig. B.5.

For application of PGPE to control of the wet clutch, choice of the policy and reward function are critical. We discard the state information entirely, and adopt an open loop approach, defining the five DOF control signal parameterized in d_1, h_1, d_2, h_2, h_3, as stated before in Fig. B.4 as the policy to be applied to the plant. Thus the RL problem is reduced to a simpler optimization problem, in which only the parameters of the control signal need to be optimized. To do so, we use a scalar reward function r as in (B.5) that favours both the objectives of fast and smooth engagement at once, which is the same as used by GA described in the previous section. Note that, the important thing is just that this reward function monotonically decreases in the objective which we intend to minimize. A more detailed description of the implementation applied to the wet clutch can be found in [124].

B.4 Experimental results

The two model-based and two model-free control techniques developed in B.2, B.3 respectively, they have been applied to the experimental setup described in section 5.7.4 to compare with the results obtained by the 2L-NMPC in 5.7.4. The objective remains uniform i.e. to keep the engagement time during slip within 1s and fill as fast as possible, such that the bound on jerk is minimized.

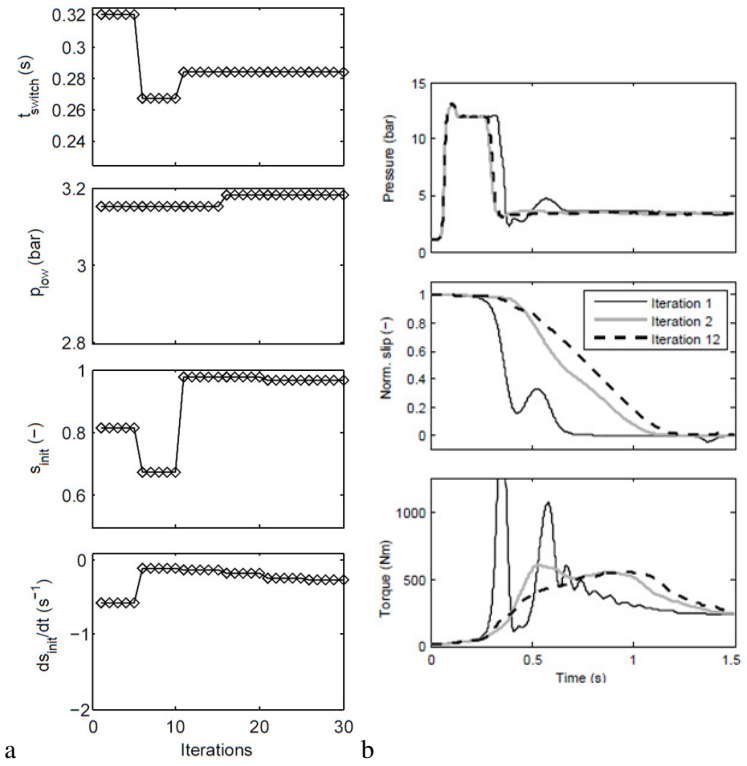


Figure B.6: (a): 2L-ILC evolution of the reference signal parameters, (b): improving engagement quality during convergence period at nominal conditions.

B.4.1 Model-based controllers

First, let's compare the two-level NMPC results of Figs. 5.13, 5.14 with the ILC approaches, for which the evolution of the reference parameters and the resulting engagements are shown in Fig. B.6. For both, the initial performance is poor, with a high torque peak due to an initial overfilling, resulting in an uncomfortable engagement for the operator. As a result, during the first parameter update, which is after 1 engagement for the NMPC approach and after 5 engagements for the ILC approach (to allow the low-level tracking time to converge), the high-level controller reacts by reducing t_{switch} as shown in Fig. B.6. Over the course of the following iterations, this and the other parameters are further adapted, and eventually smooth engagements are obtained, after 10 and 30 iterations respectively.

The results for the IO technique are shown in Fig. B.7, where similarly to the two-level NMPC and ILC approaches, we can see that the performance improves

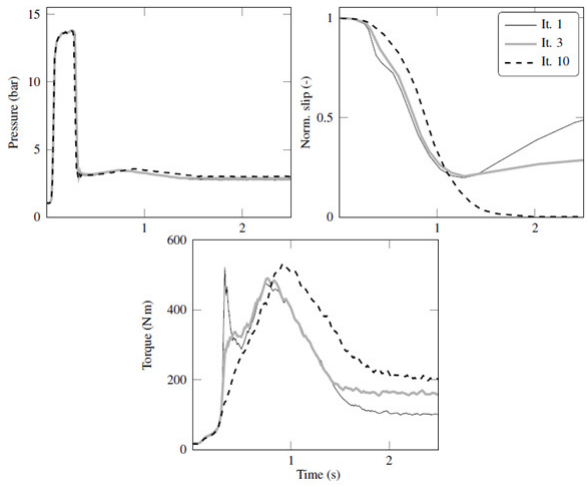


Figure B.7: Iterative optimization: Improving engagement quality during convergence period at nominal conditions.

as more iterations pass by, and eventually smooth engagements are found, synchronizing the clutch in a similar timeframe. This improvement is partially due to the learning of a few constraint parameters, but also due to the improving prediction accuracy of the models used in the low-level optimization, as shown in Fig. B.8. The convergence period is 10 trials, significantly shorter than that for the two-level ILC scheme and similar to that of the two-level MPC scheme. This reduction with respect to the two-level ILC scheme results form removing the indirect approach and instead directly optimizing based on the real specifications.

In a second test, the robustness is investigated by changing the operating conditions as in 5.7.4. Fig. B.9 depicts the obtained results of the 2L-ILC controllers, after convergence, which again takes around 30 iterations (compared to 10 iterations for 2L-NMPC shown in Fig. 5.15). For the increased temperature, the main difference is that the filling is completed sooner, which results from the decreased oil viscosity, and which has been compensated for mainly by reducing the value of p_{wid} in the pressure reference. More differences can be observed when the observed load is increased, as then higher torques and pressures are required, but despite this the slip signals remain very similar to the nominal case.

For the IO technique, the same tests have been performed, and the results after convergence are shown in Fig. B.10. As before, a good performance is still achieved by having the high-level laws adapt to the observed changes. The fact that the performance is similar to those of the two-level NMPC and ILC approaches illustrates that the parameterization used for those two is well-chosen, as they perform similar adaptations using only a few parameters. The reconvergence period is

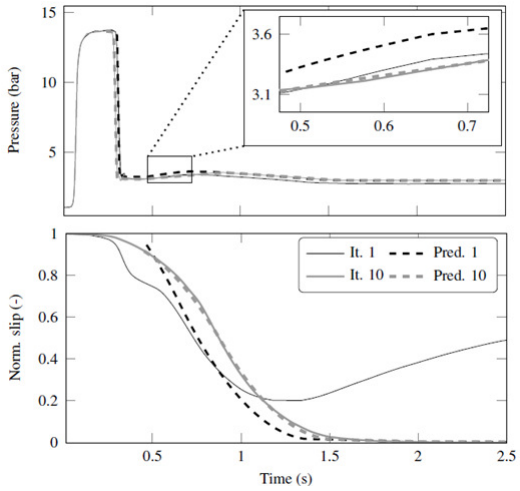


Figure B.8: Iterative optimization: Improving prediction accuracy during convergence period.

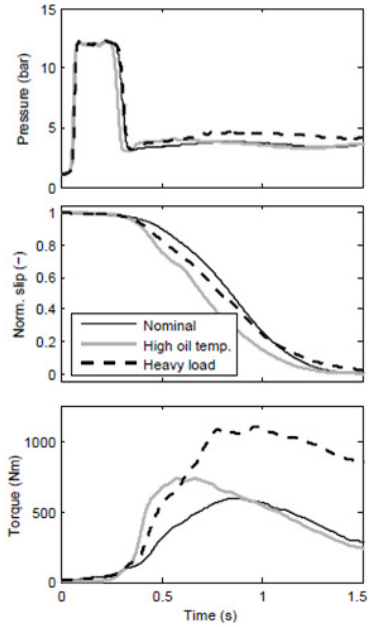


Figure B.9: Demonstration of 2L-ILC robustness to various operating conditions.

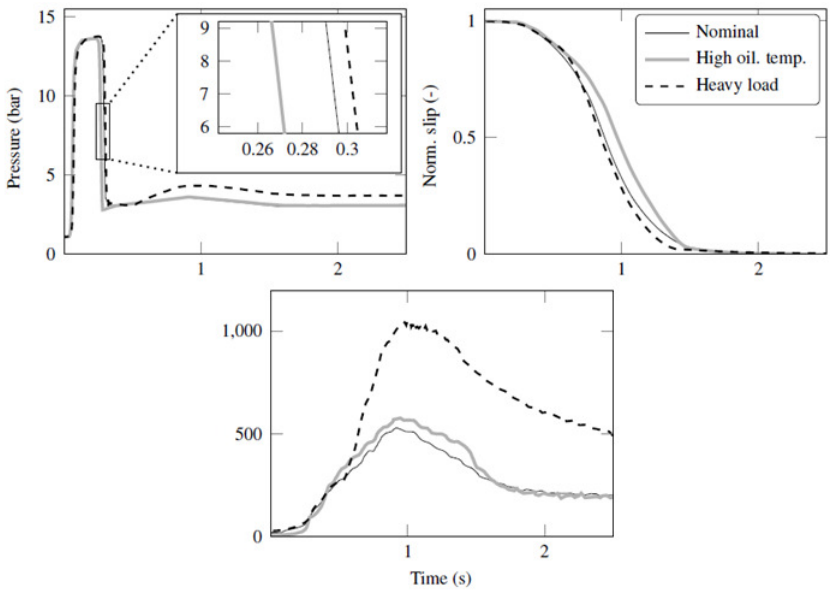


Figure B.10: Iterative optimization: Demonstration of robustness to various operating conditions.

again around 10 trials, similar to that of the two-level NMPC approach, and shorter than for the two-level ILC approach.

Apart from the shorter reconvergence period, the two-level and IO techniques share the additional advantage that if learning were to be restarted for a different operating point, it is easier to hotstart with a good guess of either the reference parameters or the constraint parameters and models, since it is easy to store and interpolate these values. For the two-level ILC approach the reference parameters could also be stored, but to fully allow a hotstarting and reuse all knowledge learned at a previous set of operating conditions, it would also be needed to transform the learned control signal to the current operating conditions, which is not a straightforward task.

B.4.2 Model-free controllers

Before we look into the results, it should be noted that each set of control signal parameters were first tested under conditions with a reduced load, to ensure it could be safely applied to the clutch under normal operating conditions. These additional tests are not included in any of the results, nor are they counted in the number of trials before convergence, but they do slow down the overall learning process. Ideally, other methods to ensure safety would need to be derived, not

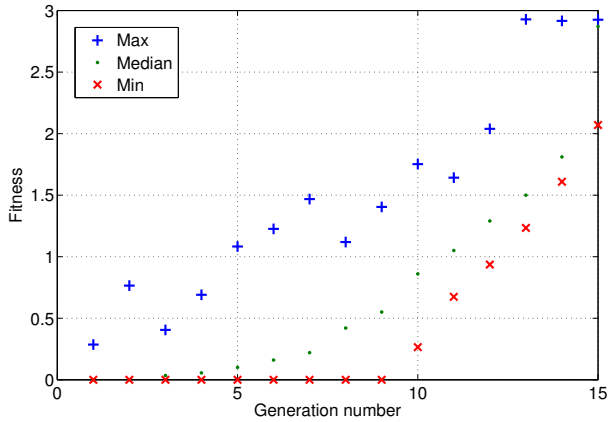


Figure B.11: GA: Minimum, median, and maximum fitness values during the GA evolution process.

requiring (as many) additional experiments.

To illustrate the convergence process of the model-free methods, the nominal conditions defined earlier for model-based controllers are reused. Under these conditions, the optimization processes for both are shown in Fig. B.11 and Fig. B.12 respectively. The GA maximized the fitness within 13 generations; each generation containing 50 individuals, while for reinforcement learning the reward is maximized after 85 test runs. The results obtained with each are presented in Fig. B.13, where it can be seen that engagements similar to those of the model-based techniques are obtained.

The robustness with respect to an increase in the oil temperature has also been checked for these methods. In this case, we reuse knowledge from the previous experiment under nominal conditions to narrow down the range of the parameter's value to reduce the amount of learning needed, and the results after convergence are also included in Fig. B.13. Similar observations can be made as before with the model-based controllers, with a good performance still being achieved by both controllers.

B.5 Comparison of model-based and model-free learning control

This section compares the 2L-NMPC results of 5.7.4 with all the other methods on the clutch, and presents a general discussion of their benefits and drawbacks.

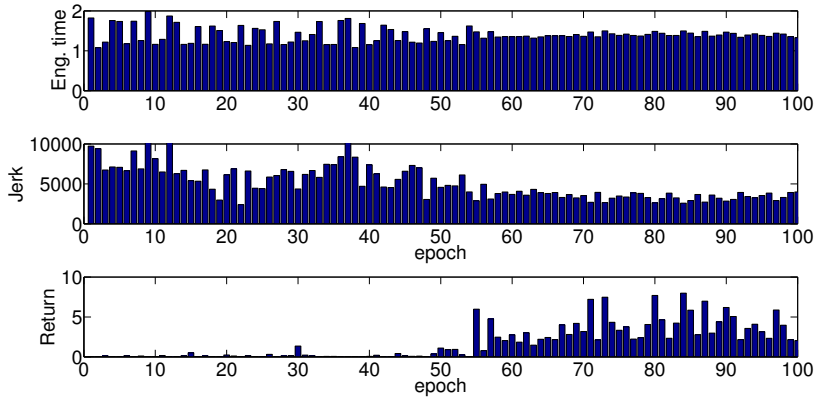


Figure B.12: PGPE: Evolution of engagement time (above), jerk (center), and reward (below) during learning process.

Index	2l-NMPC	2l-ILC	IO	GA	RL
Abs(Max(Jerk))	3.6683	2.8945	3.4133	3.9256	3.618
Eng.Time(s)	1.199	1.39	1.277	1.434	1.317

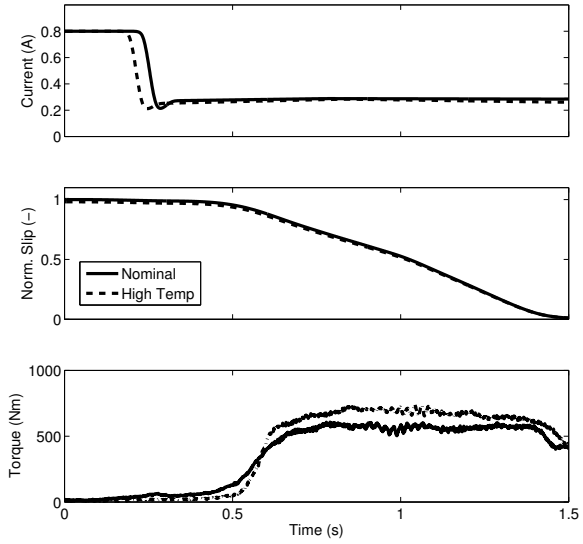
Table B.1: An empirical comparison between the model-based and non-model based control techniques based on jerk and engagement times

B.5.1 Comparison of engagement results

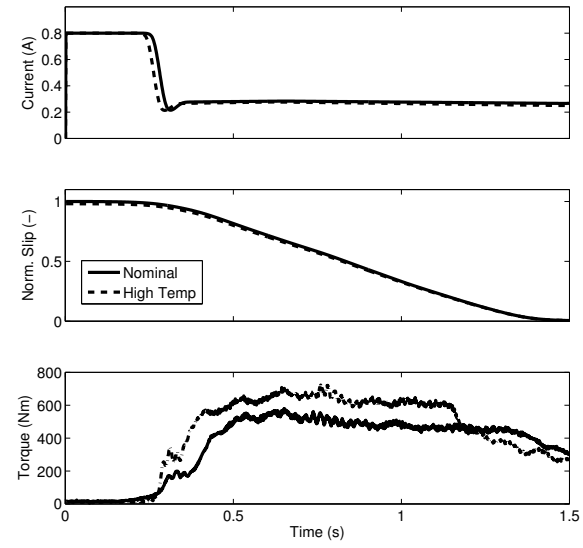
A comparison of all the presented methods for clutch control in terms of engagement time and jerk is presented in Table B.1. Among the model-based methods, the results are fairly similar, and none of them are clearly worse than any of the others in both categories. Among the model-free methods, RL performs slightly better than GA though in both categories, which probably indicates that the GA has not fully converged yet, as it should normally find a similar result as RL. Comparing the model-based and model-free techniques, we see that the model-based ones find engagements that are both faster and yield lower jerk values. This can be explained by the parameterization that is used for the model-free methods, which restricts the possible adaptations made by the controller, and which can lead to a reduced performance.

B.5.2 Discussion on model-based techniques

Comparing the model-based techniques, we immediately see that the two-level NMPC and IO technique require a similar convergence period, which is shorter than for the two-level ILC technique. This is a result of the fact that the two-level



a



b

Figure B.13: GA (a) and RL (b): Illustration of engagements achieved under nominal conditions and with an increased temperature.

ILC technique learns at both low and high levels, requiring additional iterations in which the low level converges, before good high-level updates can be made. As a result, the reconvergence period when the operating conditions change is also better for the two-level NMPC and the IO technique. Apart from the shorter reconvergence period, they have the additional advantage that if learning were to be restarted for a different operating point, it is easier to hotstart with a good guess of either the reference parameters or the constraint parameters and models, since it is easy to store and interpolate these values. This could also be done for the reference parameters of the two-level ILC approach, but to fully allow a hotstarting and reuse all knowledge learned at a previous set of operating conditions, it would also be needed to transform the learned control signal to the current operating conditions, which is not a straightforward task.

When comparing the required modeling effort, the two-level ILC technique outperforms the other methods however, as a highly accurate tracking control can be achieved despite having a large model uncertainty. In contrast, for NMPC it is needed to have an accurate non-linear model, which requires a time-consuming identification. For the IO technique this is not needed again, but here the performance is limited by the chosen model structure, of which the parameters are afterwards learnt online.

Even though the ILC approach improves the tracking behaviour, so that the references can be tracked more accurately than with the NMPC approach, it turns out that this may not be beneficial for the current application, where aggressive control can lead to unwanted vibrations and high jerk values.

B.5.3 Discussion on model-free techniques

Comparing the model-free techniques, we see that with the same reward/fitness function, both methods manage to yield similar engagements, but RL converges faster than GA.

The type of reward/fitness that was used is the same for both, and is a trade-off between the jerk and the engagement duration. These can be combined into a single scalar reward, which is the way in which RL typically operates. For GA it is however also possible not to use a fixed trade-off, but to really treat it as multiple objectives without extra cost, and find a complete pareto-front.

B.5.4 Comparison of model-based and model-free techniques

Model-based methods have a few advantages over model-free methods. First of all, they allow more freedom in the determination of the shape of the control signals. They will therefore generally be able to find the optimal solution, whereas for the model-free methods this is only possible if the chosen parameterization allows this. The convergence period is also significantly shorter for model-based methods,

Method/Property	2l-NMPC	2l-ILC	IO	GA	RL
Modeling requirement	⌒	⌣	⌣	⌣⌣	⌣⌣
Learning rate	⌣⌣	⌣	⌣⌣	⌒⌒	⌒
Stability	⌣	⌣	−−	−−	−−
Learning transient/Safety	⌣	⌣	⌣	⌒⌒	⌒⌒
Multi-objective	⌣	⌣	⌣	⌣⌣	⌣

Table B.2: Characteristic features of the model-based and model-free techniques

and hotstarting is often possible, which makes them more applicable for online adaptive purposes. Another advantage is that they posses an inherent robustness to parameter uncertainties due to the ability to use feedback. They finally also make it easier to predict the behaviour and thus ensure safety, even during the convergence period.

Despite these advantages of model-based methods, model-free methods also have some attractive properties for the control of mechatronic systems. Their main benefit is that they can operate without model, and thus require no identification or apriori system knowledge. This makes them ideal for usage as an add-on to complex existing systems, or to automate offline calibration procedures where it is not possible to rely on heuristics or insight to manually design tuning rules. It should however be stated that parameterizations are typically needed to limit the convergence period, and it is practically impossible to select the shape of the signal beforehand without system knowledge or some simple tests. While these parameterizations generally do lead to a reduced performance, a well-chosen parameterization can limit this reduction, and this choice is thus important. Since more parameters lead to a better performance but longer convergence, the difficult part is to select parameterizations with only a low number of parameters, yet which still allow a performance close to the true optimum to be achieved.

These results have been summarized in Table B.2, which gives a qualitative comparison between the different techniques. The key ⌣⌣ means the best in the category, followed by ⌣ for good and then ⌒ for bad to ⌒⌒ for worst in the category. The −− key signifies that the corresponding property has not been established.

B.6 Summary

The 2L-NMPC and other model-based methods do converge in shorter time periods, and make it easier to guarantee safety during the convergence period, which makes then more suitable to online applications. The model-free methods on the other hand can be applied to complex systems whenever models are hard to come by, and are especially useful as an automated tuning method when insight in the

dynamics does not allow an experienced user to define proper tuning rules. These model-free methods can further also be used to learn complete pareto-fronts of optimal controllers, allowing a selection of which controller to be used to be made later on.

The combination and extension of all the stated methodologies for distributed control is a work in progress.

Bibliography

[1] F. J. Christophersen, M. N. Zeilinger, C. N. Jones, and M. Morari, "Controller complexity reduction for piecewise affine systems through safe region elimination," in *Decision and Control, 2007 46th IEEE Conference on*, pp. 4773–4778, IEEE, 2007.

[2] H. E. Merritt, *Hydraulic control systems*. Wiley New York, 1967.

[3] J. Richalet, A. Rault, J. Testud, and J. Papon, "Model predictive heuristic control: Applications to industrial processes," *Automatica*, vol. 14, no. 5, pp. 413–428, 1978.

[4] C. R. Cutler and B. L. Ramaker, "Dynamic matrix control-a computer control algorithm," in *Proceedings of the joint automatic control conference*, vol. 1, pp. Wp5–B, American Automatic Control Council Piscataway, NJ, 1980.

[5] R. De Keyser and A. Van Cauwenberghe, "Extended prediction self-adaptive control," in *IFAC Symp. on Identification and System Parameter Estimation*, pp. 1255–1260, 1985.

[6] D. W. Clarke, C. Mohtadi, and P. Tuffs, "Generalized predictive controlpart i. the basic algorithm," *Automatica*, vol. 23, no. 2, pp. 137–148, 1987.

[7] J. M. Maciejowski, *Predictive control: with constraints*. Pearson education, 2002.

[8] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control engineering practice*, vol. 11, no. 7, pp. 733–764, 2003.

[9] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[10] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *Control Systems Technology, IEEE Transactions on*, vol. 18, no. 2, pp. 267–278, 2010.

[11] L. Del Re, F. Allgöwer, L. Glielmo, C. Guardiola, and I. Kolmanovsky, *Automotive model predictive control: models, methods and applications*, vol. 402. Springer, 2010.

[12] A. Dutta, R. De Keyser, C.-M. Ionescu, J. Stoev, G. Pinte, and W. Symens, "Robust predictive control design for optimal wet-clutch engagement," in *American Control Conference (ACC)*, pp. 4576–4581, IEEE, 2012.

[13] A. Dutta, C.-M. Ionescu, R. De Keyser, B. Wyns, J. Stoev, G. Pinte, and W. Symens, "Robust and two-level (nonlinear) predictive control of switched dynamical systems with unknown references for optimal wet-clutch engagement," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 228(4), pp. 233–244, 2014.

[14] A. Dutta, R. De Keyser, Y. Zhong, B. Wyns, G. Pinte, and J. Stoev, "Robust predictive control of a wet-clutch using evolutionary algorithm optimized engagement profile," in *System Theory, Control, and Computing (ICSTCC), 2011 15th International Conference on*, pp. 1–6, IEEE, 2011.

[15] A. Dutta, E. Hartley, J. Maciejowski, and R. De Keyser, "Certification of a class of industrial predictive controllers without terminal conditions," in *Decision and Control, 53rd IEEE Conference on*, IEEE, 2014.

[16] A. Dutta, M. Loccufier, C. M. Ionescu, and R. De Keyser, "Penalty adaptive model predictive control (PAMPC) of constrained, underdamped, non-collocated mechatronic systems," in *Control Applications (CCA), 2013 IEEE International Conference on*, pp. 1006–1011, IEEE, 2013.

[17] A. Dutta, M. Loccufier, C. M. Ionescu, and R. De Keyser, "Robust penalty adaptive model predictive control (PAMPC) of constrained, underdamped, non-collocated systems," *Journal of Vibration and Control*, 2014.

[18] A. Dutta, B. Depraetere, C. Ionescu, G. Pinte, J. Swevers, and R. De Keyser, "Comparison of two-level nmpc and ilc strategies for wet-clutch control," *Control Engineering Practice*, vol. 22, pp. 114–124, 2014.

[19] A. Dutta, C. Ionescu, B. Wyns, R. De Keyser, J. Stoev, G. Pinte, and W. Symens, "Switched nonlinear predictive control with adaptive references for engagement of wet clutches," in *Nonlinear Model Predictive Control, 4th IFAC conference on*, vol. 4, pp. 460–465, 2012.

[20] A. Dutta, C.-M. Ionescu, B. Wyns, R. De Keyser, J. Stoev, G. Pinte, and W. Symens, "Switched predictive control design for optimal wet-clutch engagement," in *IFAC Workshop on Engine and Powertrain Control, Simulation and Modeling (ECOSM-2012)*, pp. 319–324, 2012.

[21] A. Dutta, C. M. Ionescu, and R. De Keyser, "A pragmatic approach to distributed nonlinear model predictive control: Application to a hydrostatic drivetrain," *Optimal Control Applications and Methods*, 2014.

[22] A. Dutta, R. De Keyser, and I. Nopens, "Robust nonlinear extended prediction self-adaptive control (NEPSAC) of continuous bioreactors," in *Control & Automation (MED), 2012 20th Mediterranean Conference on*, pp. 658–664, 2012.

[23] A. Dutta, Y. Zhong, B. Depraetere, K. Van Vaerenbergh, C. Ionescu, B. Wyns, G. Pinte, A. Nowe, J. Swevers, and R. De Keyser, "Model-based and model-free learning strategies for wet clutch control," *Mechatronics*, 2014.

[24] R. E. Kalman, "Contributions to the theory of optimal control," *Bol. Soc. Mat. Mexicana*, vol. 5, no. 2, pp. 102–119, 1960.

[25] A. Propoi, "Use of linear programming methods for synthesizing sampled-data automatic systems," *Automation and Remote Control*, vol. 24, no. 7, pp. 837–844, 1963.

[26] D. Clarke, C. Mohtadi, and P. Tuffs, "Generalized predictive controlpart ii extensions and interpretations," *Automatica*, vol. 23, no. 2, pp. 149–160, 1987.

[27] D. Clarke and R. Scattolini, "Constrained receding-horizon predictive control," in *IEE Proceedings D (Control Theory and Applications)*, vol. 138, pp. 347–354, IET, 1991.

[28] E. Mosca and J. Zhang, "Stable redesign of predictive control," *Automatica*, vol. 28, no. 6, pp. 1229–1233, 1992.

[29] W. H. Kwon, A. Bruckstein, and T. Kailath, "Stabilizing state-feedback design via the moving horizon method," *International Journal of Control*, vol. 37, no. 3, pp. 631–643, 1983.

[30] J. H. Lee, M. Morari, and C. E. Garcia, "State-space interpretation of model predictive control," *Automatica*, vol. 30, no. 4, pp. 707–717, 1994.

[31] J. B. Rawlings and D. Q. Mayne, *Model predictive control: Theory and design*. Nob Hill Pub., 2009.

[32] P. O. Scokaert, J. B. Rawlings, and E. S. Meadows, "Discrete-time stability with perturbations: Application to model predictive control," *Automatica*, vol. 33, no. 3, pp. 463–470, 1997.

[33] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[34] R. De Keyser, "Model based predictive control for linear systems," *UNESCO Encyclopaedia of Life Support Systems http://www.eolss.net (available online at: http://www.eolss.net/sample-chapters/c18/e6-43-16-01.pdf). Article contribution 6.43.16.1, Eolss Publishers Co Ltd, Oxford*, p. 35 pages, 2003.

[35] L. Wang, *Model predictive control system design and implementation using MATLAB®*. Springer, 2009.

[36] R. Soeterboek, *Predictive Control: A Unified Approach*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1992.

[37] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer, 2013.

[38] A. W. Ordys and D. W. Clarke, "A state-space description for GPC controllers," *International Journal of Systems Science*, vol. 24, no. 9, pp. 1727–1744, 1993.

[39] A. Bemporad, L. Chisci, and E. Mosca, "On the stabilizing property of siorhc," *Automatica*, vol. 30, no. 12, pp. 2013–2015, 1994.

[40] J. Maciejowski and S. Redhead, "Exact state-space correspondence to GPC: observer eigenvalue locations," in *Asian Control Conference*, 2002.

[41] R. Fletcher, *Practical methods of optimization*. John Wiley & Sons, 2013.

[42] D. Mayne, "An apologia for stabilising terminal conditions in model predictive control," *International Journal of Control*, vol. 86, no. 11, pp. 2090–2095, 2013.

[43] J. Löfberg, "Oops! i cannot do it again: Testing for recursive feasibility in MPC," *Automatica*, vol. 48, no. 3, pp. 550–555, 2012.

[44] J. H. Lee, "Model predictive control: review of the three decades of development," *International Journal of Control, Automation and Systems*, vol. 9, no. 3, pp. 415–424, 2011.

[45] P. O. Scokaert and J. B. Rawlings, "Infinite horizon linear quadratic control with constraints," in *Proceedings of the 13th IFAC World Congress, San Francisco*, vol. M, pp. 109–114, 1996.

[46] D. Limón, T. Alamo, F. Salas, and E. F. Camacho, "On the stability of constrained mpc without terminal constraint," *Automatic Control, IEEE Transactions on*, vol. 51, no. 5, pp. 832–836, 2006.

[47] L. Grüne and J. Pannek, *Nonlinear Model Predictive Control*. Springer, 2011.

[48] E. C. Kerrigan and J. M. Maciejowski, "Invariant sets for constrained nonlinear discrete-time systems with application to feasibility in model predictive control," in *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, vol. 5, pp. 4951–4956, IEEE, 2000.

[49] E. C. Kerrigan and J. M. Maciejowski, "Robust feasibility in model predictive control: Necessary and sufficient conditions," in *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, vol. 1, pp. 728–733, IEEE, 2001.

[50] F. Blanchini, "Set invariance in control," *Automatica*, vol. 35, no. 11, pp. 1747–1767, 1999.

[51] E. G. Gilbert and K. T. Tan, "Linear systems with state and control constraints: The theory and application of maximal output admissible sets," *Automatic Control, IEEE Transactions on*, vol. 36, no. 9, pp. 1008–1020, 1991.

[52] C. Dórea and J. Hennet, "(a, b)-invariant polyhedral sets of linear discrete-time systems," *Journal of Optimization Theory and Applications*, vol. 103, no. 3, pp. 521–542, 1999.

[53] D. Bertsekas, "Infinite time reachability of state-space regions by using feedback control," *Automatic Control, IEEE Transactions on*, vol. 17, no. 5, pp. 604–613, 1972.

[54] D. Q. Mayne, E. C. Kerrigan, and P. Falugi, "Robust model predictive control: advantages and disadvantages of tube-based methods," in *Proc. of the 18th IFAC World Congress Milano*, pp. 191–196, 2011.

[55] P. Grieder, M. Lüthi, P. Parrilo, and M. Morari, "Stability & feasibility of constrained receding horizon control," in *Proc. of the European Control Conference 2003*, 2003.

[56] V. G. Rao and D. S. Bernstein, "Naive control of the double integrator," *Control Systems, IEEE*, vol. 21, no. 5, pp. 86–97, 2001.

[57] MATLAB, *version 8.1.0.604 (R2013a)*. Natick, Massachusetts: The MathWorks Inc., 2013.

[58] M. Baric, M. Baotic, and M. Morari, "On-line tuning of controllers for systems with constraints," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pp. 8288–8293, IEEE, 2005.

[59] Z. Wan and M. V. Kothare, "An efficient off-line formulation of robust model predictive control using linear matrix inequalities," *Automatica*, vol. 39, no. 5, pp. 837–846, 2003.

[60] A. Bemporad, "Reference governor for constrained nonlinear systems," *Automatic Control, IEEE Transactions on*, vol. 43, no. 3, pp. 415–419, 1998.

[61] Z.-c. Qiu, J.-d. Han, X.-m. Zhang, Y.-c. Wang, and Z.-w. Wu, "Active vibration control of a flexible beam using a non-collocated acceleration sensor and piezoelectric patch actuator," *Journal of sound and vibration*, vol. 326, no. 3, pp. 438–455, 2009.

[62] Y. Xie, A. G. Alleyne, A. Greer, and D. Deneault, "Fundamental limits in combine harvester header height control," *Journal of dynamic systems, measurement, and control*, vol. 135, no. 3, p. 034503, 2013.

[63] A. Preumont, *Vibration control of active structures*. Kluwer Amsterdam, 1997.

[64] J. Richelot, J. Bordeneuve-Guibé, and V. Pommier-Budinger, "Active control of a clamped beam equipped with piezoelectric actuator and sensor using generalized predictive control," in *Industrial Electronics, 2004 IEEE International Symposium on*, vol. 1, pp. 583–588, IEEE, 2004.

[65] R. Brown, J. Pusey, M. Murugan, and D. Le, "Generalized predictive control algorithm of a simplified ground vehicle suspension system," *Journal of Vibration and Control*, vol. 19, no. 16, pp. 2372–2386, 2013.

[66] A. G. Wills, D. Bates, A. J. Fleming, B. Ninness, and S. R. Moheimani, "Model predictive control applied to constraint handling in active noise and vibration control," *Control Systems Technology, IEEE Transactions on*, vol. 16, no. 1, pp. 3–12, 2008.

[67] J. L. Garriga and M. Soroush, "Model predictive control tuning methods: A review," *Industrial & Engineering Chemistry Research*, vol. 49, no. 8, pp. 3505–3515, 2010.

[68] M. L. Darby and M. Nikolaou, "MPC: Current practice and challenges," *Control Engineering Practice*, vol. 20, no. 4, pp. 328–342, 2012.

[69] R. Suzuki, F. Kawai, H. Ito, C. Nakazawa, Y. Fukuyama, and E. Aiyoshi, "Automatic tuning of model predictive control using particle swarm optimization," in *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*, pp. 221–226, IEEE, 2007.

[70] W. Ebert, *Polynomial Predictive Control Methods: Lecture Notes in Control and Information Sciences*. VDI-Verlag, 2008.

[71] R. A. Soeterboek, A. F. Pels, H. B. Verbruggen, and G. C. van Langen, "A predictive controller for the mach number in a transonic wind tunnel," *Control Systems, IEEE*, vol. 11, no. 1, pp. 63–72, 1991.

[72] I. Sokolov and V. Babitsky, "Phase control of self-sustained vibration," *Journal of sound and vibration*, vol. 248, no. 4, pp. 725–744, 2001.

[73] K. J. Astrom and B. Wittenmark, *Computer-Controlled systems: theory and design*. Prentice-Hall: Englewood Cliffs, NJ, 1990.

[74] R. De Keyser, A. Dutta, A. Hernandez, and C. M. Ionescu, "A specifications based PID autotuner," in *Control Applications (CCA), 2012 IEEE International Conference on*, pp. 1621–1626, IEEE, 2012.

[75] S. Di Cairano, A. Bemporad, I. Kolmanovsky, and D. Hrovat, "Model predictive control of nonlinear mechatronic systems: An application to a magnetically atuated," in *Analysis and Design of Hybrid Systems*, vol. 2, pp. 241–246, 2006.

[76] T. M. Obrzut, *Non-Collocation Problems in Dynamics and Control of Mechanical Systems*. 2009.

[77] C. F. Beards, *Vibrations and control systems*. E. Horwood, 1988.

[78] P. O. Scokaert, D. Q. Mayne, and J. B. Rawlings, "Suboptimal model predictive control (feasibility implies stability)," *Automatic Control, IEEE Transactions on*, vol. 44, no. 3, pp. 648–654, 1999.

[79] 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*, vol. 32, no. 3, pp. 298–312, 2011.

[80] H. Chen and F. Allgöwer, "A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability," *Automatica*, vol. 34, no. 10, pp. 1205–1217, 1998.

[81] M. Fiacchini, T. Alamo, and E. F. Camacho, "On the computation of convex robust control invariant sets for nonlinear systems," *Automatica*, vol. 46, no. 8, pp. 1334–1338, 2010.

[82] G. Conte, C. H. Moog, and A. M. Perdon, *Algebraic methods for nonlinear control systems*. Springer, 2007.

[83] G. De Nicolao, L. Magni, and R. Scattolini, "Stabilizing predictive control of nonlinear ARX models," *Automatica*, vol. 33, no. 9, pp. 1691–1697, 1997.

[84] A. Bemporad, A. Garulli, S. Paoletti, and A. Vicino, "A bounded-error approach to piecewise affine system identification," *Automatic Control, IEEE Transactions on*, vol. 50, no. 10, pp. 1567–1580, 2005.

[85] S. Wright and J. Nocedal, *Numerical optimization*, vol. 2. Springer New York, 1999.

[86] F. Borrelli, M. Baotic, A. Bemporad, and M. Morari, "An efficient algorithm for computing the state feedback optimal control law for discrete time hybrid systems," in *Proceedings of the American control conference*, vol. 6, pp. 4717–4722, 2006.

[87] X. Xu and P. J. Antsaklis, "Optimal control of switched systems: new results and open problems," in *American Control Conference, 2000. Proceedings of the 2000*, vol. 4, pp. 2683–2687, IEEE, 2000.

[88] Z. Sun and K. Hebbale, "Challenges and opportunities in automotive transmission control," in *American Control Conference, Proceedings of the*, pp. 3284–3289, IEEE, 2005.

[89] L. Glielmo and F. Vasca, "Engagement control for automotive dry clutch," in *American Control Conference, 2000. Proceedings of the 2000*, vol. 2, pp. 1016–1017, IEEE, 2000.

[90] A. V. D. Heijden, A. Serrarens, M. Camlibel, and H. Nijmeijer, "Hybrid optimal control of dry clutch engagement," *International Journal of Control*, vol. 80, no. 11, pp. 1717–1728, 2007.

[91] P. J. Dolcini, *Contribution to the clutch comfort*. Institut National Polytechnique de Grenoble, 2007.

[92] X. Song, M. A. M. Zulkefli, and Z. Sun, "Automotive transmission clutch fill optimal control: An experimental investigation," in *American Control Conference (ACC), 2010*, pp. 2748–2753, IEEE, 2010.

[93] R. Morselli, E. Sereni, E. Bedogni, and E. Sedoni, "Modeling and control of wet clutches by pressure-control valves," in *Modeling and Control of Economic Systems 2001:(SME 2001): a Proceedings Volume from the 10th IFAC Symposium, Klagenfurt, Austria, 6-8 September 2001*, p. 79, Elsevier, 2003.

[94] J. Y. Wong, *Theory of ground vehicles*. John Wiley & Sons, 2001.

[95] M. Grotjahn, L. Quernheim, and S. Zemke, "Modelling and identification of car driveline dynamics for anti-jerk controller design," in *Mechatronics, 2006 IEEE International Conference on*, pp. 131–136, IEEE, 2006.

[96] W. D. Widanage, J. Stoev, A. Van Mulders, J. Schoukens, and G. Pinte, "Nonlinear system-identification of the filling phase of a wet-clutch system," *Control Engineering Practice*, vol. 19, no. 12, pp. 1506–1516, 2011.

[97] R. Scattolini, "Architectures for distributed and hierarchical model predictive control–a review," *Journal of Process Control*, vol. 19, no. 5, pp. 723–731, 2009.

[98] P. D. Christofides, J. Liu, and D. M. de la Peana, *Networked and distributed predictive control: Methods and nonlinear process network applications*. Springer, 2011.

[99] B. Picasso, C. Romani, and R. Scattolini, "Tracking control of wiener models with hierarchical and switching model predictive control," *Optimal Control Applications and Methods*, vol. 34, no. 1, pp. 1–16, 2013.

[100] J. Liu, D. Munoz de la Pena, and P. D. Christofides, "Distributed model predictive control of nonlinear process systems," *AIChE Journal*, vol. 55, no. 5, pp. 1171–1184, 2009.

[101] A. Richards and J. How, "Robust distributed model predictive control," *International Journal of Control*, vol. 80, no. 9, pp. 1517–1531, 2007.

[102] E. Camponogara, D. Jia, and B. K. S. Talukdar, "Distributed model predictive control," *Control Systems, IEEE*, vol. 22, no. 1, pp. 44–52, 2002.

[103] P. D. Christofides, R. Scattolini, D. M. de la Pena, and J. Liu, "Distributed model predictive control: a tutorial review," *Proc. Chem. Process Control*, vol. 8, p. 22pp, 2012.

[104] J. Rawlings and B. Stewart, "Coordinating multiple optimization-based controllers: New opportunities and challenges," *Journal of Process Control*, vol. 18, no. 9, pp. 839–845, 2008.

[105] K. Rydberg, "Hydrostatic drives in heavy mobile machinery-new concepts and development trends," *SAE transactions*, vol. 107, pp. 232–238, 1998.

[106] L. Bakule, "Decentralized control: An overview," *Annual Reviews in Control*, vol. 32, no. 1, pp. 87–98, 2008.

[107] J. Maestre, D. Munoz de la Pena, and E. Camacho, "Distributed model predictive control based on a cooperative game," *Optimal Control Applications and Methods*, vol. 32, no. 2, pp. 153–176, 2011.

[108] A. Venkat, J. Rawlings, and S. Wright, "Stability and optimality of distributed model predictive control," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pp. 6680–6685, IEEE, 2005.

[109] R. De Keyser and A. Van Cauwenberghe, "A self-tuning multistep predictor application," *Automatica*, vol. 17, no. 1, pp. 167–174, 1981.

[110] L. Ljung, *System identification*. Springer, 1998.

[111] S. Bittanti, P. Bolzern, and M. Campi, "Convergence and exponential convergence of identification algorithms with directional forgetting factor," *Automatica*, vol. 26, no. 5, pp. 929–932, 1990.

[112] D. A. Bristow, M. Tharayil, and A. G. Alleyne, "A survey of iterative learning control," *Control Systems, IEEE*, vol. 26, no. 3, pp. 96–114, 2006.

[113] R. W. Longman, "Iterative learning control and repetitive control for engineering practice," *International Journal of Control*, vol. 73, no. 10, pp. 930–954, 2000.

[114] B. Depraetere, G. Pinte, W. Symens, and J. Swevers, "A two-level iterative learning control scheme for the engagement of wet clutches," *Mechatronics*, vol. 21, no. 3, pp. 501–508, 2011.

[115] B. Depraetere, G. Pinte, and J. Swevers, "A reference free iterative learning strategy for wet clutch control," in *American Control Conference (ACC), 2011*, pp. 2442–2447, IEEE, 2011.

[116] K. Deb, *Optimization for engineering design: Algorithms and examples*. PHI Learning Pvt. Ltd., 2012.

[117] R. E. Steuer, *Multiple criteria optimization: theory, computation, and application*. Krieger Malabar, 1989.

[118] T. Weise and K. Geihs, "Genetic programming techniques for sensor networks," in *Proceedings of*, vol. 5, pp. 21–25, 2006.

[119] R. J. Urbanowicz and J. H. Moore, "Learning classifier systems: a complete introduction, review, and roadmap," *Journal of Artificial Evolution and Applications*, vol. 2009, p. 1, 2009.

[120] Y. Zhong, B. Wyns, R. De Keyser, G. Pinte, and J. Stoev, "An implementation of genetic-based learning classifier system on a wet clutch system," in *14th Applied Stochastic Models and Data Analysis Conference (ASMDA 2011)*, pp. 1431–1439, 2011.

[121] A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 1998.

[122] J. Peters and S. Schaal, "Policy gradient methods for robotics," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pp. 2219–2225, IEEE, 2006.

[123] F. Sehnke, C. Osendorfer, T. Rückstieß, A. Graves, J. Peters, and J. Schmidhuber, "Parameter-exploring policy gradients," *Neural Networks*, vol. 23, no. 4, pp. 551–559, 2010.

[124] M. Gagliolo, K. Van Vaerenbergh, A. Rodriguez, A. Nowé, S. Goossens, G. Pinte, and W. Symens, "Policy search reinforcement learning for automatic wet clutch engagement," in *System Theory, Control, and Computing (ICSTCC), 2011 15th International Conference on*, pp. 1–6, IEEE, 2011.

[125] F. Sehnke, C. Osendorfer, T. Rückstieß, A. Graves, J. Peters, and J. Schmidhuber, "Policy gradients with parameter-based exploration for control," in *Artificial Neural Networks-ICANN 2008*, pp. 387–396, Springer, 2008.