Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Algorithms and Applications for Nonlinear Model Predictive Control with Long Prediction Horizon

Ph.D. candidate:
Yutao Chen

Advisor:
Prof. Alessandro Beghi

Coordinator:
Prof. Andrea Neviani

Ph.D. School in
Information Engineering

Department of Information Engineering
University of Padova
2017

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Università degli Studi di Padova/ University of Padova

Department of Information Engineering

Ph.D. course in Information Engineering

Information and Communication Technologies

XXX

Algorithms and Applications for Nonlinear Model Predictive Control with Long Prediction Horizon

Coordinator: Prof. Andrea Neviani

Supervisor: Prof. Alessandro Beghi

Ph.D. student: Yutao Chen

Acknowledgements

I have not noticed that time has gone by so fast until I start writing this part of my thesis. I can still remember the hesitation and struggling to determine a Ph.D journey in Padova, Italy. First I would like to thank Prof. Giorgio Picci, who offered me an opportunity to pursue my Ph.D in University of Padova. He is a gentleman and is still active in scientific research. Also I would like to thank Mrs. Kim Anne Barchi, who helped me a lot personally to survive and live in Padova.

I would like to deeply thank my supervisor Prof. Alessandro Beghi, who offered me inspiring research possibilities during these three years. Alessandro is always willing to listen to and to discuss research ideas. He is also patient and earnest for review and correction of my scientific paper. I was very happy during these three years since I always feel encouraged by Alessandro, no matter whether I have met his satisfaction.

Next I would like to thank Dr. Mattia Bruschetta, who worked with me as team on a daily basis for the past three years. As a team we have very clear division of work. Mattia was focusing on providing models for me to play around and I was responsible for developing and implementing algorithms. I am really appreciated to have chance to work on some real-world applications.

I would like to thank Dr. Davide Cuccato who was a Ph.D student when I arrived at Italy. We have worked closely on some of the results presented in this thesis and participated in a summer school in Freiburg. We had a lot of happy memories together.

I would like to thank all of my Ph.D colleagues who have helped me a lot in my daily life. I felt like home everyday since the atmosphere in the research office was so nice.

Finally I want to express my gratitude to my parents, who support me unconditionally for the past three years. At this point, I am privileged to have spent three years in Padova and have met so many professors, researchers and friends. This is an important period in my life.

Yutao Chen

Summary

Fast implementations of NMPC are important when addressing real-time control of systems exhibiting features like fast dynamics, large dimension, and long prediction horizon, as in such situations the computational burden of the NMPC may limit the achievable control bandwidth.

For that purpose, this thesis addresses both algorithms and applications.

First, fast NMPC algorithms for controlling continuous-time dynamic systems using a long prediction horizon have been developed. A bridge between linear and nonlinear MPC is built using partial linearizations or sensitivity update. In order to update the sensitivities only when necessary, a Curvature-like measure of nonlinearity (CMoN) for dynamic systems has been introduced and applied to existing NMPC algorithms. Based on CMoN, intuitive and advanced updating logic have been developed for different numerical and control performance. Thus, the CMoN, together with the updating logic, formulates a partial sensitivity updating scheme for fast NMPC, named CMoN-RTI. Simulation examples are used to demonstrate the effectiveness and efficiency of CMoN-RTI. In addition, a rigorous analysis on the optimality and local convergence of CMoN-RTI is given and illustrated using numerical examples.

Partial condensing algorithms have been developed when using the proposed partial sensitivity update scheme. The computational complexity has been reduced since part of the condensing information are exploited from previous sampling instants. A sensitivity updating logic together with partial condensing is proposed with a complexity linear in prediction length, leading to a speed up by a factor of ten.

Partial matrix factorization algorithms are also proposed to exploit partial sensitivity update. By applying splitting methods to multi-stage problems, only part of the resulting KKT system need to be updated, which is computationally dominant in on-line optimization. Significant improvement has been proved by giving floating point operations (flops).

Second, efficient implementations of NMPC have been achieved by developing a Matlab based package named MATMPC. MATMPC has two working modes: the one completely relies on Matlab and the other employs the MATLAB C language API. The advantages of MATMPC are that algorithms are easy to develop and debug thanks to Matlab, and libraries

and toolboxes from Matlab can be directly used. When working in the second mode, the computational efficiency of MATMPC is comparable with those software using optimized code generation. Real-time implementations are achieved for a nine degree of freedom dynamic driving simulator and for multi-sensory motion cueing with active seat.

Sommario

Implementazioni rapide di NMPC sono importanti quando si affronta il controllo in tempo reale di sistemi che presentano caratteristiche come dinamica veloce, ampie dimensioni e orizzonte di predizione lungo, poichÃľ in tali situazioni il carico di calcolo dell'MNPC puÃš limitare la larghezza di banda di controllo ottenibile.

A tale scopo, questa tesi riguarda sia gli algoritmi che le applicazioni.

In primo luogo, sono stati sviluppati algoritmi veloci NMPC per il controllo di sistemi dinamici a tempo continuo che utilizzano un orizzonte di previsione lungo. Un ponte tra MPC lineare e non lineare viene costruito utilizzando linearizzazioni parziali o aggiornamento della sensibilitÃă. Al fine di aggiornare la sensibilitÃă solo quando necessario, Ãĺ stata introdotta una misura simile alla curva di non linearitÃă (CMoN) per i sistemi dinamici e applicata agli algoritmi NMPC esistenti. Basato su CMoN, sono state sviluppate logiche di aggiornamento intuitive e avanzate per diverse prestazioni numeriche e di controllo. Pertanto, il CMoN, insieme alla logica di aggiornamento, formula uno schema di aggiornamento della sensibilitÃă parziale per NMPC veloce, denominato CMoN-RTI. Gli esempi di simulazione sono utilizzati per dimostrare l'efficacia e l'efficienza di CMoN-RTI. Inoltre, un'analisi rigorosa sull'ottimalitÃă e sulla convergenza locale di CMoN-RTI viene fornita ed illustrata utilizzando esempi numerici.

Algoritmi di condensazione parziale sono stati sviluppati quando si utilizza lo schema di aggiornamento della sensibilitÃă parziale proposto. La complessitÃă computazionale Ãĺ stata ridotta poichÃľ parte delle informazioni di condensazione sono sfruttate da precedenti istanti di campionamento. Una logica di aggiornamento della sensibilitÃă insieme alla condensazione parziale viene proposta con una complessitÃă lineare nella lunghezza della previsione, che porta a una velocitÃă di un fattore dieci.

Sono anche proposti algoritmi di fattorizzazione parziale della matrice per sfruttare l'aggiornamento della sensibilitÃă parziale. Applicando metodi di suddivisione a problemi a piÃź stadi, Ãĺ necessario aggiornare solo parte del sistema KKT risultante, che Ãĺ computazionalmente dominante nell'ottimizzazione online. Un miglioramento significativo Ãĺ stato dimostrato dando operazioni in virgola mobile (flop).

In secondo luogo, sono state realizzate implementazioni efficienti di NMPC sviluppando un pacchetto basato su Matlab chiamato MATMPC. MATMPC ha due modalitÃă operative: quella si basa completamente su Matlab e l'altra utilizza l'API del linguaggio MATLAB C. I vantaggi di MATMPC sono che gli algoritmi sono facili da sviluppare e eseguire il debug grazie a Matlab e le librerie e le toolbox di Matlab possono essere utilizzate direttamente. Quando si lavora nella seconda modalitÃă, l'efficienza computazionale di MATMPC ÃÍ paragonabile a quella del software che utilizza la generazione di codice ottimizzata. Le realizzazioni in tempo reale sono ottenute per un simulatore di guida dinamica di nove gradi di libertÃă e per il movimento multisensoriale con sedile attivo.

Contents

1 Introduction

Model Predictive Control (MPC) has been a popular advanced control technique since its success in process industry in 1980s. During the past decades, MPC has been widely used in the application of chemical , aerospace, paper manufacturing and automotive industries (Garcia, Prett, and Morari, 1989; Qin and Badgwell, 2003; Camacho and Alba, 2013). Comparing to classical PI/PID controllers, MPC employs a mathematical model of the control system and exploits optimization techniques to find the optimal control law. The capability of tackling multi-input-multi-output (MIMO) systems with constraints enables MPC superior control performance in a lot of scenarios that require complex control strategies and careful controller design. In addition, MPC is able to respond to disturbances really fast, due to the fact that it makes use of repeated optimization w.r.t. the latest state measurement or estimation.

Nowadays, there are increasingly more applications of MPC in the field of mechanical, electrical and electronic engineering. Such applications take advantage of growing computational power of embedded chips and the development of optimization algorithms, especially convex optimization. For these systems which usually work at a frequency above 1Hz, it is of crucial importance that MPC can work in real-time, i.e. optimization problems must be solved on-line within a limited time. There exists a type of explicit MPC which solves optimization problems off-line analytically (Bemporad, Morari, Dua, and Pistikopoulos, 2002). However, there are growing evidence showing that on-line optimization based MPC is superior in terms of scalability and robustness. In this thesis, we focus on the on-line optimization algorithms

specialized for MPC applications.

The nonlinear nature of the world has urged scientists and engineers to develop MPC algorithms that take into account the nonlinearities of control systems. Nonlinear MPC (NMPC) considers a nonlinear mathematical model of the control system and nonlinear constraints that may be very complex to describe. The theoretical studies and industrial applications of NMPC can be found in excellent review papers and books, e.g. (Qin and Badgwell, 2000; Allgöwer and Zheng, 2012; Diehl, Ferreau, and Haverbeke, 2009; Ricker and Lee, 1995; Mayne, Rawlings, Rao, and Scokaert, 2000; Mayne, 2000). However, due to the difficulties to obtain nonlinear mathematical models, to solve the nonlinear optimization problem on-line and to implement NMPC algorithms in embedded systems, NMPC has a very limited number of applications (Qin and Badgwell, 2000), especially in the field of mechanical, electrical and electronic engineering.

Only recently, fast implementations of NMPC are becoming necessary and popular, from benchmark systems like the Pendubot (Gulan, Salaj, Abdollahpouri, and Rohal-Ilkiv, 2015), to industry relevant systems like combustion engines (Albin, Frank, Ritter, Abel, Quirynen, and Diehl, 2016), tethered airfoils (Vukov, Gros, Horn, Frison, Geebelen, Jørgensen, Swevers, and Diehl, 2015), and electrical motor drives (Bolognani, Bolognani, Peretti, and Zigliotto, 2009). Fast implementations of NMPC are particularly relevant when addressing real-time control of systems exhibiting features like fast dynamics, large dimension, and long prediction horizon, as in such situations the computational burden of the NMPC may limit the achievable control bandwidth. As a motivating example, consider dynamical driving simulators (introduced in Chapter 7), and the related Motion Cueing Algorithms (MCAs), that is, the simulator motion strategies aiming to providing the driver with a realistic driving experience. The most recent implementations of MCA (Bruschetta, Maran, and Beghi, 2017a) are based on fast MPC, which results to be a suitable technique to take into account perceptive models of the human being and models of the platforms dynamic, and, if available, information about future driver behavior. In particular, a real time linear MPC implementation has been developed with long prediction horizon, exploiting available information about future driver behavior (Beghi, Bruschetta, and Maran, 2013). Also, a real-time NMPC implementation considering a more complex model has been proposed, taking into account exact actuators constraints (Bruschetta, Maran, and Beghi, 2017c). Intuitively, applying in real-time an NMPC controller with a long prediction horizon would lead to a better driver perception and better platform work space exploitation. However, this comes at a cost of significantly increased on-line computational load, since the dimension of the on-line optimization problem and its complexity scales with the prediction horizon.

Therefore, this thesis particularly addresses fast NMPC algorithms for controlling continuous-

time dynamic systems, using a long prediction horizon for better control performance. The starting point is to build a bridge between linear and nonlinear MPC using partial linearizations. The idea is that, within the prediction window, the predicted trajectories of the control systems can be partial linearized, i.e. being linearized only when it is considered highly nonlinear. To achieve this goal, a measure of nonlinearity (MoN) for dynamic systems has to be introduced and applied to existing NMPC algorithms. However, the nonlinearity of a dynamic system is time-varying and condition dependent. A super nonlinear system, at least seems so at its expression, may become almost linear when running at its steady state. A mildly nonlinear system, may become highly nonlinear when working far from its steady state. Therefore, MoN is desired to be measured on-line to reflect the current and local nonlinearity of the control system. The resulting NMPC algorithm thus becomes a mixed linear/nonlinear MPC which requires less computational effort in the evaluation of the linearization, matrix production and factorization.

To begin with, a guiding example is shown in the following to give a first idea on when linear MPC may work well with a highly nonlinear system. This example will be referenced several times in this thesis.

1.1 A Guiding Example: Inverted Pendulum

Consider a classical nonlinear mechanical system: an inverted pendulum. The inverted pendulum is mounted on top of a cart and can swing up to 360 degrees (Quirynen, Vukov, Zanon, and Diehl, 2015). A schematic illustration is given in Fig. 1.1. The nonlinear dynamics

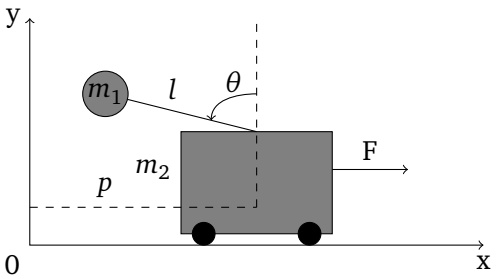


Figure 1.1: A schematic illustration of the inverted pendulum control problem. m_1, m_2, l, F are the mass of the pendulum, the mass of the cart, the length of the pendulum and the horizontal control force, respectively. The swing angle θ is defined as drawn.

are given by

$$\ddot{p} = \frac{-m_1 l \sin(\theta)\dot{\theta}^2 + m_1 g \cos(\theta)\sin(\theta) + F}{m_2 + m_1 - m_1(\cos(\theta))^2}, \tag{1.1a}$$

$$\ddot{\theta} = \frac{F\cos(\theta) - m_1 l \cos(\theta)\sin(\theta)\dot{\theta}^2 + (m_2+m_1)g\sin(\theta)}{l(m_2 + m_1 - m_1(\cos(\theta))^2)}, \tag{1.1b}$$

where p, θ are cart position and swinging angle, respectively. The values of model parameters are given in Table 1.1. Looking at the dynamic function (1.1), we can immediately make the

Variable	Value
m_1	1 kg
m_2	0.1 kg
l	0.8 m
g	9.81 ms^{-2}

Table 1.1: The values of model parameters of the inverted pendulum.

first argument that the inverted pendulum is highly nonlinear. We employ a multiple-shooting based Sequential Quadratic Programming (SQP) algorithm ((Leineweber, 1999), see also in Chapter 2) for solving the resulting nonlinear MPC problem. Suppose that the trajectory of the dynamic system (1.1) is obtained by employing a numerical integration operator Ξ as

$$x_{k+1} = \Xi(x_k, u_k), \tag{1.2}$$

where x_k, u_k are system states and controls at the prediction time point t_k. A common approach to deal with a such nonlinear problem is to linearize it, either at the steady-state or at the current transient point in state space. The linearization of the dynamic constraint (**??**) is given by

$$\Delta x_{k+1} = A_k \Delta x_k + B_k \Delta u_k + c_k. \tag{1.3}$$

where $\Delta x_k = x - x_k$ is the increment of the state variable and $A_k = \frac{\partial \Xi}{\partial x}(x_k, u_k), B_k = \frac{\partial \Xi}{\partial u}(x_k, u_k)$ are linearized transition matrices. c_k is a variable related to the integration results. We refer to *linear MPC* (LMPC) using constant matrices $A_k = A, B_k = B$ [1] and *nonlinear MPC* (NMPC) using locally linearized dynamics. In the sequel, we will investigate two control tasks using these two MPC algorithms and illustrate how a linear MPC is able to control a nonlinear system in a certain circumstances.

[1] Precisely speaking, this is a semi-linear scheme as described in Chapter 2

Invert the pendulum

The first control task is to invert the pendulum from bottom to top, i.e. $\theta : \pi \to 0$. At each sampling instant, the following nonlinear programming problem is solved

$$\min_{x,u} \quad \sum_{k=0}^{N-1} \|x_k - x_{ref}\|_Q + \|u_k\|_R + \|x_N - x_{ref}\|_{Q_N} \tag{1.4a}$$

$$s.t. \quad x_0 = \hat{x}_0, \tag{1.4b}$$

$$x_{k+1} = \Xi(x_k, u_k),\ k = 0, 1, \ldots, N-1, \tag{1.4c}$$

$$-2 \le p_k \le 2, k = 1, 2, \ldots, N, \tag{1.4d}$$

$$-20 \le u_k \le 20, k = 0, 1, \ldots, N-1, \tag{1.4e}$$

where $x = [p, \theta, \dot{p}, \dot{\theta}]^\top$ is system state and $u = F$ the control input. The initial condition is given by $\hat{x}_0 = [0, \pi, 0, 0]^\top$ and the reference is $x_{ref} = [0, 0, 0, 0]^\top$. The values of tuning parameters are given in Table 1.2. The simulation results are shown in Figure 1.2. In this

Variable	Description	Value
Q	weight on state	diag(10,10,0.1,0.1)
R	weight on control	0.01
Q_N	terminal weight	diag(10,10,0.1,0.1)
N	prediction horizon	40
T_s	sampling time	0.05 s

Table 1.2: The values of NMPC tuning parameters of the inverted pendulum.

situation, NMPC has perfectly completed the control task while LMPC is always struggling to achieve the goal. No wonder LMPC has such a terrible performance due to the highly nonlinearity of the system at hand. However, it is too early to draw a conclusion that a linear MPC is not suitable for the inverted pendulum.

Shake the pendulum

The second control task is to "shake" the pendulum around its unstable equilibrium point $\theta = 0$. Provided that the shaking angle is sufficiently small, a locally linear behavior can be

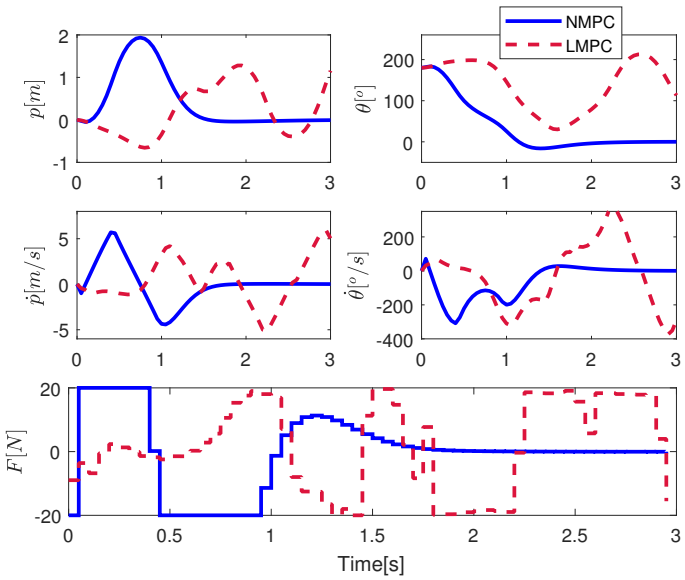


Figure 1.2: State and control trajectories of the inverted pendulum using NMPC and LMPC for the first control task. For LMPC, the static transition matrices are obtained by linearizing the nonlinear system (1.1) at the initial point.

expected. In this situation, the following nonlinear programming problem is solved

$$\min_{x,u} \quad \sum_{k=0}^{N-1} \|\theta_k - \theta_{ref}\|_Q + \|\dot{u}_k\|_R + \|\theta_N - \theta_{ref}\|_{Q_N} \tag{1.5a}$$

$$s.t. \quad x_0 = \hat{x}_0, \tag{1.5b}$$

$$x_{k+1} = \Xi(x_k, u_k),\ k = 0, 1, \ldots, N-1, \tag{1.5c}$$

$$-20 \leq u_k \leq 20, k = 0, 1, \ldots, N-1, \tag{1.5d}$$

where $\hat{x}_0 = [0, 0, 0, 0]^\top$, $Q = 100$, $R = 10^{-3}$ and other tuning parameters are identical to those in Table 1.2. In Figure 1.3, the simulation results using NMPC and LMPC for shaking the pendulum up to 5 degrees are shown. Although there are some input oscillations at around $t = 4$s, the performance of LMPC is in general very similar to that of NMPC. This demonstrates that a linear MPC scheme can work with a highly nonlinear system in a certain circumstances, as long as the local nonlinearity is small enough. Indeed, if the shaking angle is too big ,e.g. up to 20 degrees, this linear MPC can no longer provide a reliable control to the inverted pendulum.

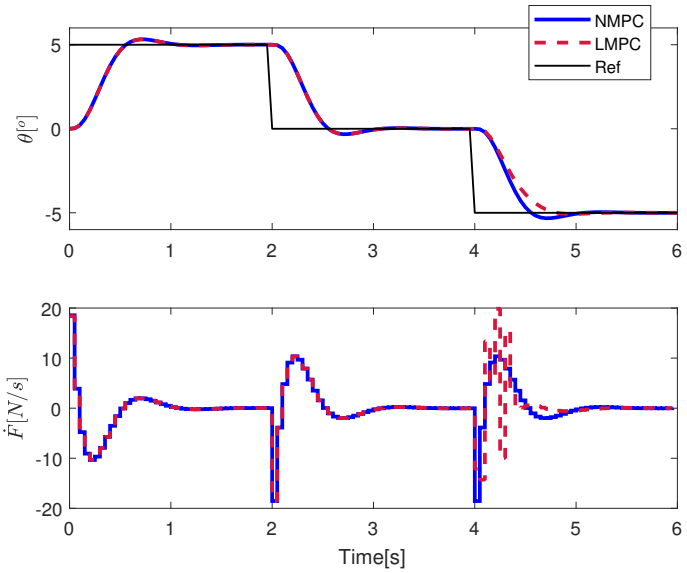


Figure 1.3: State and control trajectories of the inverted pendulum using NMPC and LMPC for the second control task. For LMPC, the static transition matrices are obtained by linearizing the nonlinear system (1.1) at the initial point. The shaking angle has the range of $[-5^o, 5^o]$.

1.2 The Second Example: Chain of Masses

The second example is a chain of masses which is a mechanical system with a number of masses connected by springs. One end of the chain is fixed on a wall and the other end is controlled by external forces. A schematic illustration is shown in Figure 1.4. To stabilize the chain with n masses in the X-Z plane, the following NLP problem is solved at each sampling instant.

$$\min_{\mathbf{x},u} \sum_{i=0}^{N} \quad (\sum_{j=1}^{n} \|v_j^i\|_{W_v}^2 + \|p_n^i - p_n^i\|_{W_p}^2) + \sum_{i=0}^{N-1} \sum_{j=1}^{n} \|u_j^i\|_{W_u}^2 \tag{1.6a}$$

$$s.t. \quad \dot{p}_j = v_j,\ j = 1,\dots,n-1, \tag{1.6b}$$

$$\dot{v}_j = \frac{n}{m}(F_{j+1} - F_j) - g,\ j = 1,\dots,n-1, \tag{1.6c}$$

$$\dot{p}_n = u, \tag{1.6d}$$

where $p_i \in \mathbb{R}^3, v_i \in \mathbb{R}^3, u \in \mathbb{R}^3$ are the positions, velocities of the masses and the control input in 3D space, respectively, and

$$F_j = k(x_j - x_{j-1})(n - \frac{l_r}{||x_j - x_{j-1}||_2}),\ j = 1,\dots,n-1, \tag{1.7}$$

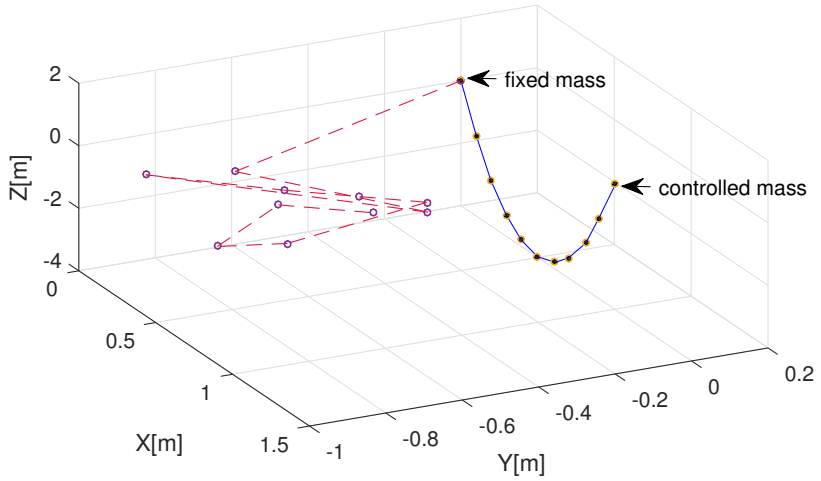


Figure 1.4: A schematic illustration of the chain of masses system.

is the force generated by the spring between mass j and $j-1$. System state is $\mathbf{x} = [p_0^\top, \ldots, p_n^\top, v_0^\top, \ldots, v_{n-1}^\top]^\top$ and parameters are k, m, l_r, g with values given in (Kirches, Wirsching, Bock, and Schlöder, 2012).

Problem (1.6) penalizes the velocities of every mass and the final position of the last mass. The steady positions for each mass will automatically be obtained by solving (1.6) until optimum. Although the spring dynamics (1.7) are linear, it contains the distance between two masses which results in a nonlinear model. This example will be used for testing algorithms developed in this thesis as it allows for arbitrary number of states. Therefore, a sufficiently large system can be built.

1.3 Contributions and Outline

Example 1.1 shows that the nonlinearity of a system is condition dependent and a linear MPC may be capable of controlling nonlinear systems when considered locally linear. As the nonlinearity of a system is varying, a mixed strategy that combines linear and nonlinear MPC is expected to provide acceptable control performance. Therefore, if optimization algorithms can make use of such mixed structure, their computational efficiency can be significantly improved. In the following, the structure of this thesis is described as in Fig. 1.5 that addresses different aspects of on-line optimization algorithms.

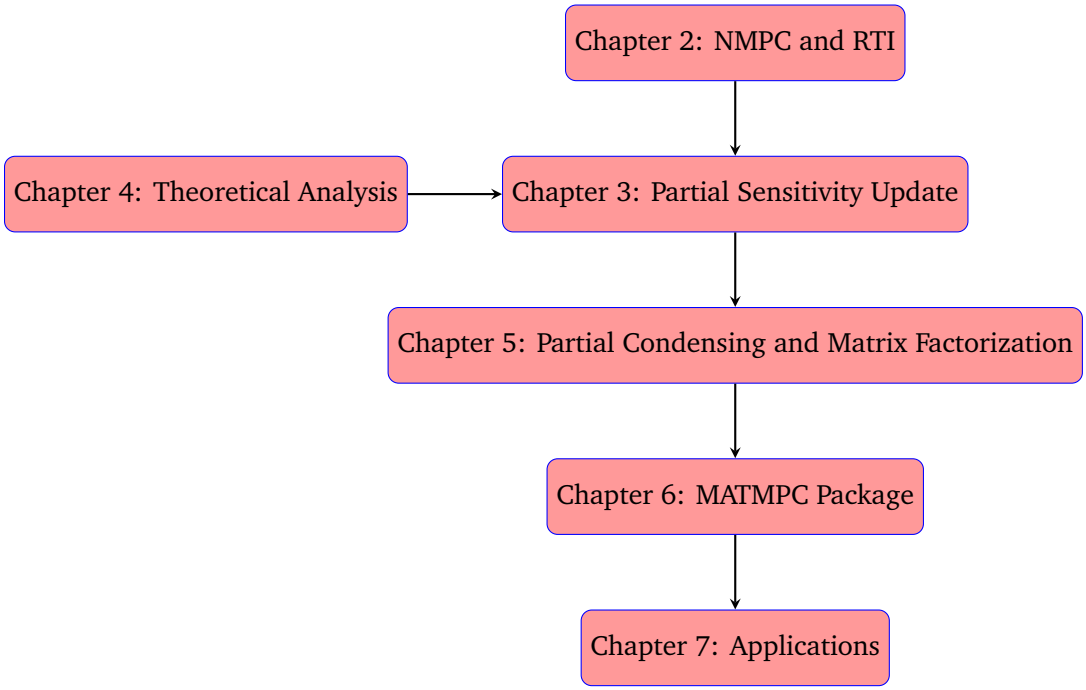


Figure 1.5: Structure of the thesis.

Chapter 2 - Fast Nonlinear Model Predictive Control Algorithms

This chapter introduces a popular algorithmic framework for optimal control problem formulation, which is the target of the thesis. It starts with an overview of direct multiple shooting and nonlinear programming algorithms. In particular, fast algorithms to solve nonlinear programming problem on-line in the scenario of NMPC are discussed. The Real-Time Iteration (RTI) scheme (Diehl, Bock, Schlöder, Findeisen, Nagy, and Allgöwer, 2002) is one of the most popular fast NMPC algorithms and is presented in detail. In addition, several inexact sensitivity schemes in the framework of RTI are introduced, which will be used for comparisons with new developments in this thesis.

Chapter 3- Partial Sensitivity Update

In this chapter, we present a framework for partially updating sensitivities of optimization problems using a Curvature-like Measure of Nonlinearity (CMoN). First the classical methods of Measure of Nonlinearity (MoN) are introduced. Then a novel Curvature-like Measure of Nonlinearity (CMoN) is proposed and analyzed in detail for dynamic systems. Finally, two updating logic are proposed in this framework, including a fixed-time block updating logic and an adaptive one with a fixed threshold. The numerical and control performance of each

updating logic are presented by means of simulations. Results in this chapter are mainly from paper (Chen, Cuccato, Bruschetta, and Beghi, 2017a) and (Chen, Davide, Bruschetta, and Beghi, 2017b).

Chapter 4- Optimality and Convergence of Partial Sensitivity Update

This chapter gives an analysis of the optimality and convergence properties of partial sensitivity updating presented in the previous chapter. An updating logic that uses adaptive thresholds is proposed. The resulting inexact sensitivity problem is re-formulated as a parametric problem and the parameter has been related to the distance to optimum (DtO). As a result, the tuning of the thresholds is converted to the tuning of DtO tolerance, which is defined by users. By this means, a partial sensitivity updating scheme is available with a guarantee on DtO. Such scheme is extended to the case of Sequential Quadratic Programming and its local convergence is proved. Both numerical and simulation examples are given. Results of this chapter have been written in a journal paper under review by now.

Chapter 5- Partial Condensing and Matrix Factorization

We further investigate possible improvements for a RTI step after partial sensitivity update. Given that the QP subproblem can be solved either in a sparse form or a dense one, efficient algorithms are proposed in both cases. To solve a dense QP, two algorithms are proposed by combining partial condensing and partial sensitivity update, with a quadratic and linear complexity in prediction horizon length, respectively. To solve a sparse QP, a partial matrix factorization algorithm is proposed to exploit part of the KKT matrix factorization from previous sampling instants. Comparison is made in computational cost in terms of floating point operations (flops). Results of this chapter will be submitted to a conference soon.

Chapter 6- MATMPC: a Matlab-based Nonlinear MPC package

This chapter discuss an implementation of NMPC algorithms based Matlab, which is called MATMPC. The motivation of MATMPC is firstly addressed, and its code structure is introduced. MATMPC has two working modes: one is in pure Matlab environment and the other is constructed using MATLAB C language API. We show how the first mode is useful and flexible for non-experts in computer science and programming when developing and debugging their solutions. The second mode shows comparable numerical performance w.r.t state-of-the-art NMPC solvers based on C code generation.

Chapter 7- Applications to Automotive Industry

A complex, real-world problem, named as dynamic driving simulator, is tackled in this chapter. First, a simulator with nine degree of freedoms connected by tripod and hexapod is introduced. A nonlinear model with four reference frames is derived and a high-pass and low-pass filters are used to distinguish the movement of the tripod and hexapod. Actuator constraints, which are nonlinear and very complex, are taken into account. A real-time implementation is obtained by using MATMPC working in its second mode. Second, a Multi-Sensory Cueing with Active Seat framework is constructed. The coupling between simulator's movements and the use of active seat is explicitly embedded into the dynamic model. The NMPC implementation can be run in real-time by means of adjoint RTI scheme and the alternating direction method of multipliers (ADMM). Results of this paper are from (Bruschetta, Cunico, Chen, Beghi, and Minen, 2017b).

2

Nonlinear Model Predictive Control

Model Predictive Control (MPC) is a an advanced computer control algorithm that exploits explicit process models and constraints, and the power of optimization. The key feature of MPC is to determine a control sequence at time t_0 in order to optimize a cost function representing the future behavior of the controlled process over a prediction horizon $[t_0, t_f]$. However, only the first control input is forwarded to the plant and the optimization procedure is repeated at the next sampling instant. As a result, MPC refers to a control strategy that solves a sequence of optimization problems on-online (when system is running), with respect to the latest state measurement of the plant. Figure 2.1 shows the building blocks of MPC. Firstly proposed in 1980s (Cutler and Ramaker, 1980), MPC has become a mature modern control technique in the application of chemical , aerospace, paper manufacturing and automotive industries (Garcia et al., 1989; Qin and Badgwell, 2003; Camacho and Alba, 2013).

Nonlinear Model Predictive Control (NMPC) refers to MPC exploiting nonlinear plant models. NMPC can provide better control performance and increasing productivity by accurately capturing the nonlinear behaviors of plants. The theoretical studies and industrial applications of NMPC can be found in excellent review papers and books, e.g. (Qin and Badgwell, 2000; Allgöwer and Zheng, 2012; Diehl et al., 2009; Ricker and Lee, 1995; Mayne et al., 2000; Mayne, 2000).

However, the limitation of NMPC is obvious. While the optimization problems arising from linear MPC can be solved reliably and in a extreme fast speed, the optimization problems

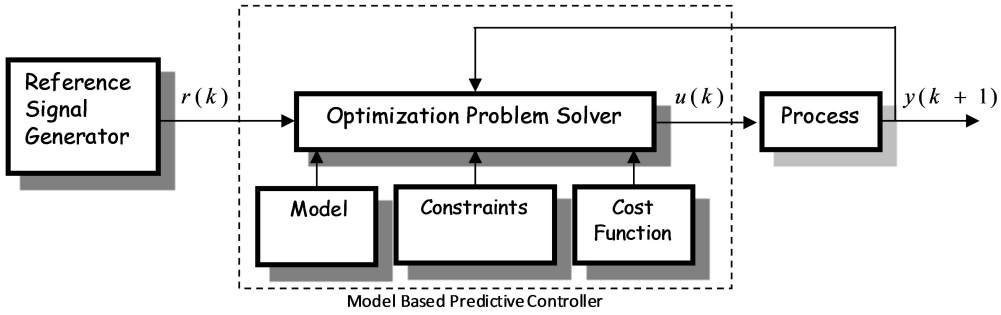


Figure 2.1: An illustrative building block of MPC

in NMPC are often hard to solve and are difficult to solve within the real-time restriction. This is due to the difficulty of reliably solving an open-loop nonlinear optimal control problem at each sampling instant, respecting state, control and other physical and safety constraints. In this chapter, we review how NMPC is formulated and solved using modern algorithms and some existing fast NMPC algorithms that intend to accelerate the on-line optimization.

2.1 Optimal Control Problem

Consider a nonlinear plant which is modeled by a first principle model like the following

$$\dot{x}(t) = f(t, x(t), u(t); p), \tag{2.1}$$

where $x \in \mathcal{R}^{n_x}, u \in \mathcal{R}^{n_u}, p \in \mathcal{R}^{n_p}$ are plant state, control and parameter variables. Equation (2.1) refers to a system of Ordinary Differential Equations (ODE) and represents the dynamics of the controlled plant [1]. Starting at an initial condition $x(t_0) = \hat{x}_0$, (2.1) is an Initial Value Problem (IVP) given a control trajectory $u(t)$. The existence of a unique solution $x(t)$ on a certain interval $t \in [t_0, t_f]$ is proved by the famous Picard's existence theorem, under the assumption that f is uniformly Lipschitz continuous in x, u and continuous in t. We follow the same assumption throughout the thesis.

MPC requires to solve an open-loop Optimal Control Problem (OCP) at each sampling

[1] For simplicity we consider ODE systems throughout the thesis, although there are systems modeled by Differential-Algebraic Equations

instant, respecting the dynamics of the plant (2.1), as follows

$$\min_{x(\cdot),u(\cdot)} \quad \tilde{J} = \int_{t_0}^{t_f} \phi(t, x(t), u(t); p)\mathrm{d}t + \Phi(x(t_f)) \tag{2.2a}$$

$$s.t. \quad x(t_0) = \hat{x}_0, \tag{2.2b}$$

$$\dot{x}(t) = f(t, x(t), u(t); p), \quad \forall t \in [t_0, t_f], \tag{2.2c}$$

$$r(x(t), u(t); p) \leq 0, \quad \forall t \in [t_0, t_f], \tag{2.2d}$$

$$l(t_f) \leq 0, \tag{2.2e}$$

where ϕ and Φ are the optimization *objective* of the OCP (2.2), r the *path constraints* which includes state and input constraints, l the *boundary conditions*. Given the current state measurement $\hat{x}_0$, an optimal solution trajectory $(x(\cdot), u(\cdot))$ in the range of the *prediction length* $t_f - t_0$ can be obtained by solving (2.2).

2.2 Direct Multiple Shooting

There are three well-known approaches for solving the OCP (2.2) (Binder, Blank, Bock, Bulirsch, Dahmen, Diehl, Kronseder, Marquardt, Schlöder, and von Stryk, 2001; Quirynen, 2017):

1. *Dynamic Programming* using Hamilton-Jacobi-Caratheodory-Bellman(HJCB) and partial differential equations (PDEs),

2. *Indirect Methods* using Calculus of Variations and Pontryagin's Maximum Principle, and

3. *Direct Methods* based on a finite dimensional parameterization of the continuous-time OCP (2.2).

Indirect methods are hardly used in today's NMPC applications since their shortcomings. In (Rao, 2009), Rao wrote that "indirect shooting method is extremely sensitive to the unknown boundary condtions" and "requires the derivation of the first-order optimality conditions of the OCP". A more complete analysis of the shortcomings of indirect methods is addressed in (Binder et al., 2001).

On the other hand, direct methods are more popular mainly due to their flexibility and the fast development of numerical optimization solvers. Applying a finite dimensional parameterization to the OCP (2.2), we obtain a Nonlinear Programming (NLP) problem which can be solved by state-of-art numerical solvers. There exist a variety of approaches to parameterize the state and control variables in (2.2), e.g. single shooting, multiple

shooting and collocation, however, we focus on multiple shooting throughout the thesis. This is because multiple shooting has been demonstrated an effective approach in NMPC applications (Leineweber, 1999; Diehl et al., 2002) and the fast NMPC algorithms described in the later chapters are based on this kind of parameterization. For a complete overview of different parameterization approaches, see (Binder et al., 2001; Rao, 2009).

In direct multiple shooting, the prediction length is divided into N *shooting intervals* defined as $[t_k, t_{k+1}]$, $k = 0, 1, \ldots, N-1$ resulting in $N+1$ time grid points $t_0 < t_1 < \ldots < t_N = t_f$. The control trajectory is then parameterized over these shooting intervals, usually by a piece-wise constant representation given by

$$u(t) = u_k, \quad \forall t \in [t_k, t_{k+1}). \tag{2.3}$$

The application of higher order parameterization of the control trajectory is beyond the scope of the thesis. Differently from single shooting, multiple shooting also parameterizes the state trajectory $x(t)$ over the N shooting intervals by considering the dynamics of the plant (2.1). A total of $N+1$ *shooting points* $(s_0, s_1, \ldots, s_N)$ are introduced as additional optimization variables and each shooting point $s_k, k = 0, 1, \ldots, N$, is defined exactly on the time grid point t_k. In multiple shooting, s_k is the initial condition of the following IVP over the shooting interval $[t_i, t_{i+1})$

$$\dot{x}_k(t) = f(t, x_k(t), u_k(t); p), \; \forall t \in [t_k, t_{k+1}), \; x_k(t_k) = s_k, \tag{2.4}$$

so that the dynamic constraints (2.2c) are transcribed into *continuity constraints* as

$$s_{k+1} = \Xi(t_k, s_k, u_k; p), \quad k = 0, 1, \ldots, N-1, \tag{2.5}$$

where $\Xi(\cdot)$ is an integration operator which solves the IVP (2.4) and returns the solution at the terminal time point t_{k+1}. Similarly, the path constraints (2.2c) are parameterized on the discrete time points given by

$$r(s_k, u_k; p) \leq 0, \quad , k = 0, 1, \ldots, N-1. \tag{2.6}$$

The optimization objective $\tilde{J}$ in (2.2a) is re-formulated as

$$\sum_{k=0}^{N-1} \bar{J}_k = \sum_{k=0}^{N-1} \int_{t_k}^{t_{k+1}} \phi(t_k, s_k, u_k; p) \mathrm{d}t + \Phi(s_N), \tag{2.7}$$

and can be approximated using a discrete sum (Leineweber, 1999; Quirynen, 2017) as

$$\sum_{k=0}^{N-1} J_k = \sum_{k=0}^{N-1} \phi(t_k, s_k, u_k; p) + \Phi(s_N). \tag{2.8}$$

Given the specific parameterizations (2.5), (2.6) and (2.8), a NLP problem is formulated as

$$\begin{aligned}
\min_{\mathbf{s},\mathbf{u}} \quad & \sum_{k=0}^{N-1} \phi(t_k, s_k, u_k; p) + \Phi(s_N) && \text{(2.9a)}\\
s.t. \quad & s_0 = \hat{x}_0, && \text{(2.9b)}\\
& s_{k+1} = \Xi(t_k, s_k, u_k; p),\ k = 0, 1, \ldots, N-1 && \text{(2.9c)}\\
& r(s_k, u_k; p) \leq 0,\ k = 0, 1, \ldots, N-1, && \text{(2.9d)}\\
& l(s_N) \leq 0, && \text{(2.9e)}
\end{aligned}$$

where $\mathbf{s} = [s_0^\top, s_1^\top, \ldots, s_N^\top]^\top$ and $\mathbf{u} = [u_0^\top, u_1^\top, \ldots, u_{N-1}^\top]^\top$ are discrete state and control variables. The features of multiple shooting parameterization and the resulting NLP (2.9) are listed as follows.

1. An initial guess of the entire state and control trajectory over the prediction horizon is needed to solve (2.9). However, this initial guess is not necessarily feasible.

2. The resulting NLP (2.9) is numerically stable, even if the dynamics (2.1) is unstable or even chaotic

3. Existing software are directly available for solving $\Xi(\cdot)$ of (2.9c) and the NLP (2.9). There is no need to re-design the NMPC for different problems at hand.

4. The NLP (2.9) is also called a *multi-stage problem* (Zanelli, Domahidi, Jerez, and Morari, 2017), as the objective (2.8), constraints (2.5) and (2.6) can be easily decoupled on different shooting intervals or stages. Parallel computation is straightforward.

It should be noted that all the constraints in (2.9), particularly the continuity constraints (2.9c), are usually violated at the beginning and in the process of solving (2.9) using an iterative approach. Only at the optimal solution (up to a given accuracy) the constraints (2.9) will be fulfilled. This is illustrated in Figure 2.2 and 2.3.

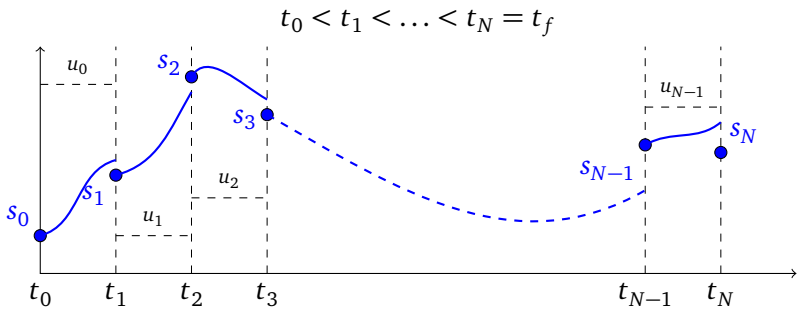


Figure 2.2: The initial discontinuous trajectory of multiple shooting

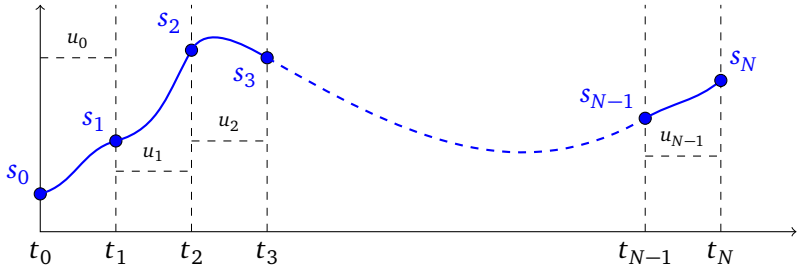


Figure 2.3: The terminal continuous trajectory of multiple shooting

2.3 Nonlinear Programming Algorithms

The NLP problem (2.9) can be re-written in a standard or compact form as

$$\min_{\mathbf{z}} \quad a(\mathbf{z}) \tag{2.10a}$$
$$s.t. \quad b(\mathbf{z}) = 0, \tag{2.10b}$$
$$c(\mathbf{z}) \leq 0, \tag{2.10c}$$

where

$$\mathbf{z} = [z_0^\top, z_1^\top, \ldots, z_{N-1}^\top, s_N]^\top \in \mathbb{R}^{n_z}, \tag{2.11}$$
$$z_k = [s_k^\top, u_k^\top]^\top \in \mathbb{R}^{n_x+n_u},\ k = 0, 1, \ldots, N-1, \tag{2.12}$$

collects all the optimization variables and $b : \mathbb{R}^{n_z} \to \mathbb{R}^{n_b}, c : \mathbb{R}^{n_z} \to \mathbb{R}^{n_c}$ collects all the constraints, which are given by

$$b(\mathbf{z}) = \begin{bmatrix} \hat{x}_0 - s_0 \\ \Xi(s_0, u_0) - s_1 \\ \vdots \\ \Xi(s_{N-1}, u_{N-1}) - s_N \end{bmatrix}, c(\mathbf{z}) = \begin{bmatrix} r(s_0, u_0) \\ \vdots \\ r(s_{N-1}, u_{N-1}) \\ l(s_N) \end{bmatrix} \tag{2.13}$$

This standard NLP problem (2.10) can be solved via efficient NLP solvers, such as Ipopt (Wächter and Biegler, 2006), Knitro (Byrd, Nocedal, and Waltz, 2006) and SNOPT (Gill, Murray, and Saunders, 2005). A comprehensive classification and comparison of optimization solvers with open-source availability and commercial license is given by (Mittelmann, 2016). For NMPC practitioners and users, the only remaining step is to select an appropriate NLP solver. However, for researchers, it is important to go deep into the optimization to develop and improve optimization algorithms. Mathematically speaking, there are mainly two classes of NLP optimization algorithms, namely Sequential Quadratic Programming (SQP) and Interior Point Method (IPM) (Nocedal and Wright, 2006). A comparison of these methods for solving optimal control problems can be found in (Works, 2002).

For clarity we first introduce the Karush-Kuhn-Tucker (KKT) first order necessary conditions of optimality of (2.10). Assumming $\mathbf{z}^*$ a *local minimizer* of (2.10), there exist KKT multipliers $\lambda^* \in \mathbb{R}^{n_b}, \mu^* \in \mathbb{R}^{n_c}$ corresponding to constraints (2.10b) and (2.10c) respectively, such that

Stationarity

$$\nabla_{\mathbf{z}} \mathcal{L}(\mathbf{z}^*, \lambda^*, \mu^*) := \nabla_{\mathbf{z}} a(\mathbf{z}^*) + \nabla_{\mathbf{z}} b(\mathbf{z}^*)^\top \lambda^* + \nabla_{\mathbf{z}} c(\mathbf{z}^*)^\top \mu^* = 0, \tag{2.14a}$$

Primal feasibility

$$b(\mathbf{z}^*) = 0, \tag{2.14b}$$

$$c(\mathbf{z}^*) \leq 0, \tag{2.14c}$$

Dual feasibility

$$\mu^* \geq 0, \tag{2.14d}$$

Complementary slackness

$$\mu_k^* c_k(\mathbf{z}^*) = 0,\ k = 0, 1, \ldots, n_c. \tag{2.14e}$$

where $y^* = [\mathbf{z}^{*\top}, \lambda^{*\top}, \mu^{*\top}]^\top$ is the optimal primal and dual solutions and is called a KKT point. Throughout the thesis, we also assume the fulfillment of Linear Independence Constraint Qualification (LICQ), Strict Complementary Slackness (LCS) and Second Order Sufficient

Conditions (SOSC). These are general assumptions in most optimization problems and detailed descriptions can be found in any relevant textbooks, e.g. (Nocedal and Wright, 2006).

2.3.1 Sequential quadratic programming

The idea of Sequential Quadratic Programming (SQP) (Han, 1976; Powell, 1978) is to solve the NLP problem (2.10) iteratively by using a local quadratic approximation of the objective and linearized constraints at each iteration. Specifically, given an initial guess $y^i = [\mathbf{z}^{i\top}, \lambda^{i\top}, \mu^{i\top}]^\top$ at the i^{th} iteration, a standard Quadratic Programming (QP) problem is formulated as follows

$$\min_{\Delta \mathbf{z}} \quad \frac{1}{2}\Delta \mathbf{z}^\top H(\mathbf{z}^i)\Delta \mathbf{z} + g(\mathbf{z}^i)^\top \Delta \mathbf{z} \tag{2.15a}$$

$$s.t. \quad b(\mathbf{z}^i) + B(\mathbf{z}^i)\Delta \mathbf{z} = 0, \tag{2.15b}$$

$$c(\mathbf{z}^i) + C(\mathbf{z}^i)\Delta \mathbf{z} \leq 0, \tag{2.15c}$$

where $\Delta \mathbf{z} = \mathbf{z} - \mathbf{z}^i$ is the increment, $H(\mathbf{z}^i) := \nabla^2_{\mathbf{z}} \mathcal{L}(\mathbf{z}^i, \lambda^i, \mu^i)$ the *exact* Hessian of the Lagrangian, $g(\mathbf{z}^i) := \nabla_{\mathbf{z}} a(\mathbf{z}^i)$ the gradient of the objective, $B(\mathbf{z}^i) := \nabla_{\mathbf{z}} b(\mathbf{z}^i)$ and $C(\mathbf{z}^i) := \nabla_{\mathbf{z}} c(\mathbf{z}^i)$ the Jacobian matrices of the constraints b, c, respectively. The resulting solution is the primal increment $\Delta \mathbf{z}$ together with new multipliers λ^{i+1}, μ^{i+1}. The initial guess y^i is then updated according to

$$\mathbf{z}^{i+1} = \mathbf{z}^i + \alpha^i \Delta \mathbf{z}, \tag{2.16a}$$

$$\lambda^{i+1} = (1-\alpha)\lambda^i + \alpha^i \lambda^{i+1}, \tag{2.16b}$$

$$\mu^{i+1} = (1-\alpha)\mu^i + \alpha^i \mu^{i+1}, \tag{2.16c}$$

where α^i is the *step size* and can be determined by globalization strategies (Nocedal and Wright, 2006). This procedure is repeated until the satisfaction of the KKT conditions (2.14) up to a given accuracy. SQP is proved to exhibit a quadratic or superlinear convergence rate given a sufficiently good initial guess (Han, 1976) and is widely used for solving NLP problems arising from NMPC applications (Leineweber, 1999; Binder et al., 2001; Rao, 2009; Diehl et al., 2009).

Solving the QP subproblem

In addition to the fast convergence rate, SQP is able to exploit existing solvers for solving QP subproblems, which are typical in linear MPC applications. Therefore, the algorithmic effort

needed for upgrading the linear MPC to NMPC is usually affordable. For linear MPC, efficient and reliable QP algorithms and solvers have been developed for decades. There are three main classes of algorithms for solving QPs (Kouzoupis, Zanelli, Peyrl, and Ferreau, 2015c; Ferreau, Almer, Verschueren, Diehl, Frick, Domahidi, Jerez, Stathopoulos, and Jones, 2017) which are summarized in the following.

1. *First-order methods* are the simplest and fastest algorithms for solving QPs with simple constraints, e.g. box constraints on optimization variables. Fast primal and dual gradient methods (Nesterov, 2013), multiplicative update dual optimization (Di Cairano and Brand, 2013) and Alternating Direction Method of Multipliers (ADMM) (Boyd, Parikh, Chu, Peleato, and Eckstein, 2011; O'Donoghue, Stathopoulos, and Boyd, 2013) all belong to this class. A given solution accuracy can be achieved after finite iterations. In case of equality and polytopic constraints, primal gradient methods often struggle or fail the optimization but dual gradient methods and ADMM are able to proceed. A code generation tool that implements first-order methods is FiOrdOs (Ullmann, 2011).

2. *Active-set methods* first guess the active inequality constraints and consider them as equality constraints together with existing ones. As a consequence, a linear system is solved if the guess is correct, otherwise the algorithms take new guesses and update the solution until optimum. The most promising advantage of active-set methods is the possibility to use warm-start strategy, which exploit an initial guess from the past solution. By this means, only a few iterations are needed. A fantastic active-set QP solver is qpOASES (Ferreau, Kirches, Potschka, Bock, and Diehl, 2014).

3. *Interior point methods* remove the inequality constraints by introducing slack variables. A penalty term or *barrier parameter* is used in the objective to ensure that the entire iterative procedure is feasible, so that the solution trajectory is in the "interior" of the feasible region. The most famous interior point method for solving convex QP problems is probably Mehrotra's predictor-corrector scheme (Mehrotra, 1992). NLP solvers using interior point methods such as Ipopt and Knitro (Wächter and Biegler, 2006; Byrd et al., 2006), are capable of solving QP subproblems. The algorithmic performance can be further improved if the structure of the QP is exploited, e.g. Forces Pro (Domahidi and Jerez, 2014) solves multi-stage QP problems linearized from the multi-stage NLP problems.

Linear MPC practitioners can select suitable QP solvers depend on their applications and the features of solvers. However, additional considerations are necessary when implementing SQP algorithm in NMPC applications. When the plant model is nonlinear, all the QP data in (2.15) are in general time-varying hence advantages of some QP algorithms may disappear.

For instance, for fast gradient methods, the optimal step size is computationally expensive since it require to calculate the eigenvalues of the Hessian matrix. This calculation can be moved off-line if the model is linear and the Hessian is time-invariant. However, on-line computation is not affordable for nonlinear models (Kouzoupis, Ferreau, Peyrl, and Diehl, 2015a). As we will show in the following chapters, the type of constraints, the number of active constraints, the number of shooting points and dimension of the model are all important factors for selecting appropriate QP solvers in the framework of SQP in NMPC.

2.3.2 Interior Point Methods

Interior point methods (IPM) are arguably the best algorithms for solving a general constrained NLP problem in the form of (2.10). Here we focus on solving NLP problems, although QP problem is a special class of NLP problem and can be solved via IMP as described in Chapter 2.3.1.

The idea of interior point methods is to solve the following modified NLP problem by introducing a barrier parameter $\tau > 0$:

$$\min_{\mathbf{z},s} \quad a(\mathbf{z}) - \tau \sum_{i=1}^{n_c} \log s_i \tag{2.17a}$$

$$s.t. \quad b(\mathbf{z}) = 0, \tag{2.17b}$$

$$c(\mathbf{z}) + s = 0, \tag{2.17c}$$

where s is a slack variable and is inherently non-negative as the minimization of the objective prevents s from being too close to zero. IPM iteratively finds approximate solutions of (2.17) for a sequence of positive barrier parameters $\{\tau_i\}$ that converges to zero (Nocedal and Wright, 2006).

Given the value of τ, the equality constrained NLP (2.17) can be solved by applying Newton's method to the nonlinear system characterized by the KKT condition of (2.17). As a result, at each iteration, we solve the following linear system

$$\begin{bmatrix} H & \nabla_{\mathbf{z}} b^\top & \nabla_{\mathbf{z}} c^\top & \\ \nabla_{\mathbf{z}} b & & & \\ \nabla_{\mathbf{z}} c & & & I \\ & & S & \mathcal{M} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{z} \\ \Delta \lambda \\ \Delta \mu \\ \Delta s \end{bmatrix} = - \begin{bmatrix} \nabla_{\mathbf{z}} \mathcal{L} \\ b(\mathbf{z}) \\ c(\mathbf{z}) \\ S\mu - \tau e \end{bmatrix}, \tag{2.18}$$

where H has the same definition as in (2.15), $S, \mathcal{M}$ diagonal matrices with vectors s, μ on their diagonal entries and $e = [1, 1, \ldots, 1]^\top$. Such a linear system can be solved either by $LDL^\top$ factorization as in CVXGEN (Mattingley and Boyd, 2012), or by structure exploiting

factorization as in Forces Pro (Domahidi and Jerez, 2014).

A key factor for the success of IPM is the barrier parameter updating strategy. While Mehrotra's predictor-corrector scheme (Mehrotra, 1992) has shown excellent performance for convex QP problems, such a scheme is not necessarily that good for NLP problems (Nocedal and Wright, 2006; Zanelli et al., 2017). A monotonically decreasing scheme for τ is shown to be robust (Nocedal and Wright, 2006). Other schemes like adaptive barrier parameter (Nocedal, Wächter, and Waltz, 2009) and centering parameter (Vanderbei and Shanno, 1999) are also widely used.

Another key factor of IPM is the globalization strategy. There are mainly three types of globalization strategies (Nocedal and Wright, 2006): i) line-search by merit function; ii) line-search by filter; iii) trust-region methods. As indicated in (Ferreau et al., 2017), the filter line-search method is most popular and has been adopted by state-of-art IPM solvers, including Ipopt (Wächter and Biegler, 2006), Forces NLP (Zanelli et al., 2017) and PIPS-NLP (Chiang, Huang, and Zavala, 2017).

2.4 Fast NMPC Algorithms

While an introduction of NLP optimization algorithms is addressed in Chapter 2.3, the focus of this thesis is not on general numerical optimization. NMPC control engineers are willing to solve NLP problems arising in NMPC applications instead of general ones from finance or computer science. Since the real-time requirement of NMPC is so critical, algorithms are desired to exploit the specialty of NLP problems from NMPC for possible speed-up. Therefore, a summary is given in the following on the features of NMPC NLP problems and their effects on optimization algorithms.

1. NLP problems at two consecutive sampling instants are very similar if the sampling frequency is high enough. This may help select initial guess and pre-solve the problem.

2. Multi-stage problems in the form of (2.9) results in banded block diagonal Hessian and constraint Jacobians, which can be exploited by solvers.

3. NMPC often requires only a moderate accuracy of the NLP solution. Inexact NLP algorithms can often be used.

Existing fast NMPC algorithms more or less exploit the above features and deliver a suboptimal solution with a certain degree of optimality guarantee. In the following we introduce three popular fast NMPC algorithms.

1. *Continuation/GMERES method* (Ohtsuka, 2004) applies IPM with a single Newton step and use a *tangential predictor* to approximate the optimal solution. However, as pointed

out in (Ferreau et al., 2017), the solution manifold becomes highly nonlinear at an active-set change, which leads to a less accurate predictor compared to SQP methods.

2. *Advanced-step NMPC* (Zavala and Biegler, 2009) also applies IPM but in a fully convergence manner. A future NLP problem can be pre-solved at the current sample using predicted states. The solution of the future NLP problem is then corrected at the corresponding sample by exploiting parametric property of the NLP problem. The pre-solving process, which is usually computationally expensive, can be run in background, while the correcting step is able to deliver a solution to the plant very quickly, resulting in much less feedback delay. However, advanced-step NMPC also suffers from the increasing nonlinearity due to active-set changes.

3. *Real-time Iteration (RTI)* (Diehl et al., 2002) is an early-stop SQP algorithm which performs only one SQP iteration for solving NLP problem (2.10). An *initial value embedding* strategy is exploited so that the resulting suboptimal solution is a tangential predictor of the exact solution, even at the presence of active-set changes. The on-line optimization process is divided into a preparation phase where the QP problem (2.15) is formulated without knowing the initial condition, and a feedback phase where the QP problem is solved once the initial condition measurement is obtained. If the initial guess is sufficiently close to the exact solution, the tangential predictor obtained from RTI is able to provide a sufficiently good approximate solution (Diehl, Findeisen, Allgöwer, Bock, and Schlöder, 2005). An excellent implementation of RTI is provided by the ACADO Toolkit (Houska, Ferreau, and Diehl, 2011), which exploits symbolic modelling languages and exports optimized C-code for real-time applications.

2.4.1 Real-Time Iteration Scheme

Real-time Iteration (RTI) is today's one of the most promising fast NMPC algorithms. At the i^{th} sampling instant, RTI performs only one SQP iteration to solve the NLP problem (2.10) parameterized by multiple shooting. This corresponds to solve the QP problem (2.15) once. The first equality linearized constraint in (2.15b) is given by

$$\Delta s_0 = \hat{x}_0 - s_0. \tag{2.19}$$

This is called an initial value embedding strategy where the initial condition $\hat{x}_0$ enters the QP problem (2.15) linearly so that the constraint (2.9b) is fulfilled after one SQP iteration. Therefore, RTI scheme solves a sequence of similar QP problems on-line with varying initial conditions and can be seen as a special case of linear time-varying MPC strategy (Gros, Zanon,

Quirynen, Bemporad, and Diehl, 2016).

In RTI, the Hessian matrix $H(\mathbf{z}^i)$ is usually approximated by using Gauss-Newton method, which employs only the first derivative of the objective function and grants good numerical performance when the OCP (2.2) is a least square problem (Diehl et al., 2002) [2]. This approximation also excludes the Lagrangian multipliers from Hessian computation. As a result, the initialization $\mathbf{z}^i$ for each QP problem is updated using a full Newton step

$$\mathbf{z}^{i+1} = \mathbf{z}^i + \Delta\mathbf{z}^i, \tag{2.20}$$

where $\Delta\mathbf{z}^i$ is the minimizer of (2.15). Since the initialization is of critical importance for RTI scheme, different *shifting strategies* for improving the initialization are proposed (Diehl, 2001; Diehl et al., 2002). A *warm start* strategy directly takes the updated trajectory $\mathbf{z}^{i+1}$ to initialize the QP problem at the next sampling instant. This is often useful when the sampling frequency is high enough and the OCP does not change too much. If the sampling time does make an effect, the initialization can be updated by shifting $\mathbf{z}^{i+1}$ one sampling time forward using predicted states. The new initialization is given by

$$\mathbf{z}^{i+1} = [z_1^\top, z_2^\top, \ldots, z_{N-1}^\top, s_N^\top, u_{N-1}^\top, s_N^\top]^\top, \tag{2.21}$$

where z_0 is abandoned because it would have been outdated at the next sampling instant. As will be shown in Chapter 7, there are other shifting strategies such as state trajectory interpolation. All the strategies are motivated to provide a better initialization for the next QP problem. The warm start or shifting strategy are in fact the essential ideas of RTI, which try to exploit previous information as much as possible, including primal solutions as well as multipliers.

[2] In (Diehl et al., 2002), exact Hessian and several approximations are discussed, but Gauss-Newton approximation is widely adopted and is the only option for Hessian approximation in ACADO Code Generation Toolkit (Houska et al., 2011)

Decomposing a RTI step

Given an initialization trajectory $\mathbf{z}^i$, RTI firstly needs to build up the QP subproblem (2.15). Re-write (2.15) in a detailed form as

$$\begin{aligned}
\min_{\Delta\mathbf{s},\Delta\mathbf{u}} \quad & \sum_{k=0}^{N-1}\left(\frac{1}{2}\begin{bmatrix}\Delta s_k\\ \Delta u_k\end{bmatrix}^\top H_k^i \begin{bmatrix}\Delta s_k\\ \Delta u_k\end{bmatrix} + g_k^{i\top}\begin{bmatrix}\Delta s_k\\ \Delta u_k\end{bmatrix}\right) && \text{(2.22a)}\\
& +\frac{1}{2}\Delta s_N^\top H_N^i \Delta s_N + g_N^{i\top}\Delta s_N \\
s.t. \quad & \Delta s_0 = \hat{x}_0 - s_0, && \text{(2.22b)}\\
& \Delta s_k = A_{k-1}^i \Delta s_{k-1} + B_{k-1}^i \Delta u_{k-1} + d_{k-1}^i,\ k=1,\dots,N && \text{(2.22c)}\\
& C_k^i \begin{bmatrix}\Delta s_k\\ \Delta u_k\end{bmatrix} \le -c_k^i,\ k=0,1,\dots,N-1, && \text{(2.22d)}\\
& C_N^i \Delta s_N \le -c_N^i && \text{(2.22e)}
\end{aligned}$$

where $A_k^i = \frac{\partial \Xi}{\partial s}(s_k^i, u_k^i), B_k = \frac{\partial \Xi}{\partial u}(s_k^i, u_k^i)$ are called (integration) sensitivities w.r.t. initial states and controls, respectively. The gap between the end of the previous shooting trajectory and starting point of the next one (See Figure 2.2) is denoted by $d_{k-1}^i = \Xi(t_{k-1}, s_{k-1}^i, u_{k-1}^i; p) - s_k^i$. Here, H_k^i, C_k^i are the k^{th} blocks of the Hessian and constraint Jacobian matrices and g_k^i is the k^{th} sub-vector of $g(\mathbf{z}^i)$.

In the preparation phase of RTI, all QP data need to be computed for all shooting intervals. To evaluate g_k^i, C_k^i, we need to take first order derivatives of the objective and inequality constraints a, r. The integrator Ξ together with its sensitivity evaluations are performed by numerical integrating schemes, e.g. Runge-Kutta schemes. The exact Hessian requires the second order derivatives of the objectives and the integration. These derivatives can be computed by means of finite difference, or by a more modern technique called Automatic Differentiation (AD). Refer to (Griewank and Walther, 2008) for a comprehensive overview.

The QP problem (2.22) can be solved directly in a sparse form, since H, B, C in (2.15)

have the following banded block or block diagonal structures.

$$H^i = \begin{bmatrix} H_0^i & & & \\ & H_1^i & & \\ & & \ddots & \\ & & & H_N^i \end{bmatrix}, C^i = \begin{bmatrix} C_0^i & & & \\ & C_1^i & & \\ & & \ddots & \\ & & & C_N^i \end{bmatrix}, \tag{2.23}$$

$$B^i = \begin{bmatrix} -\mathbb{I} & & & & & & \\ A_0^i & B_0^i & -\mathbb{I} & & & & \\ & & A_1^i & B_1^i & -\mathbb{I} & & \\ & & & & \ddots & \ddots & \\ & & & & & A_{N-1}^i & B_{N-1}^i & -\mathbb{I} \end{bmatrix}, \tag{2.24}$$

where $\mathbb{I}$ stands for an identity matrix of appropriate dimension. QP solvers that exploit such a sparsity pattern have been developed, e.g. Forces Pro (Domahidi and Jerez, 2014) and qpDUNES (Frasch, Sager, and Diehl, 2015). An alternative is to *condense* the QP problem by removing the states $\Delta \mathbf{s}$ from optimization variables based on the equality constraints (2.22c). The resulting dense QP problem has much less decision variables with only inequality constraints and is suitable for small-scale QP solvers, but the sparsity has been lost. In principle, the condensing step has a computational complexity of order $\mathcal{O}(N^3)$ (Vukov, Domahidi, Ferreau, Morari, and Diehl, 2013), but an order of $\mathcal{O}(N^2)$ has been achieved by exploiting the banded block structure of QP matrices (Andersson, 2013). In the end, a full Newton step is taken and the initialization is shifted.

2.4.2 Multi-Level Scheme

In (Bock, Diehl, Kostina, and Schlo'der, 2007), a Multi-Level scheme is proposed for reducing the on-line computational cost of SQP based NMPC algorithms. The idea is to compute each component of the QP problem (2.15) in different levels at different sampling rates. We follow the definition in (Kirches, Wirsching, Sager, and Bock, 2010) for introducing different levels.

Level A: Feedback step

At this level, we only consider solving the QP problem (2.15) with fixed QP data. Given a reference initial trajectory $\bar{y} = [\bar{\mathbf{z}}^\top, \bar{\lambda}^\top, \bar{\mu}^\top]^\top$, we obtain Hessian matrix $\bar{H}$, constraint Jacobians $\bar{B}, \bar{C}$, objective gradient $\bar{g}$ and constraint residuals $\bar{b}, \bar{c}$. This is equivalent to a linear MPC in which all data are time-invariant. The difference is that, these data come from upper levels that may be performed more slowly than Level A.

Level B: Feasibility improvement

At this level, we update the constraint residuals $b(\mathbf{z}^i), c(\mathbf{z}^i)$ while keeping the Hessian H and Jacobian matrices B, C constants as in Level A. The objective gradient is updated as

$$g^i = \bar{g} + \bar{H}(\mathbf{z}^i - \bar{\mathbf{z}}). \tag{2.25}$$

It has been proved that such a SQP algorithm converges to a feasible (suboptimal) point of the NLP problem (2.10). The additional computational costs are evaluations of the mentioned functions. The computationally expensive matrix factorization is not necessary.

Level C: Optimality improvement

At this level, apart from updating constraint residuals as in Level B, we further compute a modified objective gradient by

$$g^i = \nabla_{\mathbf{z}} a(\mathbf{z}^i) + (B(\mathbf{z}^i) - \bar{B})^\top \lambda^i + (C(\mathbf{z}^i) - \bar{C})^\top \mu^i, \tag{2.26}$$

where $\bar{B}, \bar{C}$ are computed at the reference trajectory as in Level A and B. Assuming a SQP algorithm solving this sequence of gradient modified QPs converges to a limit $(\mathbf{z}^*, \lambda^*, \mu^*)$, then at the limit we have $(\Delta z^*, \lambda^*, \mu^*)$ the solution of the QP problem (2.15) with $\Delta z^* = 0$. The gradient of the Lagrangian of the QP is

$$\begin{aligned} & \nabla_{\Delta \mathbf{z}} \mathcal{L}_{QP} = H \Delta z^* + g^* + \bar{B}^\top \lambda^* + \bar{C}^\top \mu^* && (2.27) \\ = \; & \nabla_{\mathbf{z}} a(\mathbf{z}^*) + B^\top(\mathbf{z}^*) \lambda^* + C^\top(\mathbf{z}^*) \mu^* && (2.28) \\ = \; & \nabla_{\mathbf{z}} \mathcal{L}_{NLP} && (2.29) \\ = \; & 0 && (2.30) \end{aligned}$$

Hence, starting from a fixed x^0, this SQP algorithm converges to the exact optimizer of the original NLP problem (2.10).

Comparing to Level B, the additional computational cost is the evaluation the modified gradient (2.26), which can be efficiently performed by computing the adjoint sensitivities $\bar{B}^\top \lambda, \bar{C}^\top \mu$ (Kirches et al., 2010, 2012). It has been shown that the cost for computing adjoint sensitivities is no more than five times the cost of the corresponding function evaluation (Griewank and Walther, 2008). The matrix factorization is neither necessary since the Hessian and constraint Jacobians are constants.

Level D: Full step

At this level, the first and second order derivatives H, B, C, which are the most computationally expensive QP data, are updated at the current trajectory. Note that the multi-level scheme does not necessarily require to employ all levels of computations and it is in nature more suitable for parallel computations (Kirches et al., 2010). A combination of Level A and D using RTI is employed in (Albersmeyer, Beigel, Kirches, Wirsching, Bock, and Schlöder, 2009) for an application of automotive control. A parallel implementation of multi-level schemes in the SQP framework has been developed in (Lindscheid, Haßkerl, Meyer, Potschka, Bock, and Engell, 2016).

The Multi-Level schemes can be summarized in Algorithm 2.1. [3]

Algorithm 2.1 Multi-Level inexact sensitivity RTI scheme

1: Initialize at $(\mathbf{z}^0, \lambda^0, \mu^0)$. Choose a sensitivity update interval $N_m \in \mathbb{N}^+$.
2: **for** $i = 0, 1, \ldots$ **do**
3: Compute $H^i, g^i, b^i, B^i, c^i, C^i$,
4: **if** $i \mod N_m = 0$ **then**
5: Update the constraint Jacobian B^i
6: Set $\bar{B} \leftarrow B^i$
7: **end if**
8: Solve (2.15) with equality constraint Jacobian $\bar{B}$
9: Update the initialization for the next sampling instant by (2.16) with a full step $\alpha^i = 1$.
10: **end for**

2.4.3 Adjoint Sensitivity Scheme

The adjoint sensitivity scheme refers to a special class of Multi-level Scheme which contains only the Level C. It is motivated by the observation that the constraint Jacobians, especially the one of the dynamic constraint (2.22c), are computationally extensive even for state-of-the-art numerical integrators (Serban and Hindmarsh, 2005; Kühl, Ferreau, Albersmeyer, Kirches, Wirsching, Sager, Potschka, Schulz, Diehl, Leineweber, et al., 2007; Quirynen et al., 2015). The computational efforts increase when multiple shooting parameterization is employed, since the dynamic sensitivities are supposed to be evaluated for every shooting interval. As a consequence, the adjoint sensitivity scheme adopts the constraint Jacobians $\bar{B}, \bar{C}$ at the the reference trajectory $\bar{\mathbf{z}}$ which is selected off-line and kept constant throughout the on-line computations.

[3] All QP components in (2.15) can be evaluated at different rates but we consider the constraint Jacobian matrix here only for later comparison.

Firstly proposed in (Wirsching, Bock, and Diehl, 2006; Wirsching, Albersmeyer, Kühl, Diehl, and Bock, 2008), the adjoint sensitivity scheme is shown to have significantly reduced the on-line computational cost in a RTI framework. If the condensing step is adopted, the on-line computational cost can be further reduced by using matrix-vector condensing as described in (Kirches et al., 2012). In the authors point of view, the biggest benefit brought by using adjoint sensitivity scheme is such cheap condensing computation rather than fixing the sensitivities.

As mentioned in Chapter 2.4.2, given a fixed initial guess y^0, a SQP algorithm using the adjoint sensitivity scheme converges to the exact KKT point of the NLP (2.10) (Bock et al., 2007). An extension of this result is the feasibility and stability study of the adjoint sensitivity scheme (Zanelli, Quirynen, and Diehl, 2016).

2.4.4 Mixed-Level Iteration Schemes

The idea of Mixed-Level Iteration Schemes (Frasch, Wirsching, Sager, and Bock, 2012) is to update the QP components not only based on different sampling rates, but also over the prediction horizon for different stages (or shooting intervals). As a consequence, the multi-stage nature of the QP problem (2.22) is exploited. Mixed-Level Iteration Schemes consist two levels of iterations.

Fractional-level iterations

In this iteration strategy, only the QP components in the first $N_{frac} < N$ shooting intervals are updated at every sampling instant. This is motivated by the observation that to model the system better in earlier parts of the prediction horizon is more important than in later parts.

Mixed-level iterations

In this iteration strategy, the fractional-level iterations are mixed with the multi-level iteration scheme. For example, a D/B level iteration updates all the QP components in the first $N_{frac} < N$ shooting intervals and updates the computationally cheap components, including constraint residuals and gradient computations on the other intervals. If $N_{frac} \ll N$, significant savings in computing derivatives can be expected. In addition, the fractional-level iterations can be performed every $N_m > 1$ sampling instants, also reducing the computational burden.

A tailored condensing algorithm for the QP problem (2.15) has been proposed in (Frasch et al., 2012) for Mixed-Level iteration schemes that further reduces the computational effort for solving the NLP problem (2.10) on-line. It has been proved that the lower right

$(N - N_{frac}) \times (N - N_{frac})$ blocks of the condensed Hessian matrix are constant. This leads to a computational complexity reduction for condensing from $\mathcal{O}(N^3)$ to $\mathcal{O}(NN_{frac} + N_{frac}^3)$.

2.4.5 Remarks

There remain some open issues when applying Multi-Level schemes, adjoint sensitivity scheme and Mixed-Level schemes in the RTI framework:

- It is not trivial to choose an appropriate sensitivity update interval N_m, or a pre-defined trajectory $\bar{\mathbf{z}}$, such that the inexact Jacobian $\bar{B}$ is a good approximation of the exact one B^i for any i.
- In Algorithm 2.1, the integration sensitivity in each shooting interval over the prediction horizon, either updated or not, are treated as a whole. The block structure of the Jacobian matrix B is not exploited.
- In Mixed-Level schemes, updating only the first N_{frac} blocks in the Jacobian matrix is often not adequate to control highly nonlinear and fast changing systems. In addition, it is not trivial to choose an appropriate N_{frac} for all possible operating conditions.

3

Partial Sensitivity Update

The computational burden of NMPC using on-line optimization has long been a difficulty since NMPC is developed. While the optimization problems arising from linear MPC can be solved reliably and in a extremely fast speed, the optimization problems in NMPC are often hard to solve and cannot be solved within the real-time restriction. This is due to the difficulty of reliably solving an open-loop nonlinear optimal control problem at each sampling instant, respecting state, control and other physical and safety constraints. In Chapter 2, state-of-the-art fast NMPC algorithms have been reviewed. However, there is more to do to improve the computational efficiency.

The solution we provide is to build a bridge between linear and nonlinear MPC. Real-Time Iteration (RTI) (Diehl et al., 2002) has built such a bridge by approximately solving the NLP problem (2.10) on-line (Gros et al., 2016). RTI performs only one Newton step of SQP hence only one QP problem is solved at each sampling instant, which is very similar to the case of linear MPC. In this chapter, we go further than RTI by exploiting the multi-stage feature of the NLP problem as in (2.22) and employing a partial linearization of the nonlinear dynamics within prediction window. As a result, a mixed linear and nonlinear MPC is formulated and computational cost reduction can be expected.

A question immediately arises that at what time linear MPC is appropriate and linearization is necessary. As can be seen from the example (1.1), nonlinearities come from either dynamics, constraints or operating conditions and not all nonlinear systems require nonlinear controllers

(Nikolaou and Misra, 2003). To answer this question, recognizing a system as being linear or nonlinear is not sufficient. While this recognition is not difficult to obtain by applying the definition of linearity to system's model or output response, quantifying the degree of nonlinearity (DoN) is not that easy. In the following section, the idea of Measure of Nonlinearity (MoN) is reviewed to perform such a quantification.

3.1 Measure of Nonlinearity

The research of MoN can trace back to 1980s (Desoer and Wang, 1980). Early studies focus on quantifying the input-output (IO) nonlinearity of a system, without the consideration of feedback control law. These belong to the class of open-loop MoNs. However, open-loop MoNs may be biased because of the interaction between nonlinearity and feedback (Nikolaou and Misra, 2003; Schweickhardt and Allgower, 2004). For example, the nonlinearity of a system can be decreased by introducing feedback (Desoer and Wang, 1980; Schweickhardt and Allgower, 2004). Hence, there exist a class of closed-loop MoNs that explicitly take feedback into account.

3.1.1 Open-loop MoN

Distance between a nonlinear and a linear system

The fundamental idea of measure of nonlinearity (MoN) is to compare the nonlinear system at hand to a (set of) linear system. Consider a linear and a nonlinear transfer operators denoted by L and N

$$L[u] : u \to y_L, \forall u \in \mathcal{U}, y \in \mathcal{Y}, \tag{3.1}$$

$$N[u] : u \to y_N, \forall u \in \mathcal{U}, y \in \mathcal{Y}, \tag{3.2}$$

where u is the input signal and y_L, y_N are outputs from the linear and nonlinear operator, respectively. The very first definition of MoN is probably proposed in (Desoer and Wang, 1980) as

$$\epsilon_1 := \inf_{L \in \mathcal{L}} \sup_{u \in \mathcal{U}} \|N[u] - L[u]\|, \tag{3.3}$$

where $\mathcal{L}$ is a set of linear systems. The definition (3.3) is intuitive and has the following properties

1. ϵ_1 measures the DoN in the worst case, i.g. with the worst possible input signal u.
2. The best linear approximation L is found among a set of linear models.

3. ϵ_1 depends on the input set $\mathcal{U}$ and the linear model set $\mathcal{L}$, and is scale sensitive to the input and output.

A scale-free definition is given in (Schweickhardt and Allgower, 2004) to overcome the shortcoming of (3.3). The resulting MoN, named relative MoN, is defined by

$$\epsilon_2 := \inf_{L\in\mathcal{L}} \sup_{u\in\mathcal{U}} \frac{\|N[u]-L[u]\|}{\|N[u]\|}. \tag{3.4}$$

The most important property of ϵ_2 is that its value is in the range of $[0,1]$ since it is normalized by the nonlinear output. The interpretation of ϵ_2 is intuitive: when ϵ_2 is close to zero, the nonlinearity of the operator N is small; ϵ_2 close to one corresponds to a more severe nonlinearity.

(Helbig, Marquardt, and Allgöwer, 2000) generalize the definition (3.4) to state-space models and take the initial condition into account. Consider a nonlinear system given by

$$\dot{x}_N(t) = f(x_N(t),u(t)),\ x_N(0) = x_{N,0}, \tag{3.5}$$
$$y_N(t) = h(x_N(t),u(t)), \tag{3.6}$$

and a linear system

$$\dot{x}_L(t) = Ax_L(t) + Bu(t),\ x_L(0) = x_{L,0}, \tag{3.7}$$
$$y_L(t) = Cx_L(t) + Du(t). \tag{3.8}$$

The MoN is defined as

$$\epsilon_3(t_f) := \inf_{L\in\mathcal{L}} \sup_{(u,x_{N,0})\in\mathcal{S}} \inf_{x_{L,0}\in\mathcal{X}_{L,0}} \frac{\|N[u,x_{N,0}]-L[u,x_{L,0}]\|}{\|N[u,x_{N,0}]\|}, \tag{3.9}$$

where

$$\mathcal{S} = \{(u,x_{N,0}) : u\in\mathcal{U}, x_{N,0}\in\mathcal{X}_{N,0}, N[u,x_{N,0}]\in\mathcal{Y}\}, \tag{3.10}$$

and all signals are defined over a time interval $[0,t_f)$. The meaning of ϵ_3 is the difference between the best linear approximation L over a linear system set $\mathcal{L}$, and the nonlinear system N, considering the worst-case combination of input u and initial condition $x_{N,0}$, and the best-case initial condition $x_{L,0}$. As a result, definition (3.9) is not restricted to the MoN of a nonlinear system at the neighborhood of steady state as (3.4). The most important property of such a definition is that ϵ_3 can be employed for transient processes by optimally adjusting the initial condition $x_{N,0}$.

(Li, 2012) further extend the idea of (3.4) to stochastic systems. Instead of computing

the difference between a nonlinear system and its best linear approximation, the authors manage to measure the deviation of the nonlinear system from all linear functions. This idea is motivated from the most popular definition of distance between a point (the nonlinear system at hand) and a subspace (the set of all linear functions), in the functional space. Consider a discrete-time nonlinear stochastic system

$$x_{k+1} = f_k(x_k) + u_k + w_k, \tag{3.11}$$

$$y_k = h_k(x_k) + v_k, \tag{3.12}$$

where x_k is the (random) state, u_k a deterministic and known control input, w_k, v_k are zero-mean white process noise and measurement noise. The nonlinear system can be written in compact by

$$g_k(x_k) = [f_k(x_k)^\top, h_k(x_k)^\top]^\top. \tag{3.13}$$

The MoN is defined as

$$\epsilon_4 := \frac{\inf_{L_k \in \mathcal{L}} (\mathrm{E}[\|L_k(x) - g_k(x)\|_2^2])^{1/2}}{[\mathrm{tr}(C_{g_k})]^{1/2}}, \tag{3.14}$$

where E is the expectation operator, $\mathcal{L}$ is the set of all linear functions in the form of $L(x) = Ax + b$ and C_{g_k} is the covariance matrix of $g_k(x)$. In contrast to definition (3.4) and (3.9), ϵ_4 is a global measure rather than one evaluated at some point. Moreover, ϵ_4 is relatively neutral (i.e. not pessimistic (Li, 2012)) since it does not consider only the worst-case situation.

Gap metric

The distance between a nonlinear system and a linear system in definitions $\epsilon_1 - \epsilon_4$ are based on norms. However, they are not valid for unstable systems, since the distance between unstable systems cannot be measured in terms of norms (Tan, Marquez, Chen, and Liu, 2005). To overcome this limitation, (El-Sakkary, 1985) propose the gap metric to measure the distance of two linear but not necessarily stable systems. Recently, (Du, Song, and Li, 2009) propose a nonlinearity measure based on gap metric between two linear systems linearized at two operating points of a nonlinear system.

Consider two linear systems denoted by transfer matrices P_1, P_2 with the same number of inputs and outputs. They can be factorized by the normalized right coprime factorization as

$$P_i = N_i M_i^{-1}, \quad i = 1, 2, \tag{3.15}$$

where $\tilde{M}_i M_i + \tilde{N}_i N_i = I$ and $\tilde{M}$ is the complex conjugate of M. The gap between P_1, P_2 is

defined by

$$\xi(P_1, P_2) := \max \left\{ \begin{array}{l} \inf_{Q \in H_\infty} \left\| \begin{bmatrix} M_1 \\ N_1 \end{bmatrix} - \begin{bmatrix} M_2 \\ N_2 \end{bmatrix} Q \right\|_\infty \\ \\ \inf_{Q \in H_\infty} \left\| \begin{bmatrix} M_2 \\ N_2 \end{bmatrix} - \begin{bmatrix} M_1 \\ N_1 \end{bmatrix} Q \right\|_\infty \end{array} \right\}. \tag{3.16}$$

The metric defined by (3.16) is bounded between 0 and 1. A gap ξ close to 0 indicates two close linear systems, while ξ close to 1 means distant linear systems. A more important property of gap metric is that at the presence of a small gap ξ, there exists at least one feedback controller that stabilizes both linear systems and the distance between the two closed-loop systems is small in the ∞-norm sense (Du et al., 2009). We can exploit this property to design linear controllers.

Strictly speaking, definition (3.16) is not a nonlinearity measure since it evaluates the distance between linear systems. (Du et al., 2009) propose a MoN based on the gap (3.16), given as

$$\epsilon_5 := \sup_{p_i, p_j \in \Omega} \xi(L(p_i), L(p_j)), \tag{3.17}$$

where p_i, p_j are operating points in the operating space Ω of a nonlinear system N. $L(p)$ is a linear approximation of N linearized at point p. This definition avoids the selection of a linear system set as in ϵ_1-ϵ_4. In addition, ϵ_5 is a global measure that evaluates the MoN in the worst-case over the entire operating space. This reveals a fundamental feature of "nonlinearity" that the degree of nonlinearity of a system is closely related to its operating point, which has been illustrated by example (1.1).

Relative curvature measure

Although definitions of MoNs introduced in previous sections are intuitive, the corresponding cost for evaluating global MoNs is not affordable for most applications. Alternatively, (Bates and Watts, 1980) develop a simple relative local measure of nonlinearity based on curvature which is a concept in differential geometry. The authors apply the so-called curvature MoN to a model-experimental design-parameterization combination and find that all the nonlinear data investigated has a low intrinsic MoN, which is inherent for the model, and a large parametric MoN, which is determined by parameterization of the model.

Consider a least-square estimation problem with experimental settings $x_t (t = 1, 2, \ldots, n)$ and responses $y_t (t = 1, 2, \ldots, n)$. The relationship between the settings and the response is given by

$$y_t = \eta(x_t, \theta) + \varepsilon_t \tag{3.18}$$

where $\theta = [\theta_1, \theta_2, \ldots, \theta_p]^\top$ is a set of unknown parameters and ε_t is an additive random measurement error of zero-mean. We make exact distinctions between *model* and *model function*. The model is the relationship between the response and the experimental settings and is independent of the parameterization. The model function is related to the particular parameterization of the model. In this sense, (3.18) refers to the model function which depends on the parameter θ.

The linear approximation of the model function (3.18) around a point θ_0 is given by

$$\eta(\theta) = \eta(\theta_0) + \sum_{i=1}^{p} (\theta_i - \theta_{i,0}) v_i \tag{3.19}$$

where $v_i = [v_i(x_1), v_i(x_2), \ldots, v_i(x_n)]^\top = \frac{\partial \eta}{\partial \theta_i}(\theta_0)$ and $v_i(x) = \frac{\partial \eta}{\partial \theta_i}(\theta_0)$. In a geometric point of view, a straight line in the parameter space through θ_0 is given by

$$\theta(b) = \theta_0 + b\mathbf{h}, \tag{3.20}$$

where $\mathbf{h} = [h_1, \ldots, h_p]^\top$ is any non-zero vector. The corresponding curve on the solution locus is

$$\eta_h(b) = \eta(\theta_0 + b\mathbf{h}). \tag{3.21}$$

The *velocity* and *acceleration* vectors are defined by

$$\dot{\eta}_h = \frac{\mathrm{d}\eta_h}{\mathrm{d}b}(0) = V_{\cdot}\mathbf{h}, \tag{3.22}$$

$$\ddot{\eta}_h = \frac{\mathrm{d}^2\eta_h}{\mathrm{d}^2 b}(0) = \mathbf{h}^\top V_{\cdot\cdot}\mathbf{h}. \tag{3.23}$$

By writing

$$\ddot{\eta}_h = \ddot{\eta}_h^N + \ddot{\eta}_h^P + \ddot{\eta}_h^G, \tag{3.24}$$

the acceleration vector is decomposed by three components, with $\ddot{\eta}_h^N$ normal to the tangent plane, $\ddot{\eta}_h^P$ parallel to $\dot{\eta}_h$ and $\ddot{\eta}_h^G$ parallel to the tangent plane normal to $\dot{\eta}_h$.

The local curvature is defined as

$$K_h = \frac{\|\ddot{\eta}_h\|}{\|\dot{\eta}_h\|^2}, \tag{3.25}$$

in the direction $\mathbf{h}$. A straightforward classification is that an intrinsic curvature K_h^N and a parametric curvature K_h^T. The former adopts the normal acceleration component $\ddot{\eta}_h^N$ for its numerator and the latter adopts the parallel components $\ddot{\eta}_h^P + \ddot{\eta}_h^G$ for its numerator.

Given a confidence region of interest, the curvature MoN (3.25) is able to reflect the

degree of nonlinearity locally. Then it is meaningful to judge a linear confidence region to be valid or not. More importantly, the use of intrinsic and parametric curvature helps with the parameter design. For example, for model functions having a low intrinsic curvature but a high parametric curvature, re-parameterization can be used to improve the accuracy of the linear approximation of the confidence region. Applications of curvature MoN to nonlinear estimation problems can be found in (Mallick and La Scala, 2006; La Scala, Mallick, and Arulampalam, 2007; Niu, Varshney, Alford, Bubalo, Jones, and Scalzo, 2008).

The idea of curvature MoN is extended to dynamic control system by (Guay, McLellan, and Bacon, 1995), who evaluate the local MoN for input-output pairs around the steady state. Consider an asymptotically stable nonlinear system

$$\dot{x} = f(x,u),\ x \in \mathcal{X}, u \in \mathcal{U}, \tag{3.26}$$

$$y = h(x). \tag{3.27}$$

The linear approximation at a stationary point (x_0, u_0) is given by

$$\dot{x} = A(x - x_0) + B(u - u_0), \tag{3.28}$$

where $A = \frac{\partial f}{\partial x}(x_0,u_0), B = \frac{\partial f}{\partial u}(x_0,u_0)$ are assumed to be controllable and rank$(B) = P_B$. A P_B-dimensional sub-manifold is defined in the state space by $f(x,u) = 0$. As a result, there exists a nonlinear map $\Psi : \mathcal{U} \to \mathcal{X}$ in a neighborhood of $u_0 \in \mathcal{U}$, which locally defines a P_B-dimensional manifold in $\mathbb{R}^{n_x}$ denoted by $\Psi(\mathcal{U})$. This surface is referred to *steady-state locus*.

Compute the P_B-dimensional tangent space of $\Psi(u)$ at the point u_0 by

$$\dot{v}_i = \frac{\partial \Psi}{\partial u_i},\ 1 \le i \le P_B. \tag{3.29}$$

where $\dot{v}_i$ refer to velocity vectors, and the acceleration vector by

$$\ddot{v}_{pq} = \frac{\partial^2 \Psi}{\partial u_p \partial u_q}. \tag{3.30}$$

Collect the velocity and acceleration vectors into matrices as

$$V = [\dot{v}_1, \ldots, \dot{v}_p]^\top, \tag{3.31}$$

$$W = [\ddot{v}_{1,1}, \ldots, \ddot{v}_{1,p}, \ldots, \ddot{v}_{p,p}]^\top. \tag{3.32}$$

Applying QR factorization to the combined matrix $[V, W]$ (Bates and Watts, 1980) gives

$$[V, W] = QR = [Q_1^t, Q_1^n, Q_2] \begin{bmatrix} R_1 & A^t \\ & A^n \\ & \end{bmatrix} \tag{3.33}$$

where Q is an N by N orthonormal matrix and R is an N by $P_B(P_B+3)/2$ matrix. The first P_1^t columns of Q, denoted by Q_1^t, span the tangent space of the steady state locus and the next P_1^n columns, denoted by Q_1^n, span a P_1^n-dimensional subspace of the space normal to Q_1^t. The remaining sub-matrix, denoted by Q_2, is an orthonormal basis of P_2 column vectors. This factorization is similar to the curvature MoN (3.25) with three components. The curvature is defined by

$$K_c := \frac{\|e^\top A_r e\|}{\|R_1 e\|^2}, \tag{3.34}$$

where A_r is an N-dimensional array of P by P matrices (tensor) obtained from A_t, A_n, and e is a direction vector in the input space. Note that definition (3.34) measures the instant curvature or nonlinearity which is scale-free on e, since e square occur in both numerator and denominator.

3.1.2 Closed-loop MoN

In Chapter 3.1.1, we introduce a number of definitions of the measure of nonlinearity. However, they share a common feature that these measures are evaluated for input and output pairs and do not consider the role of feedback control. (Nikolaou and Misra, 2003) once pointed out that "the proximity of a nonlinear system to a linear one is neither necessary nor sufficient for good closed-loop performance". As a consequence, knowing how nonlinear a system is does not suffice. There should be measures to evaluate the degree of nonlinearity of a system interacting with feedback controllers.

To obtain a general measure of closed-loop MoN is not easy, since there are a number of feedback structures and laws in the control theory. Hence, most of the closed-loop MoNs are defined in a specific control structure. (Nikolaou and Misra, 2003) consider the closed-loop MoN in the framework of internal model control (IMC) (Garcia and Morari, 1982), illustrated by Figure 3.1. The Youla control operator Q may not have an analytic form and is defined implicitly, e.g. via on-line optimization as in the case of MPC. Define an operator

$$\Delta N := NQ(I + NQ - LQ)^{-1} - LQ, \tag{3.35}$$

as the difference between the nonlinear and linear closed loop. A lower and an upper bound

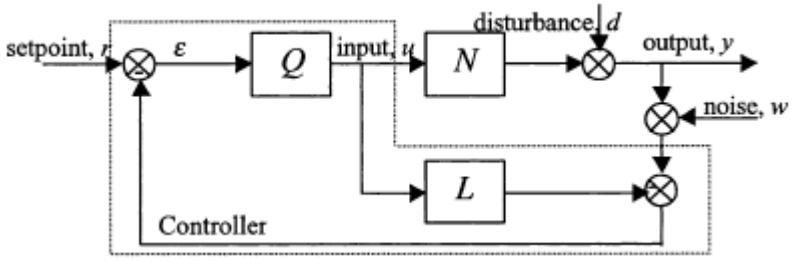


Figure 3.1: Block diagram of IMC for a nonlinear process N and its (linear or nonlinear) model L. Q is the Youla parameter of the controller.

is developed by (Eker and Nikolaou, 2002) that is

$$\frac{\|W(I-LQ)(N-L)Q\|_{\Delta E}}{1+\|(N-L)Q\|_{\Delta E}} \leq \epsilon_6 \leq \|W(I-LQ)\|\frac{\|(N-L)Q\|_{\Delta E}}{1-\|(N-L)Q\|_{\Delta E}}, \tag{3.36}$$

where $\epsilon_6 := \|W\Delta N\|_{\Delta Z}$ is the closed-loop MoN, set E contains signals ϵ and set $Z = [I + (N-L)Q](E)$. There are two interpretations of this result:

1. The closed-loop MoN depends on the open-loop MoN $N-L$ and the controller Q.

2. An appropriate controller design may lead to a virtually linear closed-loop performance for nonlinear processes.

Indeed, (Nikolaou and Misra, 2003) conclude that (3.36) are valid if $\gamma := \|(N-L)Q\|_{\Delta E} < 1$, and if Q is designed such that $\gamma \ll 1$ and $\frac{\gamma}{1-\gamma} \ll 1$, then the closed loop in Figure 3.1 is virtually linear.

(Schweickhardt and Allgower, 2004) study the closed-loop MoN in an optimal control framework. Consider an optimal control problem

$$\min_{u} \quad J(x_0)[u] = \int_{t_0}^{T} F(x,u)\mathrm{d}t \tag{3.37a}$$

$$s.t. \quad \dot{x} = f(x,u),\ x(t_0) = x_0,\ x_0 \in \mathcal{X} \tag{3.37b}$$

For $T \to \infty$, since the absence of state constraints, the solution of the infinite horizon optimal control problem (3.37) can be formulated as a static state feedback control law $u = k(x)$. As a result, the OCP (3.37) depends only on the initial condition x_0. The MoN of the optimal control law (OCL) is defined as

$$\epsilon_7 = \inf_{K\in\mathbb{R}^{n_x\times n_u}} \sup_{x_0\in\mathcal{X}} \frac{\|N_{OCL}[x^*] - Kx^*\|}{\|N_{OCL}[x^*]\|}, \tag{3.38}$$

where $N_{OCL}[x^*] = u^* = k(x^*)$ with x^* the solution of (3.37). In this definition, the nonlinear operator $N_{OCL}[x^*]$ is compared to the static linear system $u = k(x)$. The resulting measure ϵ_7 considers only the the nonlinearity on the optimal trajectory of the closed-loop system. Moreover, ϵ_7 can be evaluated without the knowledge of the feedback law k, which is beneficial for system analysis before controller design.

(Guay, Dier, Hahn, and McLellan, 2005) propose a local MoN as the quantification of the sensitivity of controller performance for linear controllers operating on a nonlinear plant. The authors focus on linear quadratic regulators and evaluate the deviation from optimality for controlling nonlinear systems using linear controllers. Given a quadratic regulator

$$\min_{u} \quad \int_0^\infty x^\top(t)Qx(t) + u^\top(t)Ru(t)\mathrm{d}t \tag{3.39a}$$

$$s.t. \quad \dot{x} = f(x, u(t)), \tag{3.39b}$$

$$y = h(x(t)), \tag{3.39c}$$

where system state x is time-invariant. Linearizing the nonlinear system about the origin we obtain a linear time-invariant system

$$\dot{x} = Ax + Bu(t), \tag{3.40}$$

$$y = Cx, \tag{3.41}$$

where $A = \frac{\partial f}{\partial x}(0,0), B = \frac{\partial f}{\partial u}(0,0), C = \frac{\partial h}{\partial x}(0)$. The optimal regulation law is given by

$$u(t) = -R^{-1}BPx(t) = Kx(t), \tag{3.42}$$

where K is the gain matrix and P is the solution of the following Riccati equation

$$A^\top P + PA - PBR^{-1}B^\top P + C^\top QC = 0. \tag{3.43}$$

The optimal cost can be approximated by

$$J^* = x^\top(t)Px(t). \tag{3.44}$$

The measure of nonlinearity in this context, refers to the performance sensitivity measure (PSM). Assuming that there is a perturbation $v(t)$ to the control law

$$u^*(t) = -R^{-1}B^\top P\hat{x}(t) + v(t), \tag{3.45}$$

where $\hat{x}(t)$ is the current state estimate. The degree of nonlinearity can be assessed by

computing the third order derivative of the objective functional, as

$$\bar{\epsilon}_8 = \nabla_v^3 J^*. \tag{3.46}$$

For a detailed definition and computation of such a measure refer to (Guay et al., 2005). This measure is a local one around the steady state point and requires only the knowledge of the LQR design cost parameters and the operating region of interest.

3.2 Curvature-like MoN for NMPC

3.2.1 MoN in Dynamic Optimization

Due to extensive computational efforts, the global MoNs such as the distance measure and the gap metric, are not suitable for on-line control algorithms. Indeed, the best linear model in a global sense may deteriorate local performance and a highly nonlinear system (probably of large dimensions) may require to identify a prohibitive number of local linear models using gap metric. Since system nonlinearity depends not on dynamics, but also on local operating conditions (as observed by example (1.1)), a local MoN suffices to assess the nonlinearity of a system and provide guiding information for controllers. The measure should be adaptive to operating conditions so that the latest MoN is always obtained.

Based on the analysis given above, the curvature MoN is selected to fulfill our purposes. However, the computation of curvature (3.34) in Chapter 3.1.1 requires the knowledge of the acceleration vector, which is the second order derivative of a nonlinear system. In addition, computational complexities increases when the dynamics are characterized by a numerical integration operator Ξ. There are two issues associated with this computation:

1. The second order derivative of a numerical integration operator is usually very difficult to compute. The computational burden introduced by this derivative may far exceed the reduced amount by using MoNs.

2. The curvature measure cannot completely reflect the nonlinearity of a system due to lack of higher order terms (Li, 2012). A more accurate measure for highly nonlinear system is needed.

To overcome these two drawbacks, we propose a novel curvature-like MoN that can be applied to real-time NMPC applications. The main advantages are

1. only the first order derivatives from past sampling instants are needed to compute such a curvature-like MoN but it still contains higher order terms, and

2. local sensitivity change between two sampling instants are measured, and

3. the novel MoN depends not only on system dynamics, but also on the magnitude and direction of inputs, which implicitly takes feedback into consideration.

Therefore, an accurate local MoN can be obtained with negligible computational efforts and is an ideal tool for implementing on-line NMPC controllers.

3.2.2 Curvature-like MoN for Numerical Integration Operator

Consider a nonlinear, at least C^2 continuous and differentiable vector function $\mathcal{Z}(x)$, where $x \in \mathbb{R}^n, \mathcal{Z} \in \mathbb{R}^m$. The Taylor expansion of $\mathcal{Z}$ at a point x_0 with an increment p reads

$$\mathcal{Z}(x_0+p) = \mathcal{Z}(x_0) + \frac{\partial \mathcal{Z}}{\partial x}(x_0)p + \frac{1}{2!}p^\top \frac{\partial^2 \mathcal{Z}}{\partial x^2}p^\top + \mathcal{O}(\|p^3\|), \tag{3.47}$$

where $x_0, p \in \mathbb{R}^n$, $\frac{\partial z}{\partial x}$ is the first order m by n Jacobian matrix and $\frac{\partial^2 z}{\partial x^2}$ is the second order tensor, which is a vector of m by n matrices of length n. The product $p^\top \frac{\partial^2 z}{\partial x^2} p^\top$ is performed by computing the vector-matrix-vector multiplication for each matrix in the tensor. Refer to (Bates and Watts, 1980) for more mathematical details. In the sequel, we use $\frac{\partial^2 z}{\partial x^2}p^2$ for this product using the scalar form for notation simplicity.

The original curvature MoN is defined as

$$\kappa_0 := \frac{\|\frac{\partial^2 \mathcal{Z}}{\partial x^2}p^2\|}{\|\frac{\partial \mathcal{Z}}{\partial x}(x_0)p\|^2} \tag{3.48}$$

which is the ratio of the norm of the second order term over the square norm of the first order one. Note that definition (3.48) eliminates the scaling effect of the directional input p as both numerator and denominator contain p^2. As a consequence, (3.48) evaluates the instantaneous MoN at point x_0 and that is the source of the name curvature. However, we argue that measuring the instantaneous degree of nonlinearity in sampled digital systems can barely provide any information about the nonlinearity of the system. In fact, the nonlinearity (or the change of linearization) from one sample to the next is more useful, since our computer cannot know what happened within two samples. Assume that a very small MoN of a system is evaluated by definition (3.48) at $x(t_0)$ along the direction q that moves the system towards $x(t_1)$. This means that the instantaneous nonlinearity is very low. After a (not necessarily long) sample time $t_1 - t_0$, however, the linear representation of the system at $x(t_1)$ may be considerably different from it at $x(t_0)$.

To consider the system evolution from one sample to the next, a novel curvature-like

MoN is proposed as

$$\kappa_1 := \frac{\|\frac{\partial^2 \mathcal{Z}}{\partial x^2} p^2\|}{\|\frac{\partial \mathcal{Z}}{\partial x}(x_0)p\|} \tag{3.49}$$

by removing the square computation in the denominator. Given that the function $\mathcal{Z}$ is at least twice differentiable, the Jacobian matrix can be expanded as

$$\frac{\partial \mathcal{Z}}{\partial x}(x_0+p) = \frac{\partial \mathcal{Z}}{\partial x}(x_0) + \frac{\partial^2 \mathcal{Z}}{\partial x^2} p + \frac{1}{2}\frac{\partial^3 \mathcal{Z}}{\partial x^3} p^2 + \mathcal{O}(\|p\|^3). \tag{3.50}$$

Hence, we have

$$\begin{aligned} [\frac{\partial \mathcal{Z}}{\partial x}(x_0+p) - \frac{\partial \mathcal{Z}}{\partial x}(x_0)]p &= \frac{\partial^2 \mathcal{Z}}{\partial x^2} p^2 + \frac{1}{2}\frac{\partial^3 \mathcal{Z}}{\partial x^3} p^3 + \mathcal{O}(\|p\|^4) && (3.51) \\ &\approx \frac{\partial^2 \mathcal{Z}}{\partial x^2} p^2 + \mathcal{O}(\|p\|^3) && (3.52) \end{aligned}$$

As a result, the definition of κ_1 can be re-written as

$$\kappa_2 = \frac{\|[\frac{\partial \mathcal{Z}}{\partial x}(x_0+p) - \frac{\partial \mathcal{Z}}{\partial x}(x_0)]p\|}{\|\frac{\partial \mathcal{Z}}{\partial x}(x_0)p\|} \tag{3.53}$$

that evaluates the relative change of the directional derivative with an approximation error of order $\mathcal{O}(\|p\|^3)$.

A more appropriate definition can be used by considering also the higher order terms. Define the MoN as

$$\kappa_3 = \frac{\|\mathcal{Z}(x_0+p) - \mathcal{Z}(x_0) - \frac{\partial \mathcal{Z}}{\partial x}(x_0)p\|}{\|\frac{\partial \mathcal{Z}}{\partial x}(x_0)p\|} \tag{3.54}$$

where the numerator equals to $\frac{1}{2}\|\frac{\partial^2 \mathcal{Z}}{\partial x^2} p^2\| + \mathcal{O}(\|p\|^3)$. Here, we compute the ratio of all the terms of order higher than one over the first order term. This is a precise definition of nonlinearity. The advantages of using definition (3.54) over (3.48) and the others are summarized in the following.

1. Higher order terms are considered.

2. No prior knowledge of the derivatives at the next point $x_0 + p$ is required.

3. The computation of (3.54) exploits only the first order derivative of the function $\mathcal{Z}$.

Replace the function $\mathcal{Z}$ by a functional, or a numerical integration operator Ξ operating on a dynamic system $\dot{x}(t) = f(x), x(0) = x_0$ and the directional vector p by the initial condition perturbation $p = \tilde{x}_0 - x_0$. The CMoN defined by (3.54) evaluates the sensitivity of

the integration operator w.r.t. to initial condition perturbations. It should be noted that this CMoN is integrator dependent, i.e. using different integrators can lead to different MoNs. However, as long as the integrator is sufficiently accurate to catch the nonlinear dynamics of the system, the resulting MoN suffices to evaluate the degree of nonlinearity of the dynamic system.

3.2.3 Geometric Interpretation and Numerical Examples

A geometric interpretation of definition (3.54) is given in Figure 3.2. The linearization $\frac{\partial \mathcal{Z}}{\partial x}(x_0)$

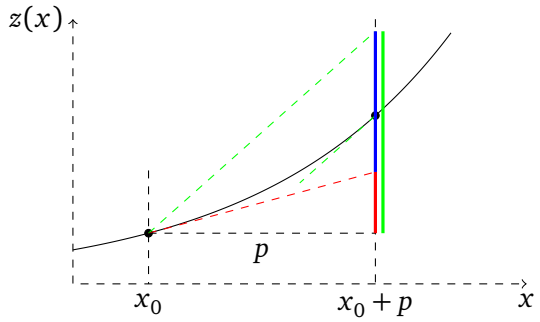


Figure 3.2: Geometric interpretation of the definition of the CMoN ((3.54)) in one dimensional space. The function $\mathcal{Z} : \mathbb{R} \to \mathbb{R}$. The direction is given by vector p. Red solid line: $\frac{\partial \mathcal{Z}}{\partial x}(x_0)p$. Green solid line: $\frac{\partial \mathcal{Z}}{\partial x}(x_0 + p)p$. Blue solid line: The difference between the red and the green one. The CMoN κ_3 is approximately half of the ratio between the length of the blue solid line over the red one.

is the slope of the red dashed line and the linearization $\frac{\partial \mathcal{Z}}{\partial x}(x_0 + p)$ is the slope of the green dashed line. The directional derivative maps the slopes into distances in the range space, as shown by the blue and red solid lines.

Consider a functional $\Xi(x(t))$ that is the numerical integration operator of the following ODE

$$\dot{x}(t) = e^{-5x(t)}, x(0) = x_0 \tag{3.55}$$

over a period of $T_s = [0, t_f]$. A 4^{th} order explicit Runge-Kutta operator is assigned to F and the simulation time is chosen to be $t_f = 0.01$s. The initial condition is set to be $x_0 = -1$. Two perturbed initial conditions $x_0^1 = -1.5, x_0^2 = -0.5$ are used to compare their sensitivities. The corresponding curvature-like MoN and the sensitivities are given in Table 3.1, where the CMoN is measured by definition (3.54) where x_0 is the initial condition and $p = x_0^i - x_0$ for $i = 1, 2$ respectively. The integration trajectories starting from the three initial conditions are shown in Figure 3.3. There are three observations from this numerical example

1. The value of CMoN depends on the numerical integration operator. In Table 3.1, the

	$s=5$		$s=10$		$s=50$	
	CMoN.	Sens.	CMoN.	Sens.	CMoN.	Sens.
x_0	0	0.11	0	0.12	0	0.12
x_0^1	17.02	-8.01	2.57	-1.89	0.63	0.01
x_0^2	2.06	0.62	1.85	0.62	1.85	0.62

Table 3.1: The CMoN and exact sensitivity measured at different initial conditions of the numerical integration operator Ξ described in (3.55) using different number of integration steps s.

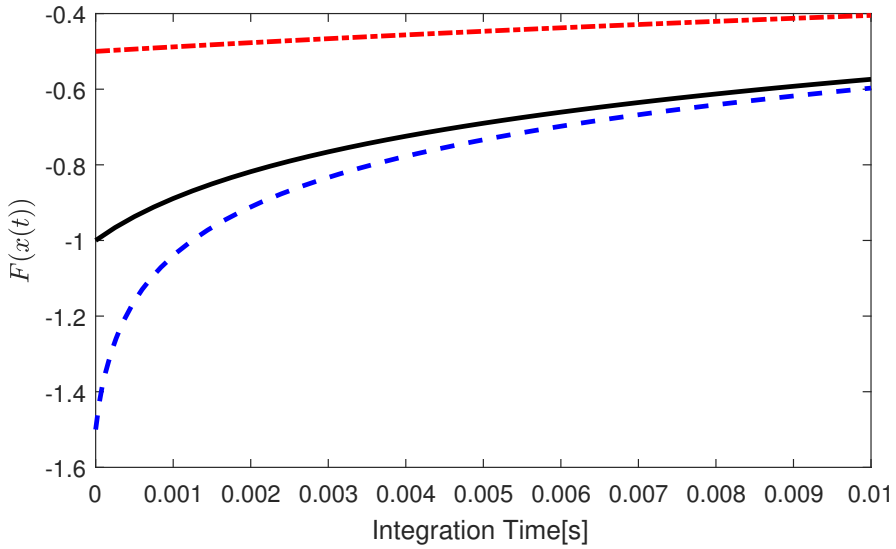


Figure 3.3: The integration trajectories of $\Xi(x(t))$ starting from $x(0) = x_0, x_0^1, x_0^2$ respectively.

values of CMoNs are significantly different when using different number of integration steps over the same time horizon.

2. The value of CMoN can reflect the degree of sensitivity deviation for a dynamic system with two different initial conditions. In Table 3.1, a larger CMoN value corresponds to a larger sensitivity distance. For example, when $s = 5$, the sensitivity of Ξ w.r.t. x_0^1 is -8.01, which is much further to 0.11 than 0.62, which is the sensitivity of Ξ w.r.t. x_0^2. Hence, the corresponding CMoN value is much bigger (17.02 vs 2.06).

3. When the integration is sufficiently accurate, the CMoN can correctly reflect the nonlinearity of system dynamics. In Figure 3.3, the integration trajectory starting from $x_0^2 = -0.5$ is much distant to the original trajectory from $x_0 = -1$ than that from $x_0^1 = -1.5$. Hence it can be concluded that the dynamic system (3.55) is not that

nonlinear in the region of $x(0) \in [-1.5, -1]$ and is more nonlinear in the region of $x(0) \in [-1, -0.5]$. The same conclusion can be drawn from third column of Table 3.1.

Therefore, the CMoN defined by (3.54) is highly potential for measuring the nonlinearity of dynamic system in the context fast NMPC algorithms, which usually solve a sequence of NLP problems with different initial conditions.

3.3 CMoN RTI

To speed up the on-line optimization of NMPC, a novel sensitivity updating scheme is proposed to overcome the limitations of existing inexact sensitivity updating algorithms described in Chapter 2. This new scheme need to not only exploit the multi-stage property of the NLP problem (2.10) for computational efficiency, but also to employ a metric to decide which Jacobian blocks should be updated. As discussed in Section 3.2.1, the CMoN (3.54) can be such a metric. Therefore, we name the RTI-based fast NMPC strategy using the novel sensitivity updating scheme as CMoN-RTI.

3.3.1 Computational Cost of Sensitivity Evaluation

Recall that in the multiple shooting, SQP-based NMPC algorithm described in Chapter 2, a sequence of QP subproblems is obtained by linearizing the NLP problem at every iteration (See the general QP (2.15) and the equivalent QP (2.22)). To linearize the equality constraint, which is formulated by enforcing the continuity condition of the predicted state trajectory, the sensitivity matrices $A_k \in \mathbb{R}^{n_x \times n_x}, B_k \in \mathbb{R}^{n_x \times n_u}$ of the numerical integration operator Ξ w.r.t the initialization z_k for each shooting interval has to be evaluated. The computational complexity of the sensitivity evaluation strongly depends on the selection of numerical integration scheme and in turn affects the computational burden of the NMPC algorithm. In the sequel, we show the computational cost for sensitivity evaluation by the example of chain of masses given in (1.6).

Consider $n = 15$ so the system has $n_x = 87$ states and $n_u = 3$ controls. To linearize the equality constraint, at each SQP iteration, there are N n_x by n_x and N n_x by n_u sensitivity matrices that need to be evaluated. The reference computational time for integration and sensitivity evaluation for each integration step, measured by running Matlab EXecutable (MEX) provided by CasADi toolbox automatic differentiation functions (Andersson, 2013) and CVODES (Serban and Hindmarsh, 2005), is given in Table 3.2. It can be seen that the sensitivity evaluation is much more computationally expensive than nominal integration. The more complex the integration scheme is, the more computational time it consumes. Although the numbers in Table 3.2 seem to be small, the overall computational time for evaluating

	Integration	Sensitivity
ERK4	0.02	0.32
IRK4	0.75	1.25
Adaptive	5*	100*

Table 3.2: Computational time[ms/step] of integration and sensitivity generation for problem (1.6). ERK4 stands for 4^{th} explicit Runge-Kutta schemes and IRK4 for its implicit counterpart. Adaptive refers to backward differential formula with adaptive step size that is adopted by CVODES. The symbol * means that the computational time is measured for the entire integration period instead of for each integration step, since the number of integration steps varies with different configurations when using an adaptive step size scheme.

sensitivities is quite big since multiple integration steps are usually performed in each shooting interval among N shooting intervals over the prediction horizon. For stiff or complex systems that require implicit or more complicated integration schemes, the computational burden for evaluating sensitivities is quite heavy.

3.3.2 Updating Logic based on CMoN

The CMoN for measuring the nonlinearity of state trajectory in the k—th shooting interval at sampling instant i can be defined as

$$K_k^i := \frac{\|\Xi_k(z_k^i) - \Xi_k(z_k^{i-1}) - \nabla\Xi_k(z_k^{i-1})q_k^{i-1}\|}{\|\nabla\Xi_k(z_k^{i-1})q_k^{i-1}\|}, \tag{3.56}$$

where $\nabla\Xi(z_k^i) = [A_k^i, B_k^i]$ is the sensitivity and q_k^{i-1} is the direction in variable space. As in RTI the initialization varies at each sampling instant, definition (3.56) can reflect the distance of sensitivities between two consecutive sampling instants.

The possibility to update only part of the Jacobian blocks in (2.24) is due to the fact that (A_k, B_k) uniquely depends on the corresponding initialization z_k of the k—th shooting interval. As a consequence, the Jacobian matrix B in (2.24) has a banded, block structure and replacing a subset of the Jacobian blocks will not affect the remaining ones. On the other hand, the evaluation of CMoN for the k—th shooting interval by (3.56) uniquely depends on information in this shooting interval, including $z_k^i, z_k^{i-1}, q_k^{i-1}$. Thus, the computation of sensitivity and the evaluation of CMoN can be both performed independently for each shooting interval, resulting in a stage-wise sensitivity updating scheme. An additional benefit from this property is that parallel implementation is straightforward. In fact, this is the intrinsic property of multiple shooting discretization method.

Once the values of CMoN for all shooting intervals are obtained, a question arises that how to use these values for the selection of updating subset, where sensitivities are exactly

evaluated. We define an updating logic which takes current CMoN values and sensitivities from previous sampling instants as inputs and provides the sensitivities of the current sampling instant as outputs.

Definition 3.3.1. In CMoN-RTI, a sensitivity updating logic $\mathcal{Q} : \mathbb{R} \times \mathbb{R}^{n_x \times (n_x+n_u)} \rightarrow \mathbb{R}^{n_x \times (n_x+n_u)}$ computes the sensitivity $\nabla \Xi_k$ by

$$\nabla \Xi_k(z_k^i) = \mathcal{Q}(K_k^i, \nabla \Xi_k(z_k^{i-1})),\ k = 0, 1, \ldots N-1, \tag{3.57}$$

where z_k^i is the initialization of (2.10) at sampling instant i, K_k^i is the CMoN computed by (3.56).

A general algorithm describing CMoN-RTI can be summarized in 3.1. In the sequel, two classes of updating logic are addressed. These two classes not only differ in the way of updating sensitivities, but also in the places of applying the updating logic among the algorithm steps.

Algorithm 3.1 A general algorithm of CMoN-RTI

1: Initialize $\mathbf{z}^i$ for $i > 0$
2: Formulate QP problem (2.15), excluding $B(\mathbf{z}^i)$
3: Compute CMoN K_k^i by (3.56) for all $k > 0$
4: Apply updating logic $\mathcal{Q}(K_k^i, \nabla \Xi(z_k^{i-1}))$ for all $k > 0$ and obtain $B(\mathbf{z}^i)$
5: Solve QP problem (2.15)

Fixed-Time Updating Logic

First we propose a class of fixed-time updating logic, where a fixed number N_f Jacobian blocks are updated at each sampling instant (Chen et al., 2017a). This strategy is very similar to Mixed-Level Schemes, where the first N_{frac} Jacobian blocks are updated. However, the advantage of CMoN-RTI is that the selection of the updating Jacobian blocks is not based on heuristics, but instead on the nonlinearity of the system. Only the sensitivities corresponding to most nonlinear shooting trajectories will be updated.

The fixed-time block updating CMoN-RTI scheme, hereafter refer to FTB-RTI is able to significantly reduce the computational cost for sensitivity evaluation, if the number N_f of updated Jacobian blocks is much smaller than the total number N, i.e. $N_f \ll N$. One may argue that the numerical and control performance may not be satisfactory by updating only a small portion of the Jacobian blocks. However, practical observations suggest that the proposed strategy is effective both in the case of constant reference along the prediction horizon and in that of a time-varying reference. In the first case the system nonlinearities are

typically excited at the beginning of the prediction window, while at the end the system is almost linear. In the second case only the parts of the reference trajectory subject to changes exhibit highly nonlinear behavior, requiring sensitivity updates. These observations reveal the facts that the number of Jacobian blocks that needed to be updated is usually small hence the fixed-time logic can often achieve a good control performance while saving considerably the computational burden.

The updating logic $\mathcal{Q}_1$ for FTB-RTI is given by Algorithm 3.2. In Algorithm 3.2, the

Algorithm 3.2 The fixed-time block updating logic $\mathcal{Q}_1$ for FTB-RTI

1: Choose N_f satisfying $N_f \in \mathbb{N}$ in $[0, N]$
2: Sort $K_k^i, k = 0, 1, \ldots, N-1$ in descending order and extract the first N_f indexes $(e_1, \ldots, e_{N_f})$
3: **for** $k = e_1, \ldots e_{N_f}$ **do**
4: Compute the exact $\nabla \Xi_k(z_k^i)$
5: **end for**

sensitivities that needed to be updated are selected by sorting the CMoN values for all shooting intervals. Therefore, $\mathcal{Q}_1$ has to be applied after the CMoN values of all shooting intervals are computed. Two remarks are made regarding the updating logic $\mathcal{Q}_1$. First, the computational time for computing sensitivities at each sampling instant is fixed. This may have two impacts. One is that if N_f is too small, the QP problem using the inexact Jacobian matrix may be infeasible, or granting unacceptable solutions. The other is that when the system under control is almost linear, e.g. at the steady state, updating N_f Jacobian blocks is unnecessary and wasting computational power. Therefore, a careful choice of N_f is of crucial importance for the success of $\mathcal{Q}_1$. Second, the type of sorting algorithm is expected to have just a little effect on the overall computational burden. For example, the Quick Sort algorithm adopted by Matlab has a complexity of $\mathcal{O}(N \log N)$, which is much lower than condensing with $\mathcal{O}(N^2)$. In addition, only the largest N_f CMoN values are needed in $\mathcal{Q}_1$, which further reduces the complexity of the sorting algorithm.

FTB-RTI is applied to the chain of masses problem (1.6). The control objective is to stabilize the chain by penalizing over the velocities and final positions of the masses within $T_f = 50$s. The initial positions and velocities of the masses are randomly assigned in space so that the sensitivity information at the initial condition is usually not a good representation of the ones computed on-line. The initial and final positions of the chain of masses are shown in Fig. 3.4.

The control performance of FTB-RTI is compared with that of three other schemes, namely, the standard RTI algorithm (Diehl et al., 2002), the ML-RTI with $N_m = 10$ (Albersmeyer et al., 2009), and the ADJ-RTI (Kirches et al., 2012). The sensitivities are calculated by

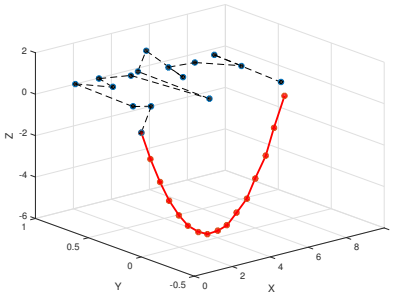


Figure 3.4: Initial (black dotted) and final positions (red solid) of the masses on the chain.

a fixed step implicit Runge-Kutta integrator (Quirynen, 2012), which is implemented by automatic differentiation in CasADi toolbox (Andersson, 2013) using c-code generation. After a condensing procedure (Vukov et al., 2013), the QP problems are solved by the active-set solver qpOASES (Ferreau et al., 2014) when $N = 40$, and by the sparse interior point solver Ipopt (Wächter and Biegler, 2006) when $N = 160$. The computing environment is Matlab R2016a on a PC with Intel core i7-4790 running at 3.60GHz.

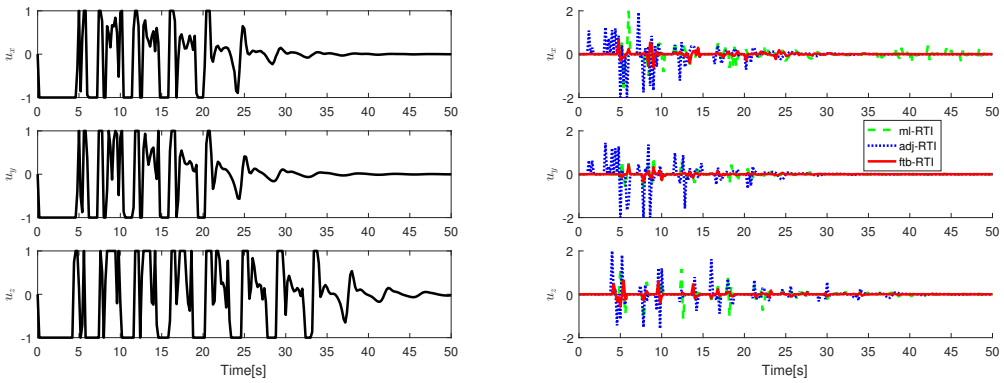


Figure 3.5: Control trajectories of the standard RTI (left) and the *differences* between the control trajectories of ML-RTI, ADJ-RTI, FTB-RTI and that of the standard RTI (right). The prediction horizon is $T = 32$s with $N = 160$ shooting points. The control variable is constrained within $[-1, 1]m/s$. For ML-RTI, the full sensitivity matrix is updated for every $N_m = 10$ sampling instants; For ADJ-RTI, the sensitivity matrix is computed off-line at the steady state trajectory; For FTB-RTI, $N_f = 10\% \times N = 16$ blocks in the sensitivity matrix are updated at every sampling instant.

In Fig. 3.5, one of the control trajectories of the standard RTI scheme with $N = 160$ are shown on the left, while the differences between the standard RTI trajectories and those generated by FTB-RTI, ML-RTI, ADJ-RTI are reported on the right. Fig. 3.6 shows the evolution of Karush-Kuhn-Tucker (KKT) values, which is the norm of the gradient of the Lagrangian of (2.10), as an indicator of the solution optimality at each time step. It

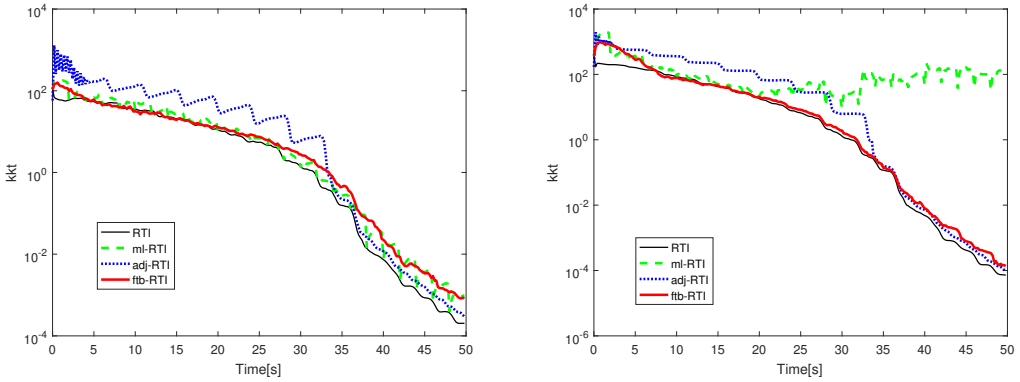


Figure 3.6: KKT evolution corresponding to results in Fig. 3.5. The prediction points are $N = 40$ (left) and $N = 160$ (right), respectively.

can be observed that the FTB-RTI control trajectory is the one closest to standard RTI one during the entire simulation, even if only 10% of sensitivity blocks are updated at every sampling instant. Performance in terms of KKT are comparable with the ones of RTI at the beginning of the simulation, when the nonlinearity of system is maximally excited (before 35s). Although ML-RTI shows a similar behavior when $N = 40$, the associated control cannot stabilize the chain when the longer prediction horizon is used, as it can also be seen from the KKT evolution. ADJ-RTI shows the poorest performance at the beginning of the simulation, both in terms of control variables and KKT values, in accordance with the coarse sensitivity approximation used. However, it is interesting to observe that the adjoint based compensation is very effective when the system is close to the steady state condition (after 35s).

In Table 3.3, the computational times and objective values corresponding to results in Fig. 3.5 are presented. The computational cost for sensitivities computation is significantly reduced by applying the three suboptimal algorithms ML-, ADJ- and FTB-RTI, whereas FTB-RTI has the smallest performance loss in terms of values of the objective function. In particular, for both $N = 40, 160$, the burden for sensitivity computation of FTB-RTI is smaller than that of ADJ-RTI, which computes only adjoint sensitivities on-line. A further advantage of FTB-RTI is that the trade-off between computational cost and solution accuracy can be enforced by tuning the parameter N_f in the range $[0\text{–}N]$. If $N_f = N$ FTB-RTI becomes the standard RTI scheme. Note that the computational time for solving the QPs of ML- and ADJ-RTI is higher than that of FTB-RTI, since large sensitivity inexactness has a negative influence on the QP solution, as previously observed in (Albersmeyer et al., 2009). Also note that we do not intend to pursue a real-time implementation by showing the computational time here, but to illustrate the effectiveness of the proposed algorithm.

Table 3.3: Average computational time corresponding to results in Fig.3.5. **Cond**: Condensing. **Int**: Nominal integration. **Sens**: sensitivity computation. **QP**: solving QP by qpOASES ($N = 40$) or Ipopt ($N = 160$). **Obj**: the normalized sum of the objective values w.r.t. the optimal one (obtained by running full SQP algorithm) of all sampling instants.

N=40	Cond[ms/call]	Int[ms/call]	Sens[ms/call]	QP[ms/call]	Total[ms/call]	Obj
RTI	14.72	60.66	98.17	4.88	194.91	1.085
ML-RTI($N_m = 10$)	9.47	61.56	9.90	5.74	98.17	1.197
ADJ-RTI	10.02	64.27	12.17	7.10	104.78	1.44
FTB-RTI($N_f = 4$)	14.41	60.68	**9.42**	4.99	101.77	**1.118**
N=160						
RTI	231.57	247.92	399.26	86.63	965.38	1.079
ML-RTI($N_m = 10$)	177.6	248.17	43.79	104.20	563.76	1.360
ADJ-RTI	203.49	251.85	52.68	160.42	668.44	1.903
FTB-RTI($N_f = 16$)	223.02	251.79	**43.11**	84.09	602.01	**1.101**

A Simple Adaptive Updating Logic

A fixed number of sensitivity updating is often not appropriate for controlling highly nonlinear and fast changing systems. On one hand, a sufficiently large N_f has to be chosen to catch the nonlinearity of the system. On the other hand, system nonlinearity varies as its operating condition changes. A universal fixed N_f may sometimes become insufficient, but sometimes conservative. Therefore, an adaptive updating logic which is able to adjust the number of sensitivity updating on-line is desired (Chen et al., 2017b).

To develop such an adaptive strategy, we first introduce a threshold $\eta_{pri} \geq 0, \eta_{pri} \in \mathbb{R}$ to assess the value of CMoN (3.56). Here the subscript $_{pri}$ denotes the *primal variable* and η_{pri} is the threshold for CMoN (3.56) working on the primal variable $\mathbf{z}$. For the numerical integration operator $\Xi_k(z_k)$, if its CMoN satisfies $K_k \geq \eta_{pri}$, $\Xi_k(z_k)$ is said to be nonlinear and its sensitivity $\nabla\Xi_k(z_k)$ is required to be updated. Conversely, if $K_k < \eta_{pri}$, $\Xi_k(z_k)$ is said to be (almost) linear and $\nabla\Xi_k(z_k)$ may be kept unchanged. The threshold η_{pri} defines a border that separates linear and nonlinear integration trajectories. In addition, the η_{pri} may be time-varying so that the distinction of linear and nonlinear shooting intervals is subject to operating conditions. The updating logic describing such a scheme is given in Algorithm 3.3.

Applying Algorithm 3.3 will implicitly (automatically) choose the sensitivity updating number N_f in Algorithm 3.2. This is because Algorithm 3.2 always updates sensitivities with N_f largest CMoN values. In Algorithm 3.3, the shooting intervals with larger CMoN values are given higher priority for sensitivity updating. As a result, in CMoN-RTI, the tuning of N_f in Algorithm 3.2 is equivalent to that of η_{pri} in Algorithm 3.3.

The choice of the threshold η_{pri} is a crucial factor for the success of Algorithm 3.3. A straightforward way is to set a constant threshold, i.e. $\eta^i_{pri} = \eta^0_{pri}$ for all sampling instants i. The value of η^0_{pri} may be determined by simulation results. When $\eta^0_{pri} = 0$, CMoN-RTI

Algorithm 3.3 The adaptive block updating logic $\mathcal{Q}_2$ for CMoN-RTI

Choose η_{pri}^i for $i > 0$
for $k = 0, 1, \ldots, N$ **do**
  if $K_k^i < \eta_{pri}^i$ **then**
    $\nabla\Xi_k(z_k^i) \leftarrow \nabla\Xi_k(z_k^{i-1})$
  else
    Compute the exact $\nabla\Xi_k(z_k^i)$
  end if
end for

becomes the standard RTI scheme where sensitivities are updated at every sampling instant. When $\eta_{pri}^0 \geq \max(K_k^i)$ for all k and i, no sensitivity is updated on-line and CMoN-RTI becomes the ADJ-RTI without a modified QP gradient. The stability and feasibility of such a scheme has been addressed by (Zanelli et al., 2016). As a consequence, the value of η_{pri}^0 can be chosen arbitrarily between $[0, \max(K_k^i)]$ to achieve a flexible tuning procedure. In practice, a number of simulations are usually needed to determine the desired η_{pri}.

Note that although $\eta_{pri} = \eta_{pri}^0$ is fixed, the resulting CMoN-RTI scheme achieves an adaptive number of sensitivity updating at each sampling instant. This is because the values of CMoNs are time-varying. When the system is approaching the steady state, most of the CMoNs evaluated are sufficiently small, resulting in a number of sensitivity updating as well. When the system is changing dramatically or is affected by external disturbances, most of the CMoN values are large so that most sensitivities are necessary to be updated.

The CMoN-RTI scheme using such an intuitive tuning strategy is applied to the inverted pendulum problem (1.1). Comparisons with the standard RTI algorithm (Diehl et al., 2002) and the ML-RTI (Albersmeyer et al., 2009) are provided. The ADJ-RTI schme (Kirches et al., 2012) failed to perform all the required control tasks, hence its performance is not presented. Different horizon lengths are used to demonstrate the effectiveness of this tuning strategy. In all simulations, the sensitivities are evaluated using automatic differentiation of the explicit Runge-Kutta method provided by CasADi toolbox (Andersson, 2013). The QP problems are solved by a structure exploiting solver qpDUNES (Frasch et al., 2015), without the condensing step. The computing environment is Matlab R2016a on a PC with Intel core i7-4790 running at 3.60GHz.

The references change every $t_r = 3$s and three different prediction lengths, $T_p = 1, 3, 8$s with shooting points $N = 20, 60, 160$, are analyzed. The threshold in CMoN-RTI is set to be a constant as $\eta_{pri} = \eta_{pri}^0$ and is tuned to obtain a good trade-off between computational cost and control performance. On the other hand, the sensitivity updating interval is chosen to be $N_m = 2$ for ML-RTI to obtain the best possible performance. Larger N_m would further reduce

computational cost but deteriorate control performance, even causing infeasible QP problems. All the simulations are performed by initializing the NMPC at an optimal trajectory, computed off-line, by solving the NLP problem at the first sampling instant until convergence.

The displacement and swing angle trajectories of the inverted pendulum obtained from the standard RTI, ML-, and CMoN-RTI, are presented in Fig. 3.7. In all cases, CMoN-RTI exhibits almost identical performance to the standard RTI, both in terms of tracking accuracy and control force regulation. As for ML-RTI, several control oscillations can be observed with $N = 60, 160$, e.g. around 12s. Note that, the sensitivity information cannot be updated more frequently than once every two steps: $N_m = 1$ would make ml-RTI coincide with standard RTI.

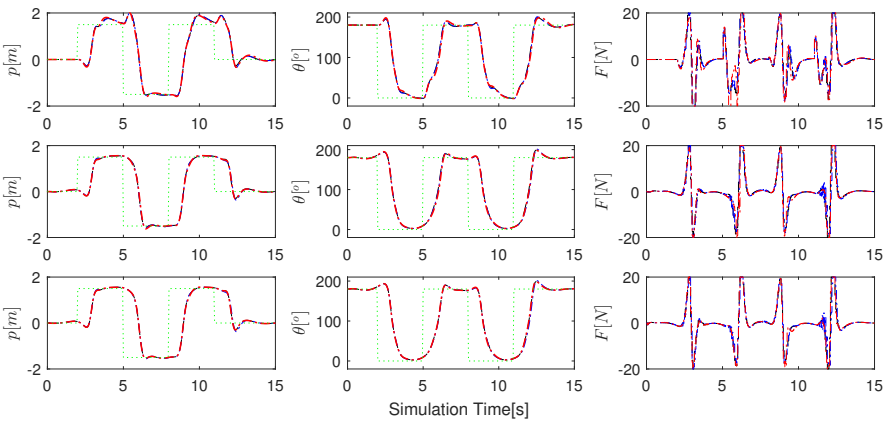


Figure 3.7: State and control trajectories of the inverted pendulum during simulation with prediction horizon lengths $N = 20, 60, 160$ from top to bottom. Black dashed line: standard RTI; Blue dot-dashed line: ML-RTI; Red dot-dashed line: CMoN-RTI; Green dotted line: reference.

In Fig. 3.8, the percentage of exactly updated sensitivity blocks at each sampling instant using CMoN-RTI is presented. Since the NMPC algorithm is initialized at an optimal trajectory, just a few sensitivities are updated at the beginning of the simulation, whereas several peaks can be observed in conjunction with the reference changes in the prediction horizon, respectively with $N = 20, 60, 160$. The maximal number of updated sensitivity blocks are $4, 4, 8$ when $N = 20, 60, 160$ respectively, since the corresponding number of predicted reference change is $1, 1, 2$. When a longer prediction horizon is used, the maximal percentage of updated sensitivities is smaller, making CMoN-RTI more promising in this case. The minimal percentage is nonzero because a shifting strategy is used, causing a high nonlinearity at the terminal shooting point in the prediction horizon (Diehl, 2001), and thus requiring the update.

The average computational time per sampling instant is given in Table 3.4. The computa-

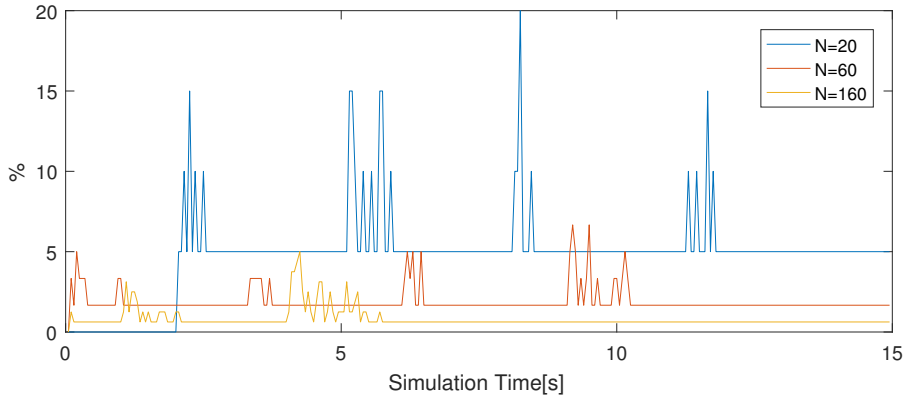


Figure 3.8: The percentage of exactly updated sensitivity blocks at each sampling instant during NMPC simulation using CMoN-RTI, with prediction horizon lengths $N = 20, 60, 160$.

tional time reduction for sensitivities is significant, especially when prediction horizon N is large. The overall time is reduced up to 45%, and a larger reduction is possible when sensitivity computations occupy more share in the overall computational cost with the increasing system dimension.

Table 3.4: Average computational time per sampling instant for all schemes considered in this paper with prediction horizon lengths $N = 20, 60, 100$. The term Int., Sens., QP. are the abbreviation of Integration, Sensitivity propagation, solving QP, respectively. The term ml., adj., c. are the abbreviation of ML-RTI, ADJ-RTI and CMoN-RTI, respectively.

N	Int.[ms]	Sens.[ms]			QP.[ms]			Total[ms]			η^0_{pri}
	all	rti	ml.	c.	rti	ml.	c.	rti	ml.	c.	
20	0.113	0.167	0.074	**0.016**	0.044	0.049	0.054	0.339	0.239	**0.187**	0.059
60	0.347	0.447	0.217	**0.026**	0.135	0.139	0.136	0.942	0.729	**0.510**	0.048
160	0.912	1.038	0.585	**0.028**	0.359	0.354	0.361	2.349	1.883	**1.363**	0.0197

In Fig. 3.9, the effect of the threshold η^0_{pri} on the value of objective function, the averagely and maximally updated sensitivity blocks when $N = 60$, is shown. The objective value is computed by summing up the squares of tracking and control deviations at all sampling instants. For each $\eta^0_{pri} \neq 0$, a suboptimal objective value is obtained due to the use of the inexact sensitivity scheme. These values are normalized w.r.t. the one of the standard RTI scheme. As η^0_{pri} grows, the numbers of averagely and maximally updated sensitivity blocks decrease and the objective value increases. The value of η^0_{pri} can be chosen in a considerable range, which demonstrates the effectiveness and flexibility of the tuning procedure of CMoNmon-RTI. In this work, different values of η^0_{pri} are chosen when using different prediction horizons as shown in Table 3.4. A more conservative η^0_{pri} is used in the case of a longer prediction horizon due to the increasing complexity of the NMPC problem.

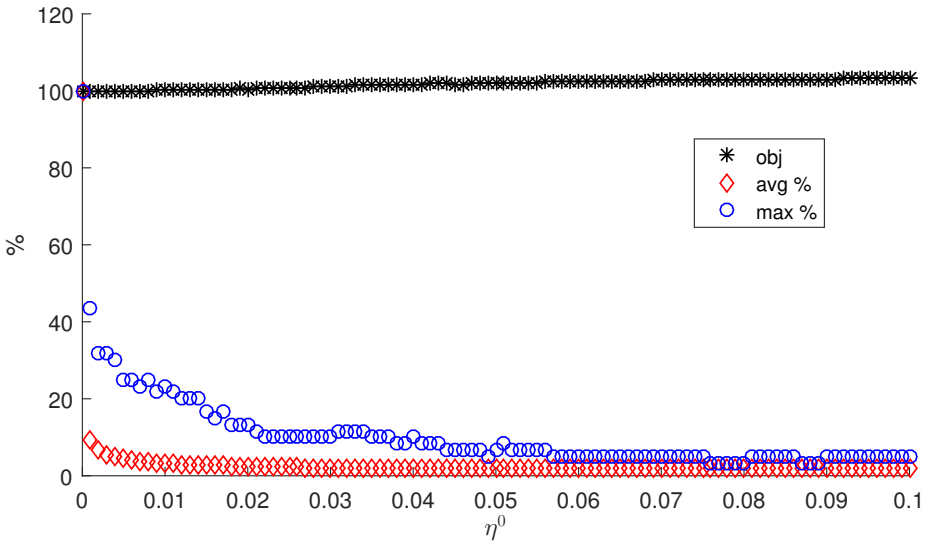


Figure 3.9: Effect of the threshold η^0_{pri} on the value of objective function, the averagely and maximally updated sensitivity blocks when $N = 60$. In this case, $\eta^0_{pri} = 0.048$ is chosen.

3.4 Implementing CMoN-RTI

When using the updating logic as summarized in Algorithms 3.2 and 3.3, two issues have to be addressed when estimating K^i_k by (3.56):

1. how to choose the direction vector q^{i-1}_k, and
2. how to compute $\nabla\Xi_k$.

3.4.1 The Direction Vector

The direction $q^{i-1}_k = z^i_k - z^{i-1}_k$ is given by the difference of the initialization of the shooting point k at two consecutive sampling instants. However, a precise definition has to consider that sensitivities may not have been updated over multiple sampling instants, and that a shifting strategy is used on the initialization (Diehl et al., 2002).

Suppose that, at sampling instant i, the k—th block of the Jacobian matrix has not been updated for r sampling instants, therefore it is equal to $\nabla\Xi_k(z^{i-r}_k)$. To ensure the validity of (3.56), the direction for computing CMoN should start from the sampling instant $i-r$. The CMoN is then computed as

$$\tilde{K}^i_k = \frac{\|\Xi_k(z^{old}_k + q^{i-1}_k) - \Xi_k(z^{old}_k) - \nabla\Xi_k(z^{old}_k)q^{i-1}_k\|}{\|\nabla\Xi_k(z^{old}_k)q^{i-1}_k\|}, \tag{3.58}$$

where $z_k^{old} = z_k^{i-r}$ and $q_k^{i-1} = z_k^i - z_k^{old}$. If a shifting on the initial guess (Diehl et al., 2002) is adopted, the corresponding sensitivities have to follow such a shifting. An intuitive shifting strategy between two consecutive sampling instants is to shift all state and control variables one shooting interval backward, say $z_k^{i+1} = z_{k+1}^i, k = 0, 1, \ldots, N-1$. In this case, the corresponding shifting on the old initialization to abandon the information of the first shooting interval is required as $z_{k+1}^{old} = z_k^{old}$.

3.4.2 Integration Scheme

The integration of the dynamics or the execution of the numerical integration operator $\Xi_k(z_k^i)$ is often performed simultaneously with the corresponding sensitivity propagation of $\nabla\Xi_k(z_k^i)$ (Albersmeyer, 2010; Quirynen et al., 2015). However, in the proposed scheme, the integration results are first used to estimate CMoN by means of (3.56), while the sensitivities are propagated later according to the updating logic in Algorithms 3.2 and 3.3. As a consequence, a deferred strategy for sensitivity propagation as described in (Albersmeyer, 2010) is adopted, where the dynamics are firstly integrated and all the intermediate steps are stored in memory, then the sensitivities are propagated after integration is finished. Note that no increase of computational burden is introduced in this way, however, additional memory allocation is required.

The complete scheme using the updating logic by Algorithm 3.3 is summarized in Algorithm 3.4.

Algorithm 3.4 CMoN-RTI using updating logic by Algorithm 3.3

Choose an initial point $(\mathbf{z}^0, \lambda^0, \mu^0)$
Set $q_k^{-1} \leftarrow \mathbf{0}, \eta_{pri}^0 \leftarrow 0, \Xi_k^{old} \leftarrow \mathbf{0}, \nabla\Xi_k^{old} \leftarrow \mathbf{0}, z_k^{old} \leftarrow \mathbf{0}$ for all k
for $i = 0, 1, \ldots$ **do**
  Compute H^i, g^i, B^i
  for $k = 0, 1, \ldots N-1$ **do**
    Perform integration and obtain Ξ_k^i
    Compute K_k^i by (3.58)
    Update $\nabla\Xi_k^i$ by Algorithm 3.3
    Update and shift $\Xi_k^{old}, \nabla\Xi_k^{old}$ and z_k^{old}
  end for
  Solve QP problem (2.15) and obtain $(\Delta\mathbf{z}^i, \lambda^i, \mu^i)$
  Update and shift the initialization by (2.16)
  Compute $q_k^i = z_k^{i+1} - z_k^{old}$ for all k
  Choose an appropriate η_{pri}^{i+1}
end for

4

Accuracy of the QP Solution and Convergence of SQP

In Chapter 3, we have presented a novel sensitivity updating scheme for RTI scheme called CMoN-RTI. Two updating logic are proposed that either update a fixed or an adaptive number of Jacobian blocks. Simulation results have shown that such schemes are able to reduce significantly the computational time for evaluating sensitivities, while largely maintaining the numerical and control performance. However, there remain some issues that are not clear:

- How the partial updating scheme and its tuning parameters, e.g. the number of updated Jacobian blocks, affect the numerical performance of RTI.
- How to choose optimally the tuning parameters w.r.t the trade-off between sensitivity computational cost and numerical performance of RTI.
- How the proposed scheme affects the local convergence of the algorithm, when applied to full SQP methods.

Since in RTI only one QP problem is solved at each sampling instant, the numerical performance of RTI is basically the numerical properties of a QP problem. Therefore, we first introduce the stability of the QP solution subject to parameter perturbations. In CMoN-RTI, such perturbations are Jacobian approximation error caused by partial sensitivity update. We

then show how far the solution of the QP problem in CMoN-RTI deviates from that in the standard RTI scheme. In this thesis, this deviation is called the *distance to optimum* (DtO). The relationship between the DtO and the value of the threshold in CMoN-RTI is also constructed. Based on this result, a novel adaptive threshold is proposed for the tuning of CMoN-RTI using updating logic by Algorithm 3.3, which guarantees the DtO respecting a certain bound. As a result, the tuning of the threshold is converted to the tuning of the DtO bound, which is much more straightforward for non-expert users. Finally, we prove that CMoN-SQP, which is CMoN-RTI applied in full SQP algorithm, is locally convergent by appropriate tuning. The convergence rate of CMoN-SQP is also tunable hence its flexibility is increased.

4.1 Stability of the QP Solution

Definition 4.1.1. Define a parametric problem QP($\mathbf{p}$) with a parameter vector $\mathbf{p} \in \mathbb{R}^{n_p}$ formulated as a matrix P in the equality constraint as

$$\begin{aligned} \min_{\Delta \mathbf{z}} \quad & \frac{1}{2}\Delta \mathbf{z}^\top H \Delta \mathbf{z} + \nabla \mathcal{L} \Delta \mathbf{z} && \text{(4.1a)} \\ s.t. \quad & b(\mathbf{z}^i) + (B(\mathbf{z}^i) + P^i)\Delta \mathbf{z} = 0, && \text{(4.1b)} \\ & c(\mathbf{z}^i) + C(\mathbf{z}^i)\Delta \mathbf{z} \leq 0 && \text{(4.1c)} \end{aligned}$$

where $\nabla \mathcal{L}$ is the gradient of the Lagrangian of (2.10), $P := \bar{B} - B$ the Jacobian approximation error, and $\bar{B}$ the inexact Jacobian with partial updated blocks. $\mathbf{p} = \mathrm{vec}(P) \in \mathbb{R}^{n_p}$ is the vectorization of P after eliminating zero elements.

By this definition, the exact sensitivity QP problem (2.15) is referred to as QP($\mathbf{0}$). Due to multiple shooting discretization, P has the following banded block structure

$$P = \begin{bmatrix} O_{n_x} & & & & \\ P_0 & O_{n_x} & & & \\ & P_1 & O_{n_x} & & \\ & & \ddots & \ddots & \\ & & & P_{N-1} & O_{n_x} \end{bmatrix}, \tag{4.2}$$

where O_a is a zero matrix of dimension a and $P_k \in \mathbb{R}^{n_x \times (n_x + n_u)}$ is the Jacobian approximation error of the $k-$th block, and in general is a dense matrix.

Definition 4.1.2. Define

$$\Delta y(\mathbf{p}) = (\Delta \mathbf{z}^\top(\mathbf{p}), \Delta \mu^\top(\mathbf{p}), \Delta \lambda^\top(\mathbf{p}))^\top \tag{4.3}$$

the solution triple of (4.1), where $\Delta\mathbf{z}(\mathbf{p}), \Delta\mu(\mathbf{p}), \Delta\lambda(\mathbf{p})$ are the increments of optimization variables, multipliers for inequality and equality constraints, respectively.

Here the solution of QP($\mathbf{p}$) are all variable increments, since QP (4.1) has a modified object gradient comparing to (2.15). It can be easily proved that these two formulations are equivalent (Diehl, Walther, Bock, and Kostina, 2010).

The following Theorem shows that the distance between the solutions of QP($\mathbf{0}$) and QP($\mathbf{p}$) is bounded, if the Jacobian approximation error is bounded.

Theorem 4.1.3 (QP Stability Theorem (Daniel, 1973))**.** *Let S and S' be the nonempty feasible sets of QP($\mathbf{0}$) and QP($\mathbf{p}$), respectively, defined by*

$$S = \{\Delta\mathbf{z} : b(\mathbf{z}^i) + B(\mathbf{z}^i)\Delta\mathbf{z} = 0\}, \tag{4.4}$$

$$S' = \{\Delta\mathbf{z} : b(\mathbf{z}^i) + (B(\mathbf{z}^i) + P^i)\Delta\mathbf{z} = 0\}. \tag{4.5}$$

Let $\Delta\mathbf{z}(\mathbf{0})$ and $\Delta\mathbf{z}(\mathbf{p})$ minimize QP($\mathbf{0}$) and QP($\mathbf{p}$) over S and S', respectively. Then there exists constants c and $\epsilon^* > 0$ such that $\|\Delta\mathbf{z}(\mathbf{p}) - \Delta\mathbf{z}(\mathbf{0})\| \leq c\epsilon$ whenever $\epsilon \leq \epsilon^*$ and $\epsilon = \|\bar{B} - B\| = \|P\|$.

The following Theorem analyzes the stability of a NLP solution subject to perturbations. Since QP problem is a special type of NLP problem, the theorem can also be applied to QP problems. It is shown that $\Delta y(\mathbf{p})$ is a unique minimizer of (4.1). Moreover, the active set is locally stable when the perturbation is sufficiently small.

Theorem 4.1.4 (Basic Sensitivity Theorem (Fiacco, 1984))**.** *If*

1. *the functions defining QP($\mathbf{p}$) are twice continuously differentiable in $\mathbf{z}$ and if their gradients w.r.t. $\mathbf{z}$ and the constraints are once continuously differentiable in $\mathbf{p}$ in a neighborhood of $\Delta\mathbf{z}^*$ and $\mathbf{p} = \mathbf{0}$.*

2. *the second-order sufficient conditions for a local minimum of QP($\mathbf{0}$) hold at $\Delta\mathbf{z}^*$, with associated Lagrange multipliers $\Delta\mu^*$ and $\Delta\lambda^*$.*

3. *the rows of constraint Jacobians $B(\mathbf{z}^i)$ and $C(\mathbf{z}^i)$ are linearly independent.*

4. *$\Delta\mu^* > 0$ when $c(\mathbf{z}^i) + C(\mathbf{z}^i)\Delta\mathbf{z}^* = 0$, i.e. strict complementary slackness holds,*

then

1. *$\Delta\mathbf{z}^*$ is a local isolated minimizer of QP($\mathbf{0}$) and the associated Lagrange multipliers $\Delta\mu^*$ and $\Delta\lambda^*$ are unique,*

2. *for* $\mathbf{p}$ *in a neighborhood of* $\mathbf{0}$*, there exists a unique, once continuously differentiable vector function* $\Delta y(\mathbf{p})$*, satisfying the second-order sufficient conditions for a local minimum of QP*$(\mathbf{p})$ *such that* $\Delta y(\mathbf{0}) = [\Delta \mathbf{z}^{*^\top}, \Delta \mu^{*^\top}, \Delta \lambda^{*^\top}]^\top = \Delta y^*$*, and hence* $\Delta \mathbf{z}(\mathbf{p})$ *is a locally unique local minimum of QP*$(\mathbf{p})$ *with associated unique Lagrange multipliers* $\Delta \mu(\mathbf{p})$ *and* $\Delta \lambda(\mathbf{p})$*,*

3. *for* $\mathbf{p}$ *in a neighborhood of* $\mathbf{0}$*, the set of active inequality constraints is unchanged, strict complementary slackness holds, and rows of the active constraint Jacobian are linearly independent at* $\Delta \mathbf{z}(\mathbf{p})$*.*

Finally, the following Theorem provides a linearly approximated relationship between the exact and inexact solutions.

Theorem 4.1.5 (First-order Approximation (Fiacco, 1984))**.** *A first order approximation of* $\Delta y(\mathbf{p})$ *in a neighborhood of* $\mathbf{p} = \mathbf{0}$ *is given by*

$$\Delta y(\mathbf{p}) = \Delta y(\mathbf{0}) + M^{-1}(\mathbf{0}) N(\mathbf{0}) \mathbf{p} + \mathcal{O}(\|\mathbf{p}\|^2) \tag{4.6}$$

where $M(\mathbf{0}), N(\mathbf{0})$ *are defined as*

$$M(\mathbf{0}) = \left[\begin{array}{c|ccc|ccc} \nabla^2_{\Delta \mathbf{z}} \mathcal{L}_{QP}(\mathbf{0}) & \nabla c_1^\top, & \ldots, & \nabla c_{n_c}^\top & \nabla b_1^\top, & \ldots, & \nabla b_{n_b}^\top \\ \hline -\Delta\mu_1 \nabla c_1 & -c_1 & & & & & \\ \vdots & & \ddots & & & & \\ -\Delta\mu_m \nabla c_{n_c} & & & -c_{n_c} & & & \\ \hline \nabla b_1 & & & & & & \\ \vdots & & & & & & \\ \nabla b_{n_b} & & & & & & \end{array}\right], \tag{4.7}$$

$$N(\mathbf{0}) = \left[-\nabla^2_{\mathbf{p}\Delta\mathbf{z}} \mathcal{L}_{QP}, \mu_1 \nabla_{\mathbf{p}} c_1^\top, \ldots, \mu_{n_c} \nabla_{\mathbf{p}} c_{n_c}^\top, -\nabla_{\mathbf{p}} b_1^\top, \ldots, -\nabla_{\mathbf{p}} b_{n_b}^\top\right]^\top, \tag{4.8}$$

and

$$\mathcal{L}_{QP}(\mathbf{0}) = \frac{1}{2} \Delta \mathbf{z}^\top H \Delta \mathbf{z} + \nabla \mathcal{L} \Delta \mathbf{z} + (b + B\Delta \mathbf{z})^\top \Delta \lambda_{QP} + (c + C\Delta \mathbf{z})^\top \Delta \mu_{QP}, \tag{4.9}$$

is the Lagrangian of QP$(\mathbf{0})$*,* ∇b_k *and* ∇c_k *are the kth row of B and C, respectively.*

Given Theorems 4.1.3-4.1.5, the inexact QP solution is related to that of the Jacobian approximation error. In addition, Theorem 4.1.5 provides a method for approximating the inexact solution given the exact solution and the perturbation. Indeed, in linear MPC when

the perturbation is the state measurement at every sampling instant, the perturbed solution can be computed analytically. This is known as explicit MPC (Bemporad et al., 2002).

4.2 An Advanced Tuning Strategy

In Algorithm 3.3, a fixed threshold η_{pri}^0 is used to assess the value of CMoN. Since this choice is application and control task dependent, the value of η_{pri}^0 becomes a tuning parameter. For non-experts, η_{pri}^0 has no physical meanings hence its tuning is not straightforward. As a result, one may desire an advanced tuning strategy for $\eta_{pri}^i, \forall i > 0$, that requires no prior knowledge of control systems and adapts to system operating conditions.

Given that CMoN-RTI is an inexact sensitivity RTI scheme, it is always preferred to update a minimal number of Jacobian blocks while being sufficiently close to the standard RTI scheme in terms of numerical performance. As the inexact Jacobian affects both the accuracy of the primal and dual solutions, an additional threshold η_{dual} for the dual variable must be introduced. By exploiting the results in the last section, the relationship between DtO and $(\eta_{pri}, \eta_{dual})$ can be constructed.

4.2.1 Theoretical Results

Assuming that $\mathbf{p}$ is in the neighborhood of the origin, (4.6) can be rewritten as

$$\Delta y(\mathbf{0}) = \Delta y(\mathbf{p}) - M^{-1}(\mathbf{p})N(\mathbf{p})\mathbf{p}\,, \tag{4.10}$$

where $\mathcal{O}(\|\mathbf{p}^2\|)$ is omitted. According to derivations from Appendix A, we have

$$N(\mathbf{p})\mathbf{p} = \begin{bmatrix} P^\top \Delta\lambda(\mathbf{p}) \\ O \\ -P\Delta\mathbf{z}(\mathbf{p}) \end{bmatrix}, \tag{4.11}$$

Therefore, at the sampling instant $i+1$ it follows that

$$\begin{aligned} \|e^{i+1}\|^2 &:= \|\Delta y(\mathbf{0}^{i+1}) - \Delta y(\mathbf{p}^{i+1})\|^2 \\ &\leq \|M^{-1}(\mathbf{p}^{i+1})\|^2 (\|P^{i+1^\top}\Delta\lambda(\mathbf{p}^{i+1})\|^2 + \|P^{i+1}\Delta\mathbf{z}(\mathbf{p}^{i+1})\|^2), \end{aligned} \tag{4.12}$$

where $\|e^{i+1}\|$ stands for *Distance to Optimum* (DtO), i.e. the distance between the solution of exact Jacobian QP($\mathbf{0}$) and that of the inexact Jacobian QP($\mathbf{p}$). Note that, given $M(\mathbf{p}^{i+1})$ a

finite dimensional and non-singular real matrix,

$$\rho^{i+1} := \|M^{-1}(\mathbf{p}^{i+1})\| \tag{4.13}$$

is bounded.Hence, the DtO (4.12) is bounded only if $\|P^{i+1^\top}\Delta\lambda(\mathbf{p}^{i+1})\|$ and $\|P^{i+1}\Delta\mathbf{z}(\mathbf{p}^{i+1})\|$ are bounded. The two bounds are referred as the *dual bound* and *primal bound*, respectively, and they are analyzed in the following.

Primal Bound

By applying the primal threshold η_{pri} to the updating logic Ω_2 by Algorithm 3.3, we have

$$\|P^{i+1}\mathbf{q}^i\| \leq 2\eta_{pri}^{i+1}\|V_{pri}^i\|, \tag{4.14}$$

where $\mathbf{q}^i = [q_0^{i^\top}, \ldots, q_{N-1}^{i^\top}]^\top$ and V_{pri}^i is a vector of directional sensitivities defined in Appendix A. Let us define $\alpha^{i+1} \geq 0$ as

$$\alpha^{i+1} = \begin{cases} \frac{\|P^{i+1}\Delta\mathbf{z}(\mathbf{p}^{i+1})\|}{\|P^{i+1}\mathbf{q}^i\|}, \text{ if } \|P^{i+1}\mathbf{q}^i\| > 0, \\ 1, \text{ if } \|P^{i+1}\Delta\mathbf{z}(\mathbf{p}^{i+1})\| = 0, \|P^{i+1}\mathbf{q}^i\| = 0 \end{cases}. \tag{4.15}$$

Hence, a bound in the direction of the primal variable is as follows

$$\|P^{i+1}\Delta\mathbf{z}(\mathbf{p}^{i+1})\|^2 \leq 4\alpha^{i+1^2}\eta_{pri}^{i+1^2}\|V_{pri}^i\|^2. \tag{4.16}$$

Dual Bound

Similarly, the threshold η_{dual} for the dual variable can be applied to adjoint sensitivities. Define the adjoint CMoN as

$$\tilde{K}_k^{i+1} := \frac{\|(\nabla\Xi^T(z_k^{i+1}) - \nabla\Xi^T(z_k^{old}))\Delta\lambda_{k+1}^i\|}{\|\nabla\Xi^T(z_k^{old}))\Delta\lambda_{k+1}^i\|}. \tag{4.17}$$

where $\nabla\Xi^T(z_k^{i+1})\Delta\lambda_{k+1}^i$ can be computed by efficient adjoint sensitivity schemes(Albersmeyer, 2010). The following updating logic can be added to Ω_2

$$\nabla\Xi_k^{i+1} = \begin{cases} \nabla\Xi_k(z_k^{old}), \tilde{K}_k^{i+1} < \eta_{dual}^{i+1}, \\ \nabla\Xi_k(z_k^{i+1}), \tilde{K}_k^{i+1} \geq \eta_{dual}^{i+1}. \end{cases} \tag{4.18}$$

As a result,

$$\|P^{i+1^\top}\Delta\lambda(\mathbf{p}^i)\| \leq \eta_{dual}^{i+1}\|V_{dual}^{i+1}\|. \tag{4.19}$$

Analogously, define $\beta^{i+1} \geq 0$ as

$$\beta^{i+1} = \begin{cases} \frac{\|P^{i+1^\top} \Delta\lambda(\mathbf{p}^{i+1})\|}{\|P^{i+1^\top} \Delta\lambda(\mathbf{p}^{i})\|}, \text{ if } \|P^{i+1^\top} \Delta\lambda(\mathbf{p}^{i})\| > 0, \\ 1, \text{ if } \|P^{i+1^\top} \Delta\lambda(\mathbf{p}^{i+1})\| = 0, \|P^{i+1^\top} \Delta\lambda(\mathbf{p}^{i})\| = 0 \end{cases}. \tag{4.20}$$

Hence, a bound in the direction of the dual variable is obtained as follows

$$\|P^{i+1^\top} \Delta\lambda(\mathbf{p}^{i+1})\|^2 \leq \beta^{i+1^2} \eta_{dual}^{i+1^2} \|V_{dual}^{i+1}\|^2 \tag{4.21}$$

Optimal Bound Estimation

If

$$\beta^{i+1^2} \eta_{dual}^{i+1^2} \|V_{dual}^{i+1}\|^2 \leq (1-c_1) \bar{e}^{i+1^2} / \rho^{i+1^2}, \tag{4.22}$$

$$4\alpha^{i+1^2} \eta_{pri}^{i+1^2} \|V_{pri}^{i}\|^2 \leq c_1 \bar{e}^{i+1^2} / \rho^{i+1^2}, \tag{4.23}$$

where $0 < c_1 < 1$ is a tuning parameter and $\bar{e}^{i+1} \in \mathbb{R}_0^+$ is a user-defined DtO tolerance, then the DtO defined in (4.12) satisfies $\|e^{i+1}\| \leq \bar{e}^{i+1}$. The primal and dual thresholds thus can be estimated by

$$0 \leq \eta_{pri}^{i+1} \leq \frac{\sqrt{c_1} \bar{e}^{i+1}}{2\alpha^{i+1} \rho^{i+1} \|V_{pri}^{i+1}\|} = \mathcal{U}_1(\eta_{pri}^{i+1}) \tag{4.24}$$

$$0 \leq \eta_{dual}^{i+1} \leq \frac{\sqrt{1-c_1} \bar{e}^{i+1}}{\beta^{i+1} \rho^{i+1} \|V_{dual}^{i+1}\|} = \mathcal{U}_2(\eta_{dual}^{i+1}). \tag{4.25}$$

Note that $\mathcal{U}_1, \mathcal{U}_2 : \mathbb{R} \to \mathbb{R}$ are implicit and discontinuous functions of $\eta_{pri}^{i+1}, \eta_{dual}^{i+1}$ according to (4.24), (4.25). $\alpha^{i+1}, \beta^{i+1}$ and ρ^{i+1} are functions of $\eta_{pri}^{i+1}, \eta_{dual}^{i+1}$ and can be computed once solved problem (4.1).

A formal solution to find the optimal $(\eta_{pri}^{i+1}, \eta_{dual}^{i+1})$ is then to solve the following problem

$$\max_{\eta_{pri}^{i+1}, \eta_{dual}^{i+1}} \quad \eta_{pri}^{i+1}, \eta_{dual}^{i+1} \tag{4.26a}$$

$$s.t. \quad \eta_{pri}^{i+1} - \mathcal{U}_1(\eta_{pri}^{i+1}) \leq 0, \tag{4.26b}$$

$$\eta_{dual}^{i+1} - \mathcal{U}_2(\eta_{dual}^{i+1}) \leq 0, \tag{4.26c}$$

The solution of problem (4.26) provides the maximum CMoN value, that corresponds to the minimum number of sensitivity updates while guaranteeing a bounded DtO, at every sampling instant. Note that for a given $\bar{e}^{i+1} \geq 0$, there always exists at least one feasible

solution to (4.26), i.e. $(\eta_{pri}^{i+1}, \eta_{dual}^{i+1}) = 0$, and CMoN-RTI becomes the standard RTI scheme.

A Practical Implementation

Problem (4.26) can be solved via enumeration, which requires to repeatedly solve problem (4.1) (up to $N+1$ times). However, this is computationally prohibitive and undermining the advantage of CMoN-RTI. A practical approach to avoid solving problem (4.26) is setting the two thresholds at their upper bounds in (4.24) and (4.25), using approximated information from previous sampling instants.

Firstly, the unknown ρ^{i+1} in (4.13) is replaced by ρ^0. The rationale for such choice is given by the fact that the matrix M in (4.7) is very sparse and its smallest singular value ρ is practically close to 0. Moreover, ρ has only a finite number of values. By choosing $\bar{e}^{i+1}$ sufficiently close to 0, the inequalities (4.24) and (4.25) are independent of ρ.

Secondly, as shown in (4.15) and (4.20), $(\alpha^{i+1}, \beta^{i+1})$ cannot be practically computed, since the Jacobian approximation error P^{i+1} is not known. However, it can be shown that the DtO tolerance can always be respected by using large enough but finite parameters $(\bar{\alpha}^{i+1}, \bar{\beta}^{i+1})$ (See Appendix A). In this paper, such two parameters are chosen to be

$$\bar{\alpha}^{i+1} = c_2 \frac{\|\Delta \mathbf{z}(\mathbf{p}^i)\|}{\|\mathbf{q}^{i-1}\|},\ \bar{\beta}^{i+1} = c_2 \frac{\|\Delta \lambda(\mathbf{p}^i)\|}{\|\Delta \lambda(\mathbf{p}^{i-1})\|}, \tag{4.27}$$

where $c_2 > 0$ is a scaling parameter and $(\bar{\alpha}^{i+1}, \bar{\beta}^{i+1})$ are approximated by using information from previous sampling instants. Intuitively, when the system is approaching steady state and the controller is converging "on the fly" (Gros et al., 2016), information from previous sampling instants are good approximations of that at the current sampling instant. A more detailed and formal discussion is given in Appendix A. Finally, the approximated thresholds estimates are given by

$$\eta_{pri}^{i+1} = \frac{\sqrt{c_1}\bar{e}^{i+1}}{2c_2\bar{\alpha}^{i+1}\rho^0\|V_{pri}^{i+1}\|} \tag{4.28}$$

$$\eta_{dual}^{i+1} = \frac{\sqrt{1-c_1}\bar{e}^{i+1}}{c_2\bar{\beta}^{i+1}\rho^0\|V_{dual}^{i+1}\|}. \tag{4.29}$$

A summary of CMoN-RTI using the advanced tuning strategy is given in Algorithm 4.1.

Algorithm 4.1 CMoN-RTI using the advanced tuning strategy

1: Choose an initial point $(\mathbf{z}^0, \lambda^0, \mu^0)$
2: Choose tuning parameters $c_1 > 0, c_2 > 0$
3: Set $q_k^{-1} \leftarrow \mathbf{0}, \eta^0 \leftarrow 0, \Xi_k^{old} \leftarrow \mathbf{0}, \nabla\Xi_k^{old} \leftarrow \mathbf{0}, z_k^{old} \leftarrow \mathbf{0}$ for all k
4: **for** $i = 0, 1, \ldots$ **do**
5: Compute $\nabla\mathcal{L}^i, H^i, B^i$
6: **for** $k = 0, 1, \ldots N-1$ **do**
7: Perform integration and obtain Ξ_k^i
8: Choose the DtO tolerance $\bar{e}^i$
9: Compute $K_k^i, \tilde{K}_k^i$ by (3.58) and (4.17)
10: Update $\nabla\Xi_k^i$ by Ω_2 in Algorithm 3.3 and (4.18)
11: Update and shift $\Xi_k^{old}, \nabla\Xi_k^{old}$ and z_k^{old}
12: **end for**
13: Solve QP (4.1) and obtain $(\Delta\mathbf{z}^{QP}, \Delta\lambda^{QP}, \Delta\mu^{QP})$
14: Update and shift the initialization by (2.16)
15: Compute $q_k^i = z_k^{i+1} - z_k^{old}$ for all k
16: Compute $(\eta_{pri}^{i+1}, \eta_{dual}^{i+1})$ by (4.28) and (4.29).
17: **end for**

4.2.2 Simulation Results

Algorithm 4.1 is applied to two NMPC examples, an inverted pendulum and a chain of masses. Numerical integration and sensitivity generation are performed by a 4$^{\text{th}}$ order explicit Runge-Kutta integrator with 4 steps per shooting interval, provided by CasADi toolbox (Andersson, 2013) using automatic differentiation. The QP problems are condensed by eliminating the state variables (Vukov et al., 2013) using algorithms presented in (Andersson, 2013), which have a quadratic computational complexity in horizon length. The condensed QPs are then solved by an interior point NLP solver Ipopt (Wächter and Biegler, 2006), which, in this work, is configured to solve convex QP problems using Mehrotra's predictor-corrector approach, thus avoiding drastic variation of computational time due to frequent active-set changes. The computing environment is Matlab R2016a on a PC with Intel core i7-4790 running at 3.60GHz, and all the aforementioned time-critical steps are compiled as Matlab executable.

Inverted Pendulum

The inverted pendulum is given in (1.1) in Chapter 1. Here, a time-varying reference is given to the inverted pendulum to track different horizontal displacements and swing angles. A *perfect* initial guess is chosen by optimally solving the first optimization problem off-line. A short ($N = 40$) and a long ($N = 120$) prediction horizons are applied with a control interval $T_s = 0.05$s. The tolerance on DtO in CMoN-RTI follows the rule given in (4.42), where a

conservative and an aggressive setting are denoted by s_1, s_2, respectively:

$$s_1 : (\epsilon^{abs} = 10^{-2}, \epsilon^{rel} = 10^{-2}) \tag{4.30}$$

$$s_2 : (\epsilon^{abs} = 10^{-1}, \epsilon^{rel} = 10^{-1}) \tag{4.31}$$

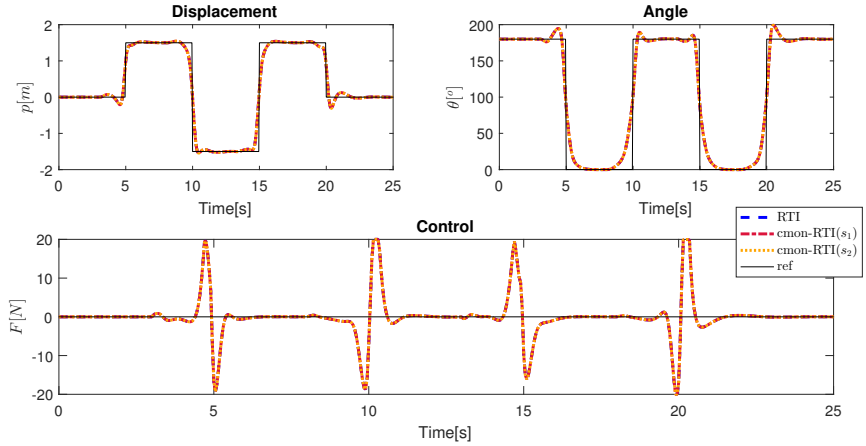


Figure 4.1: State and control trajectories of the inverted pendulum with $N = 40$. The reference signals change every 5 seconds. The constraints are $\|p\|_\infty \leq 1$ and $\|F\|_\infty \leq 20$. The configurations s_1, s_2 are assigned by (4.30) and (4.31). CMoN-RTI control performance is indistinguishable from that of Standard RTI. The trajectories obtained by using $N = 120$ is not shown as they are identical to the ones shown in the plot.

In Figure 4.1, the closed-loop state and control trajectories generated by the standard RTI and CMoN-RTI with the two prediction horizons and two tolerance settings are shown. The control performance of CMoN-RTI is indistinguishable to that of the standard RTI scheme. In Figure 4.2, the percentage of exactly computed Jacobian blocks per sampling instant is given. The CMoN-RTI scheme is able to adapt to operating conditions by computing more exact sensitivities when the reference changes, as the peaks occur at around $t = 3, 8, 13, 18$s. A more aggressive setting results to less exact sensitivity computations. A significant reduction of the percentage is observed when $N = 120$, making CMoN-RTI adequate to deal with the case of long prediction horizons. Fig. 4.3 shows the DtO at each sampling instant together with the user-defined tolerance. An additional QP with an exact Jacobian matrix is solved at each sampling instant to estimate the DtO. In all cases, the DtO is strictly lower than the tolerance.

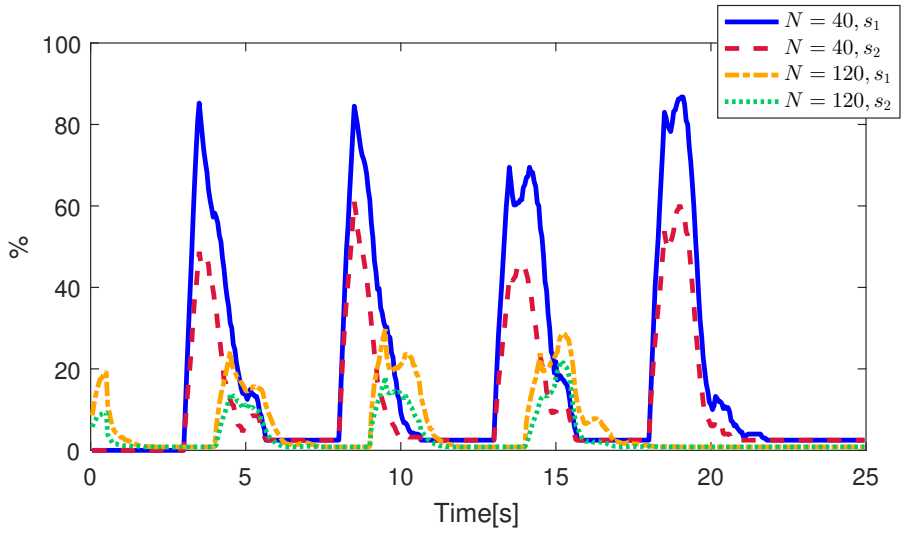


Figure 4.2: Percentage of exactly evaluated Jacobian blocks per sampling instant. The percentage starts from 0% when $N = 40$ since there is no reference change within the prediction horizon in the first 3 seconds. CMoN-RTI is able to adapt to reference changes, as can be seen from the peaks at around $t = 3, 8, 13, 18$s.

Chain of Masses with Nonlinear Springs

A chain of masses is given in (1.6), where the masses are connected by forces in (1.7) provided by linear springs. Here, nonlinear springs (Enns, 2010) are considered with forces

$$F_i(t) = D(x_i(t) - x_{i-1}(t))(1 - \frac{L}{\|x_i(t) - x_{i-1}(t)\|_2}) + F_{NL}\,, \tag{4.32}$$

where

$$F_{NL} = D_1(x_i(t) - x_{i-1}(t))\frac{(\|x_i(t) - x_{i-1}(t)\|_2 - L)^3}{\|x_i(t) - x_{i-1}(t)\|_2}. \tag{4.33}$$

A total of 50 simulations are performed using the standard, ML-, ADJ- and CMoN-RTI, with randomly assigned initial positions and velocities of the masses, see e.g. Fig 4.4, for the positions and control trajectories generated in one of the simulations. For ML-RTI, the entire constraint Jacobian matrix is updated every $N_m = 2$ sampling instants; For ADJ-RTI, the Jacobian matrix is computed off-line at the steady state trajectory; For CMoN-RTI, the DtO tolerance is set to s_2 in (4.31). To ensure that an accurate representation of the system is always used in the controller, at least 10% of Jacobian blocks are updated at each sampling instant. These blocks are those having the largest values of CMoN, hence exhibiting the most significant nonlinearities.

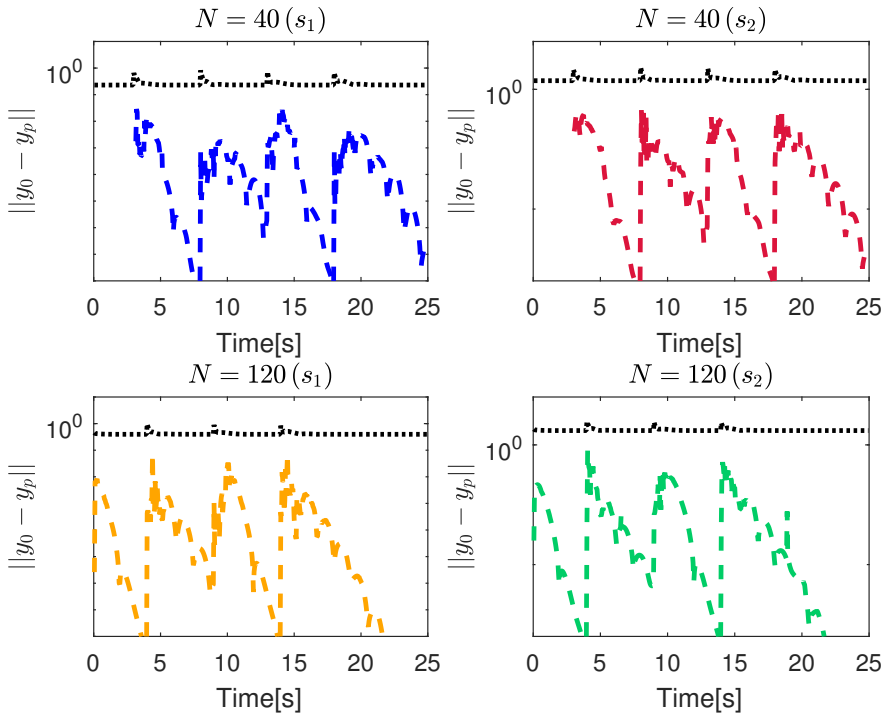


Figure 4.3: DtO estimated on-line (colored dashed line) and the user-defined tolerance (black dotted line) for $N = 40, 120$, with two tolerance configurations in (4.30) and (4.31). The DtO increases when the system is subject to a large reference change (at around $t = 3, 8, 13, 18$s). For $N = 40$, the DtO is zero in the first 3s since there is no reference change within the prediction horizon and the DtO is zero. In all cases, the DtO is strictly lower than the tolerance.

Control performance, numerical robustness, and efficiency of CMoN-RTI are evaluated and compared with standard RTI, ML-RTI, and ADJ-RTI. Control performance are firstly evaluated by collecting statistics of the *stabilizing time* t_{st}, defined as

$$t_{st} = \text{argmin}(t) \tag{4.34a}$$

$$s.t. \quad \|u(t_i)\|_\infty < 0.1, \forall t_i \geq t, \tag{4.34b}$$

from the 50 simulations. In Fig. 4.5, the statistics of the standard, ML-, ADJ- and CMoN-RTI with $N = 40, 80, 160$ are shown. Note that, if the chain is not stabilized within 50s, we set $t_{st} = 50$s, which is a conservative choice since the stabilization process may take far more than 50s. For all simulations, RTI is able to stabilize the chain within 50s. The mean and interquartile range (IQR) of t_{st} of CMoN-RTI is very close to those of the standard RTI. This means that CMoN-RTI has a similar control performance to the standard RTI in most of the

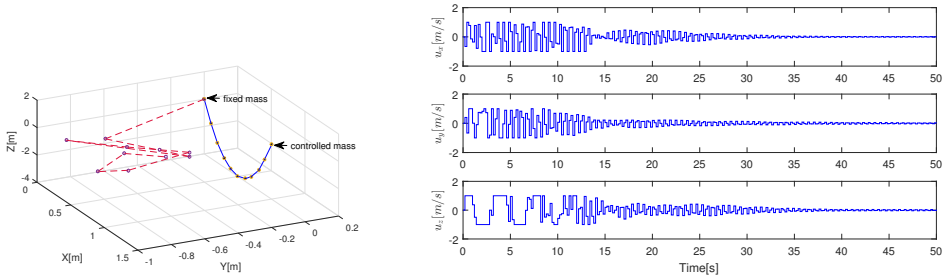


Figure 4.4: Initial and final positions of masses (left) and the control trajectories (right) in one of the simulations using the standard RTI scheme. One end of the chain is fixed on a wall, while the other end is free and under control. The control interval is $T_c = 0.2$s. The control inputs are constrained by $\|u(t)\|_\infty \leq 1$.

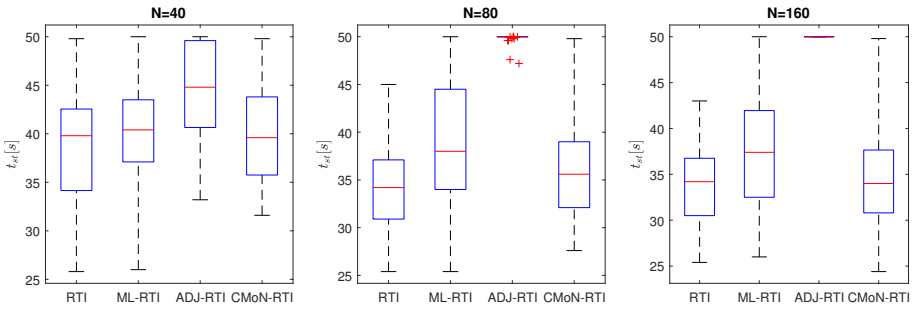


Figure 4.5: The time t_{st} needed to stabilize the chain of masses using RTI, ML- and CMoN-RTI in the total 50 simulations. The chain of masses is considered to be stabilized at time t_{st} that is computed by (4.34). For all schemes, the stabilizing time is set to be $t_{st} = 50$s if the chain is not stabilized within 50s.

situations. On the other hand, ML-RTI has a similar stabilizing time to RTI when N is short, whereas t_{st} grows significantly as N becomes larger. ADJ-RTI, initialized at the steady state trajectory, is not able to provide acceptable control performance, especially when N is large.

The control performance is also evaluated by assessing the optimality of each controller. In Fig. 4.6, the average Karush-Kuhn-Tucker (KKT) value, i.e. the norm of the gradient of the Lagrangian of the NLP (2.9), at each sampling instant is presented. It can be observed that

- ML-RTI KKT values exhibit oscillatory behavior since the Jacobian update is performed every $m = 2$ sampling instants only.

- As the system converges "on the fly", the KKT of CMoN-RTI decreases smoothly as that of the standard RTI.

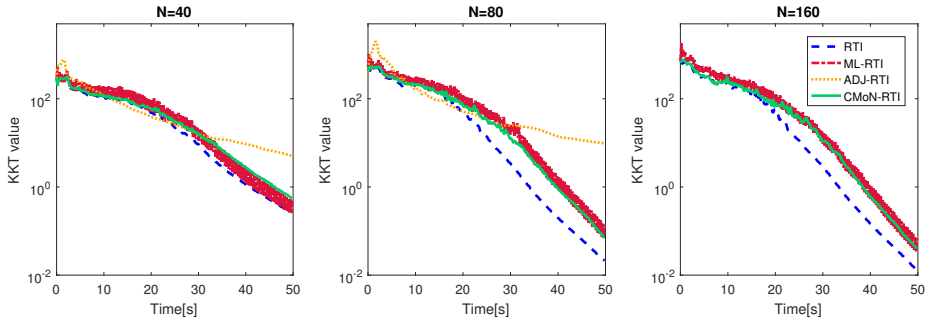


Figure 4.6: The average KKT value at every sampling instant of the successfully stabilized cases using RTI, ML- and CMoN-RTI. The KKT value is computed as the norm of the Lagrangian of the NLP (2.9) as an indicator of optimality.

Table 4.1: The number of simulations (among 50) where each controller *cannot* stabilize the chain.

N	40	80	160
ML-RTI	4	5	7
ADJ-RTI	7	42	50
CMoN-RTI	0	0	0

Table 4.2: The average computational time per sampling instant in milliseconds[ms] for the standard RTI, ML-RTI, and CMoN-RTI, with prediction length $N = 40, 80, 160$. The integration and sensitivity generation are performed by a 4^{th} order explicit Runge-Kutta integrator with 4 steps per shooting interval. The adjoint sensitivity is computed by automatic differentiation of the integration with an adjoint variable. The QP problem is solved by Ipopt using Mohrotra's predictor-corrector approach.

N	40			80			160		
	RTI	ML.	**CMoN.**	RTI	ML.	**CMoN.**	RTI	ML.	**CMoN.**
Integration	1.6			3.1			6.0		
Sensitivity	70.7	35.9	**22.7**	144.4	72.8	**34.7**	309.8	155.7	**56.0**
Adjoint sensitivity	0	2.8		0	5.7		0	9.9	
Condensing	5.6	3.5	5.7	20.2	12.6	19.8	89.9	56.9	89.4
QP	8.9	9.1	9.2	23.3	24.9	24.3	91.1	94.6	93.5
Total	90.4	57.0	**42.9**	202.8	129.5	**97.2**	513.7	346.6	**260.5**

As for the numerical robustness, the number of simulations where each controller fails to stabilize the system within 50s is reported in Tab. 4.1 . Given that the initial condition of each simulation is randomly assigned, the numerical robustness or the sensitivity w.r.t. initialization of each controller can be assessed. CMoN-RTI is able to stabilize the chain within 50s in all situations, although the maximal stabilizing time is larger than that of the standard

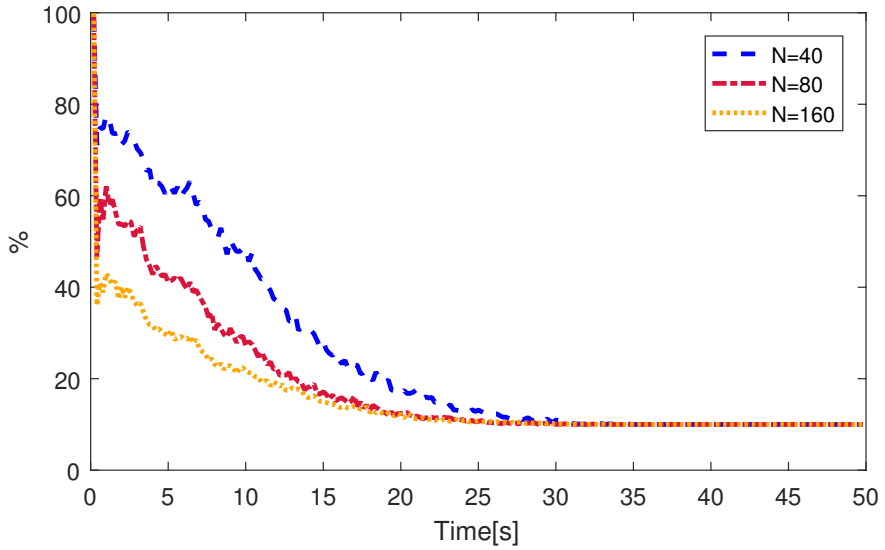


Figure 4.7: The average percentage of exactly updated Jacobian blocks at every sampling instant of the stabilized cases using CMoN-RTI. At least 10% of the Jacobian blocks are updated at each sampling instant.

RTI (see Fig. 4.5). ML-RTI has a few failed cases if $N = 40$ and this number increases as the prediction horizon grows. Not surprisingly, ADJ-RTI has the poorest robustness since its success considerably relies on the quality of the off-line Jacobian matrix.

To evaluate computational efficiency, the average percentage of exactly updated Jacobian blocks using CMoN-RTI at every sampling instant is reported in Fig. 4.7. In the first 20s, the KKT values of CMoN-RTI and RTI are almost identical, however, the number of updated Jacobian blocks is at most 80% and it reduces to 40% when N becomes larger. After $t = 30$s, when the system is close to its steady state, updating only 10% of the blocks allows to still maintain small KKT values. Table 4.2 shows the average computational time of each controller per sampling instant. Each time critical step is shown separately for a clear view on the computational cost distribution. The usage of CMoN-RTI can reduce significantly the cost for the computation of sensitivities, which dominates the overall computations in all cases. Note that the computational efficiency of CMoN-RTI can be further improved if structure-exploiting QP solvers are adopted, thus avoiding the condensing step (Vukov et al., 2013). However, for consistency, the condensing step has been introduced in all simulations and the corresponding dense QPs have been solved. The computational time of ADJ-RTI is not reported because of its poor control performance, see Fig. 4.5 and Table 4.1.

4.3 CMoN-SQP

Although Algorithm 4.1 represents an inexact sensitivity updating scheme in the framework of RTI, it can be straightforwardly extended to the SQP framework, in which a sequence of QP problems is solved until convergence is achieved. The resulting algorithm, denoted hereafter as CMoN-SQP, falls into the field of inexact Jacobian SQP methods. In this framework, the Two-side-Rank-One (TR1) updating SQP algorithm has been proposed in (Griewank and Walther, 2002) for equality constrained problems. Similar to the famous Symmetric-Rank-One (SR1) updating scheme (Nocedal and Wright, 2006), the TR1 scheme requires Hessian and Jacobian updates to satisfy both direct and adjoint secant conditions. This method is extended to linearly inequality constrained problems in (Diehl et al., 2010) and its local convergence is proved.

Differently from the TR1 scheme which adopts a rank one sensitivity update, CMoN-SQP achieves a block update by exploiting the structure of the problem. In addition, the primal and dual bounds are satisfied, instead of enforcing secant conditions. In the following, local convergence of CMoN-SQP is proved and it is shown that the convergence rate is tunable via the choice DtO tolerance.

4.3.1 Local Convergence of CMoN-SQP

Consider the parametric QP problem (4.1). Assume the active set is locally fixed thanks to Theorem 4.1.4. Solving problem (4.1) in a SQP algorithm is equivalent to solve the following nonlinear system

$$F(\mathbf{y}) = 0, \mathbf{y} := \begin{bmatrix} \mathbf{z} \\ \lambda \end{bmatrix}, F(y) = \begin{bmatrix} R^\top \nabla \mathcal{L}(\mathbf{z}, \lambda) \\ B(\mathbf{z}) \\ C_a(\mathbf{z}) \end{bmatrix} \tag{4.35}$$

where λ denotes the multiplier for equality and active inequality constraints, C_a contains the active constraints, and R is a matrix with orthonormal column vectors, such that $\nabla C_a R = 0$ (Diehl et al., 2010). The Jacobian matrix of the nonlinear system is

$$\dot{F}(\mathbf{y}^i) = \frac{\partial F}{\partial \mathbf{y}}(\mathbf{y}^i) = \begin{bmatrix} R_i^\top H^i & R_i^\top \nabla B^\top(\mathbf{z}^i) \\ \nabla B(\mathbf{z}^i) & \\ \nabla C_a(\mathbf{z}^i) & \end{bmatrix} \tag{4.36}$$

where H^i is an approximation of the exact Hessian, i.e. the Gauss-Newton approximation which is independent of the multiplier λ. Let J_i be an approximation of the exact Jacobian

$\dot{F}(\mathbf{y}^i)$ with

$$J_i = \begin{bmatrix} R_i^\top H^i & R_i^\top \nabla B^\top(\mathbf{z}^i) \\ \nabla \tilde{B}(\mathbf{z}^i) & \\ \nabla C_a(\mathbf{z}^i) & \end{bmatrix} \tag{4.37}$$

The following theorem indicates that the proposed scheme is convergent when $\mathbf{p}$ is in the neighborhood of $\mathbf{0}$.

Theorem 4.3.1. *Let $F : \mathcal{V} \to \mathbb{R}^{n_y}, \mathcal{V} \subset \mathbb{R}^{n_y}$ be continuously differentiable. Consider the two sequences*

$$\{y_*\} : y_*^{i+1} = y_*^i + \Delta y_*^i \tag{4.38}$$

$$\{y_p\} : y_p^{i+1} = y_p^i + \Delta y_p^i \tag{4.39}$$

where

$$\Delta y_*^i = -\dot{F}^{-1}(y_*^i)F(y_*^i) \tag{4.40}$$

$$\Delta y_p^i = -J^{-1}(y_p^i)F(y_p^i) \tag{4.41}$$

Assume that

1. *the Jacobian matrix is invertible and is uniformly bounded and has uniformly bounded inverses,*
2. *there exists a $\kappa_0 < 1$ such that $\|\Delta y_*^{i+1}\| \leq \kappa_0 \|\Delta y_*^i\|$ for all $i > m_1, m_1 \in \mathbb{N}$. Hence, starting from $y^0 \in \mathcal{V}$, the sequence $\{y_*\}$ converges to a local optimizer y_*^+,*
3. *$J(y_p^i)$ is generated by Algorithm 4.1,*

Then,

1. *there always exists a set of scalars $\{i \in \mathbb{N}^+ | \bar{e}^i \geq 0\}$ such that the distance between the sequences $\{y_p\}$ and $\{y_*\}$ is sufficiently small at each iteration,*
2. *there always exists a set of scalars $\{i \in \mathbb{N}^+ | \bar{e}^i \geq 0\}$ and a κ_2 satisfying $\kappa_0 \leq \kappa_2 < 1$, such that $\|\Delta y_p^{i+1}\| \leq \kappa_2 \|\Delta y_p^i\|$ for all $i > m_2, m_2 \in \mathbb{N}$, and the sequence $\{y_p\}$ converges to $y_p^+ = y_*^+$ starting from y^0.*

The proof is given in Appendix A. Theorem 1 shows that the Jacobian approximation error can be controlled by using user-defined DtO tolerances, hence the convergence property can be satisfied by using appropriate tuning configurations. The convergence rate is also shown to be tunable, which increases the flexibility of the proposed algorithm. If $\bar{e}^i = 0, \forall i \geq 0$, CMoN-SQP becomes the standard SQP algorithm with the same convergence rate.

4.3.2 Numerical Examples

As an example, the CMoN-SQP scheme is applied to two NLP problems from the Hock & Schittkowski Optimization Suite (Hock and Schittkowski, 1980). The NLP problems hs060 and hs077 contain nonlinear equality constraints, linear inequality constraints and least square objectives, which are suitable for using Gauss-Newton Hessian approximations. These one-stage problems do not have a structured Jacobian matrix, and are only used to demonstrate the convergence properties of the proposed method.

Figure 4.8 shows the results of the application of SQP algorithms with four different settings to the NLP problems. On the horizontal axis the iteration index is represented, while on the vertical axis the distance between the actual and optimal solution is displayed. The SQP using exact Hessian with exact Jacobian (setting 1) converges quadratically. The SQP using Gauss-Newton Hessian with exact Jacobian (setting 2) converges superlinearly. The tolerance DtO in CMoN-SQP is chosen as

$$\bar{e}^i = \epsilon^{abs}\sqrt{n} + \epsilon^{rel}\|\Delta y^i\| \tag{4.42}$$

where $(\epsilon^{abs}, \epsilon^{rel})$ are the absolute and relative tolerances, n is the number of optimization variables, and $y = (x, \lambda, \mu)$ is the optimal triple. Such choice ensures that the DtO tolerance scales with the size of the problem and the scale of the variable values(O'Donoghue et al., 2013). A conservative setting of CMoN-SQP scheme (setting 3), converges at the same rate of setting 2 while the exact Jacobian is not evaluated at more than the 70% of the iterations. For the less conservative setting of CMoN-SQP (setting 4), the iterations in which the exact Jacobian is not evaluated are marked by an "x" below the horizontal axis. The convergence rate is slightly slower than that of setting 2 and 3, however, the exact Jacobian is not evaluated at more than 85% of the iterations.

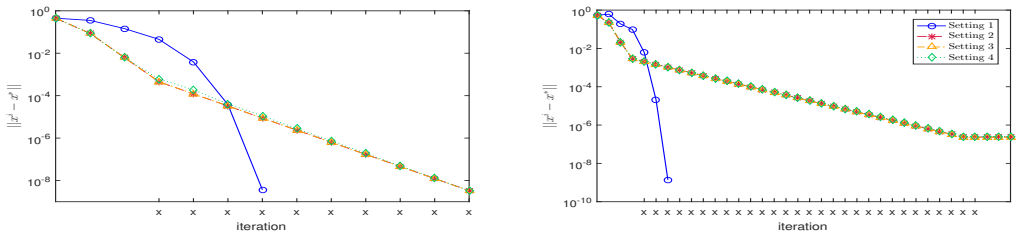


Figure 4.8: Convergence behavior of the example hs060 (left) and hs077 (right). The DtO tolerance is computed by (4.42). Setting 1: exact Hessian and exact constraint Jacobian; Setting 2: Gauss-Newton Hessian and exact constraint Jacobian; Setting 3: CMoN-SQP with conservative tuning parameters ($\epsilon^{abs} = 10^{-3}, \epsilon^{rel} = 10^{-3}$); Setting 4: CMoN-SQP with extreme tuning parameters ($\epsilon^{abs} = 10^{-1}, \epsilon^{rel} = 10^{-1}$). For Setting 4, the iterations at which the exact Jacobian is not evaluated are marked by an "x" below the horizontal axis.

5

Partial Condensing and Matrix Factorization

As shown in (2.23), the QP problem arising from multiple shooting has a block sparse structure. This formulation can be solved by sparsity-exploiting algorithms directly. Alternatively, a dense QP can be formulated by eliminating the state variables and solved by efficient dense QP solvers. While in Chapter 4 the CMoN-RTI scheme with several sensitivity updating logic are proposed to reduce the computational burden of RTI-based NMPC algorithms when preparing the QP problem (2.22), this Chapter focuses on further reducing the computational cost for solving the QP problem. With the help of partial sensitivity updates, a partial condensing algorithm is proposed to save condensing efforts for solving dense QP problems. In addition, a partial matrix factorization algorithm is proposed to solve sparse QP problems.

5.1 Prepare dense QP in CMoN-RTI

Recall that the following QP problem has to be solved at each sampling instant.

$$\begin{aligned}
\min_{\Delta\mathbf{s},\Delta\mathbf{u}} \quad & \sum_{k=0}^{N-1}\Big(\frac{1}{2}\begin{bmatrix}\Delta s_k\\ \Delta u_k\end{bmatrix}^\top H_k \begin{bmatrix}\Delta s_k\\ \Delta u_k\end{bmatrix} + g_k^\top \begin{bmatrix}\Delta s_k\\ \Delta u_k\end{bmatrix}\Big) && \text{(5.1a)}\\
& + \frac{1}{2}\Delta s_N^\top H_N \Delta s_N + g_N^\top \Delta s_N \\
s.t. \quad & \Delta s_0 = \hat{x}_0 - s_0, && \text{(5.1b)}\\
& \Delta s_k = A_{k-1}\Delta s_{k-1} + B_{k-1}\Delta u_{k-1} + d_{k-1},\ k = 1,\dots,N && \text{(5.1c)}\\
& C_k \begin{bmatrix}\Delta s_k\\ \Delta u_k\end{bmatrix} \le -c_k,\ k = 0,1,\dots,N-1, && \text{(5.1d)}\\
& C_N \Delta s_N \le -c_N. && \text{(5.1e)}
\end{aligned}$$

An equivalent multi-stage formulation can be written as

$$\begin{aligned}
\min_{\Delta\mathbf{z}} \quad & \sum_{k=0}^{N}\frac{1}{2}\Delta z_k^\top H_k \Delta z_k + g_k^\top \Delta z_k && \text{(5.2a)}\\
s.t. \quad & E_0 \Delta z_0 = \hat{x}_0 - s_0, && \text{(5.2b)}\\
& E_{k+1}\Delta z_{k+1} = D_k z_k + d_k,\ k = 0,1,\dots,N-1, && \text{(5.2c)}\\
& C_k \Delta z_k \le -c_k,\ k = 0,1,\dots,N, && \text{(5.2d)}
\end{aligned}$$

where $\Delta z_k = [\Delta s_k^\top, \Delta u_k^\top]^\top$ and $\Delta z_N = \Delta s_N$ and

$$\begin{aligned}
& D_k = [A_k, B_k],\ E_k = [\mathbb{I}, 0],\ , k = 0,\dots,N-1, && \text{(5.3)}\\
& E_N = \mathbb{I}. && \text{(5.4)}
\end{aligned}$$

Here $\mathbb{I}$ is a n_x by n_x identity matrix and 0 is a n_x by n_u zero matrix.

5.1.1 Full Condensing

The computational cost for solving the QP problem (2.22) with sparsity-exploiting algorithm grows linearly with the prediction horizon length N as $\mathcal{O}(N)$ (Axehill, 2015). However, this formulation results in $(N+1)n_x + Nn_u$ decision variables hence a large dimensional QP problem has to be solved. Alternatively, one can apply the condensing procedure to formulate a dense QP with only Nn_u decision variables. In this case, the sparsity of the original problem is completely lost and the computational cost for solving such a dense QP grows cubically with N as $\mathcal{O}(N^3)$ (Ferreau et al., 2014). In addition, the condensing step itself has a complexity of

$\mathcal{O}(N^3)$ but can be reduced to $\mathcal{O}(N^2)$ (Andersson, 2013).

The condensing step aims at eliminating the state variables Δs_k from the decision variable $\Delta \mathbf{z}$. Exploiting the coupling constraint (5.1c), yields the following dense QP problem:

$$\min_{\Delta U} \quad \frac{1}{2}\Delta U^\top H_c \Delta U + g_c^\top \Delta U \tag{5.5a}$$
$$s.t. \quad C_c \Delta U \leq -c_c, \tag{5.5b}$$

where $\Delta U = [\Delta u_0^\top, \ldots, \Delta u_{N-1}^\top]^\top$ and H_c, g_c, C_c, c_c are vectors and matrices of appropriate dimension after condensing. The condensing procedure refers to the computation of these vectors and matrices. Write the coupling constraints in a compact form as

$$\begin{bmatrix} \Delta s_1 \\ \Delta s_2 \\ \vdots \\ \Delta s_N \end{bmatrix} = \underbrace{\begin{bmatrix} G_{0,0} & & & \\ G_{1,0} & G_{1,1} & & \\ \vdots & & \ddots & \\ G_{N-1,0} & G_{N-1,1} & \cdots & G_{N-1,N-1} \end{bmatrix}}_{G} \begin{bmatrix} \Delta u_0 \\ \Delta u_1 \\ \vdots \\ \Delta u_{N-1} \end{bmatrix} + \underbrace{\begin{bmatrix} L_1 \\ L_2 \\ \vdots \\ L_N \end{bmatrix}}_{L} \tag{5.6}$$

where $G_{i,j} \in \mathbb{R}^{n_x \times n_u}$ and G, L can be computed using block-wise forward substitutions as in Algorithm 5.1.

Algorithm 5.1 Calculation of G, L in (5.6) with complexity $\mathcal{O}(N^2)$ and $\mathcal{O}(N)$ respectively (Andersson, 2013)

```
Initialize L_0 ← Δs_0 = x̂_0 − s_0
for i = 0, ..., N − 1 do
    G_{i,i} ← B_i
    for k = i + 1, ..., N − 1 do
        G_{k,i} ← A_k G_{k−1,i}
    end for
    L_{i+1} ← A_i L_i + a_i
end for
```

Note that H_k and g_k can be written as

$$H_k = \begin{bmatrix} Q_k & S_k \\ S_k^\top & R_k \end{bmatrix}, \; g_k = \begin{bmatrix} g_k^s \\ g_k^u \end{bmatrix}, \; , k = 0, \ldots, N-1, H_N = Q_N \tag{5.7}$$

where $Q_k \in \mathbb{R}^{n_x \times n_x}, S_k \in \mathbb{R}^{n_x \times n_u}, R_k \in \mathbb{R}^{n_u \times n_u}$ and $g_k^s \in \mathbb{R}^{n_x}, g_k^u \in \mathbb{R}^{n_u}$. The computation of g_c originally requires a complexity of $\mathcal{O}(N^2)$, but in the following an algorithm is given with a complexity of $\mathcal{O}(N)$ in Algorithm 5.2.

Algorithm 5.2 Calculation of g_c in (5.5) with complexity $\mathcal{O}(N)$ (Andersson, 2013)

1: Initialize $w_N \leftarrow g_N^s + Q_N L_N$
2: **for** $k = N-1, \ldots, 1$ **do**
3: $\quad g_{c_k} \leftarrow g_k^u + S_k^\top L_k + B_k^\top w_{k+1}$
4: $\quad w_k \leftarrow g_k^s + Q_k L_k + A_k^\top w_{k+1}$
5: **end for**
6: $g_{c_0} \leftarrow g_0^u + S_0^\top + B_0^\top w_1$

The most expensive part of the condensing is the computation of H_c, which is a dense matrix and can be written as

$$H_c = \begin{bmatrix} H_{0,0} & \cdots & H_{0,N-1} \\ \vdots & \ddots & \vdots \\ H_{N-1,0} & \cdots & H_{N-1,N-1} \end{bmatrix} \tag{5.8}$$

where $H_{i,j} \in \mathbb{R}^{n_u \times n_u}$. H_c can be efficiently computed by Algorithm 5.3.

Algorithm 5.3 Calculation of H_c in (5.5) with complexity $\mathcal{O}(N^2)$ (Andersson, 2013)

1: **for** $i = 0, \ldots, N-1$ **do**
2: $\quad W_{N,i} \leftarrow Q_N G_{N,i}$
3: $\quad$ **for** $k = N-1, \ldots, i+1$ **do**
4: $\quad\quad H_{k,i} \leftarrow S_k^\top G_{k,i} + B_k^\top W_{k+1,i}$
5: $\quad\quad W_{k,i} \leftarrow Q_k G_{k,i} + A_k^\top W_{k+1,i}$
6: $\quad$ **end for**
7: $\quad H_{i,i} \leftarrow R_i + B_i^\top W_{i+1,i}$
8: **end for**

The inequality constraint in (5.16) can be re-written as

$$\begin{bmatrix} C_0^s & & & \\ & C_1^s & & \\ & & \ddots & \\ & & & C_{N-1}^s \end{bmatrix} \begin{bmatrix} \Delta s_0 \\ \Delta s_1 \\ \vdots \\ \Delta s_{N-1} \end{bmatrix} + \begin{bmatrix} C_0^u & & & \\ & C_1^u & & \\ & & \ddots & \\ & & & C_{N-1}^u \end{bmatrix} \begin{bmatrix} \Delta u_0 \\ \Delta u_1 \\ \vdots \\ \Delta u_{N-1} \end{bmatrix} \leq - \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{N-1} \end{bmatrix}, \tag{5.9}$$

$$C_N^s \Delta s_N \leq -c_N \tag{5.10}$$

where $C_k^s \in \mathbb{R}^{n_c \times n_x}, C_k^u \in \mathbb{R}^{n_c \times n_u}, c_k \in \mathbb{R}^{n_c}$ for $k = 0, \ldots, N-1$ and $C_N^s \in \mathbb{R}^{n_{cN} \times n_x}, c_N \in \mathbb{R}^{n_{cN}}$. Assuming that the condensed Jacobian and residual for the inequality constraint has the

following form

$$C_c = \begin{bmatrix} C_{0,0} & & & \\ \vdots & \ddots & & \\ C_{N-2,0} & \cdots & C_{N-2,N-2} & \\ C_{N-1,0} & \cdots & C_{N-1,N-2} & C_{N-1,N-1} \\ C_{N,0} & \cdots & C_{N-,N-2} & C_{N,N-1} \end{bmatrix}, \; c_c = \begin{bmatrix} c_{c_0} \\ \vdots \\ c_{c_{N-1}} \end{bmatrix} \tag{5.11}$$

where $C_{i,j} \in \mathbb{R}^{n_c \times n_u}, c_{c_k} \in \mathbb{R}^{n_c}$. The computation of C_c and c_c are given in Algorithm 5.4.

Algorithm 5.4 Calculation of C_c and c_c in (5.5) with complexity $\mathcal{O}(N^2)$.

1: **for** $i = 0, \ldots, N-1$ **do**
2: $\quad C_{i,i} \leftarrow C_i^u$
3: $\quad c_{c_i} \leftarrow c_i + C_i^s L_i$
4: $\quad$ **for** $k = i+1, \ldots, N-1$ **do**
5: $\quad\quad C_{k,i} \leftarrow C_k^s G_{k-1,i}$
6: $\quad$ **end for**
7: $\quad C_{N,i} \leftarrow C_N^s G_{N-1,i}$
8: **end for**
9: $c_{c_N} \leftarrow c_N + C_N^s L_N$

If there are only box constraints for ΔU, i.e. $C_k^s = 0$, then $C_{i,i} = C_i^u = \mathbb{I}$ and C_c would be an identity matrix. If there are box constraints for both ΔU and Δs_k, then $C_k^s = \mathbb{I}, C_k^u = \mathbb{I}$ hence the lower triangular part of C_c is identical to G and the diagonal part is one.

Once the optimal solution ΔU is obtained, the full solution $\Delta \mathbf{z}$ and Lagrangian multipliers λ associated with the equality constraints (5.2b) and (5.2c) must be recovered. The gradient of the Lagrangian of (5.2) at its optimal solution $\Delta \mathbf{z}^*$ is given by

$$H\Delta \mathbf{z}^* + g + B^\top \lambda + C^\top \mu, \tag{5.12}$$

where H, B, C have structure in (2.23). Note that the Lagrangian multipliers μ associated with the inequality constraints are identical for dense and sparse QP problems. Therefore, $\Delta \mathbf{z}$ and λ can be computed by Algorithm 5.5, where $\lambda = [\lambda_0^\top, \ldots, \lambda_N^\top]^\top, \mu = [\mu_0^\top, \ldots, \mu_N^\top]^\top$, $\lambda_k \in \mathbb{R}^{n_x}$, $k = 0, \ldots, N$, $\mu_k \in \mathbb{R}^{n_c}$, $k = 0, \ldots, N-1$ and $\mu_N \in \mathbb{R}^{n_{cN}}$.

Algorithm 5.5 Recovering $\Delta\mathbf{z}$ and λ in (5.2) with complexity of $\mathcal{O}(N)$.

1: **for** $i = 0, \ldots, N-1$ **do**
2: $\quad \Delta s_{k+1} \leftarrow A_k \Delta s_k + B_k \Delta u_k + a_k$
3: **end for**
4: $\lambda_N \leftarrow Q_N \Delta s_N + g_N^s + C_N^s \mu_N$
5: **for** $i = N-1, \ldots, 0$ **do**
6: $\quad \lambda_i \leftarrow A_i^\top \lambda_{i+1} + Q_i \Delta s_i + S_i \Delta u_i + C_i^{s^\top} \mu_i + g_i^s$
7: **end for**

5.1.2 Partial Condensing

In (Axehill, 2015), it is shown that the choice for formulating the QP problems does not have to be binary: either fully condensed or sparse. An equivalent QP problem can be constructed with a level of sparsity in between these two choices. The underlying idea is called *block condensing* or *partial condensing*, that is to divide the horizon N in condensing blocks, each consisting of N_{pc} consecutive stages or shooting intervals. For each condensing block, the traditional condensing algorithm is performed to obtain a dense problem which is a sub-problem of the original QP problem. As a result, one can easily trade-off between the sparsity and the dimension of the QP problem to achieve a faster QP solving procedure. (Axehill, 2015) analyze the impact of partial condensing on Riccati based sparse linear algebra for soving QP problems. Using an optimal chosen block size N_{pc}, the computational complexity for solving the QP problem grows linearly in the number of controls n_u. (Kouzoupis, Quirynen, Frasch, and Diehl, 2015b) combine partial condensing with the dual newton strategy for solving the QP problem and achieves a significant speed up both in condensing and QP solving.

The idea of partial condensing is to divide the horizon N into blocks, each consisting N_{pc} consecutive stages. Hence, the number of blocks is $N_b = N/N_{pc}$. To ensure an integer-valued N_b, N must be a positive multiple of the block size N_{pc}. The decision variable of the partial condensed QP problem is

$$\Delta\tilde{\mathbf{z}} = [\Delta s_0^\top, \Delta U_0^\top, \ldots, \Delta s_{N_b-1}^\top, \Delta U_{N_b-1}^\top, \Delta s_N^\top]^\top \tag{5.13}$$

where $U_k = [\Delta u_k^\top, \Delta u_{k+1}^\top, \ldots, \Delta u_{k+N_{pc}-1}^\top]^\top$. Hence, for the m-th condensing block, $m = 0, \ldots, N_b - 1$, the decision variable is

$$\Delta z^{[m]} = [\Delta s_0^{[m]}, \Delta u_0^{[m]}, \ldots, \Delta u_{N_{pc}-1}^{[m]}]. \tag{5.14}$$

When $N_{pc} = N$ and $N_b = 1$, the full condensing is recovered. Note that the full condensing algorithms 5.1 to 5.4 completely eliminate the state variables from the optimization variable,

but for partial condensing the first state variable of each block is an optimization variable, hence the partial condensing algorithms have to be slightly modified. In each condensing block, the coupling constraint (5.6) becomes

$$\begin{bmatrix} \Delta s_0 \\ \Delta s_1 \\ \vdots \\ \Delta s_{N_{pc}-1} \end{bmatrix} = \underbrace{\begin{bmatrix} G_{0,0} & & & 0 \\ G_{1,0} & G_{1,1} & & 0 \\ \vdots & & \ddots & \vdots \\ G_{N_{pc}-1,0} & G_{N_{pc}-1,1} & \cdots & G_{N_{pc}-1,N_{pc}-1} & 0 \end{bmatrix}}_{G} \begin{bmatrix} \Delta s_0 \\ \Delta u_0 \\ \Delta u_1 \\ \vdots \\ \Delta u_{N_{pc}-1} \end{bmatrix} + \underbrace{\begin{bmatrix} L_0 \\ L_1 \\ L_2 \\ \vdots \\ L_{N_{pc}-1} \end{bmatrix}}_{L}, \quad (5.15)$$

where G and L are computed in the following algorithm.

Algorithm 5.6 Calculation of G, L in (5.15) with complexity $\mathcal{O}(N_{pc}^2)$ and $\mathcal{O}(N_{pc})$ respectively (Kouzoupis et al., 2015b)

Initialize $L_0 \leftarrow 0, B_{-1} \leftarrow \mathbb{I}$
for $i = 0, \ldots, N_{pc} - 1$ **do**
  $G_{i,i} \leftarrow B_{i-1}$
  for $k = i+1, \ldots, N_{pc} - 1$ **do**
    $G_{k,i} \leftarrow A_{k-1} G_{k-1,i}$
  end for
  if $i > 0$ **then**
    $L_i \leftarrow A_{i-1} L_{i-1} + a_{i-1}$
  end if
end for

As a result of partial condensing, a new but equivalent QP problem is formulated as

$$\min_{\Delta \mathbf{z}} \quad \sum_{k=0}^{N_b-1} \frac{1}{2} \Delta z^{[k]\top} \tilde{H}_k \Delta z^{[k]} + \tilde{g}_k^\top \Delta z^{[k]} \quad (5.16a)$$

$$s.t. \quad \tilde{E}_0 \Delta z^{[0]} = \hat{x}_0 - s_0, \quad (5.16b)$$

$$\tilde{E}_{k+1} \Delta z^{[k+1]} = \tilde{D}_k z^{[k]} + \tilde{d}_k, \; k = 0, 1, \ldots, N_b - 1, \quad (5.16c)$$

$$\tilde{C}_k \Delta z^{[k]} \leq -\tilde{c}_k, \; k = 0, 1, \ldots, N_b, \quad (5.16d)$$

The coupling relationship between two consecutive blocks in (5.16c) is defined by $\Delta s_0^{[m+1]} = \Delta s_{N_{pc}}^{[m]}$. The latter quantity can be easily obtained by running Algorithm 5.6 until N_{pc}. $\tilde{H}_k, \tilde{g}_k, \tilde{C}_k, \tilde{D}_k, \tilde{c}_k$ are results of partial condensing with appropriate dimensions and can be obtained by running Algorithms B.1 to B.3 in Appendix B. Note that $\tilde{E}_{k+1} = [\mathbb{I}_{n_x}, \mathbb{O}_{N_{pc} n_u}]$.

Figure 5.1 and 5.2 show three examples on sparsity pattern of the Jacobian of the equality

constraints in (5.16c) and the Hessian matrix, when (5.16) is written in a compact form. By manipulating N_b or N_{pc} across permissible values, the sparsity of the QP problem (5.16) can be varied and the computational cost for solving this QP can be affected (Kouzoupis et al., 2015b). The highest computational complexity of partial condensing is $\mathcal{O}(N_b N_{pc}^2)$ for Algorithm 5.3.

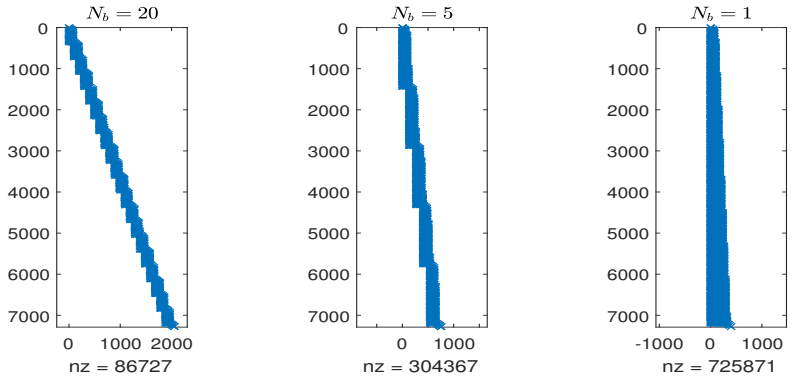


Figure 5.1: Illustrative examples on sparsity pattern of the constraint Jacobian when using partial condensing. From left to right, the number of divided blocks are $N_b = 20, 5, 1$, respectively.

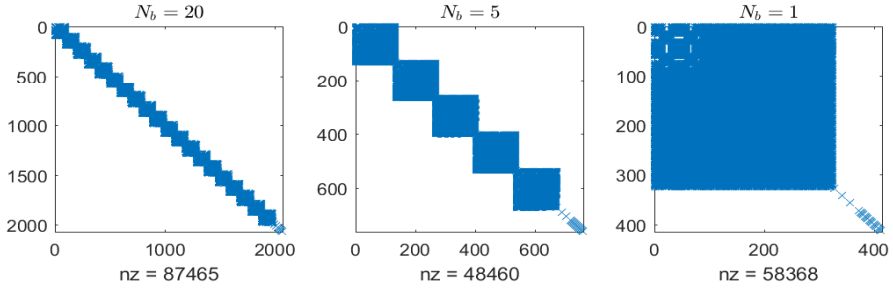


Figure 5.2: Illustrative examples on sparsity pattern of the Hessian when using partial condensing. From left to right, the number of divided blocks are $N_b = 20, 5, 1$, respectively.

5.1.3 CMoN-RTI using Partial Condensing

The CMoN-RTI scheme proposed in Chapter 4 updates part of the sensitivities A_k, B_k in the constraint Jacobian (2.24). In CMoN-RTI, the advantage that can be exploited for condensing is probably that if only the first N_{frac} blocks are updated, i.e. the biggest index of updated sensitivity in CMoN-RTI is N_{frac}, then the last $N - N_{frac}$ block columns of G in (5.6) do not change. If the Hessian does not change, its lower right $(N - N_{frac}) \times (N - N_{frac})$ condensed

Hessian blocks requires no update (Frasch et al., 2012). However, as can be seen in Algorithm 5.2 to 5.4, this fractional update introduced in Mixed-Level Schemes in Chapter 2 has few potential for reducing the computational burden of condensing, since N_{frac} varies significantly from sample to sample and the largest N_{frac} of all sampling instants is usually close to N. In fact, the complete condensing should be re-performed at every sampling instant even if only a part of the sensitivities are updated.

The aforementioned limitation can be overcome by combining CMoN-RTI and partial condensing. The intuition is simple: if all sensitivities in a condensing block do not change, the condensing results G for this block requires no re-computation. If the Hessian does not change, the condensing results $\tilde{H}_k$ for this block is a constant as well. In the following, we first illustrate how to obtain a constant Hessian hence the condensing results for $\tilde{H}_k$ can also benefit from CMoN-RTI.

Constant Hessian

Recall in (2.9) that the following NLP problem has to be solved at every sampling instant:

$$\begin{aligned}
\min_{\mathbf{s},\mathbf{u}} \quad & \sum_{k=0}^{N-1} \phi(t_k, s_k, u_k; p) + \Phi(s_N) && \text{(5.17a)}\\
s.t. \quad & s_0 = \hat{x}_0, && \text{(5.17b)}\\
& s_{k+1} = \Xi(t_k, s_k, u_k; p),\ k = 0, 1, \ldots, N-1 && \text{(5.17c)}\\
& r(s_k, u_k; p) \leq 0,\ k = 0, 1, \ldots, N-1, && \text{(5.17d)}\\
& l(s_N) \leq 0, && \text{(5.17e)}
\end{aligned}$$

The QP problem (5.1) is obtained by linearizing (5.17). Using Gauss-Newton approximation, the Hessian H_k is computed by $H_k = \nabla\bar{\phi}_k^\top \nabla\bar{\phi}_k$ if $\phi_k = \phi(t_k, s_k, u_k; p) = \frac{1}{2}\|\bar{\phi}_k\|_2^2$. If $\bar{\phi}_k$ is a linear function of s_k and u_k, H_k is a constant matrix for all sampling instants. In case $\bar{\phi}_k$ is nonlinear, define $y_k = \bar{\phi}_k$ and augment the system state by

$$\bar{s}_k = [s_k^\top, u_k^\top, y_k^\top]^\top. \tag{5.18}$$

The derivatives of the augmented state is

$$\dot{\bar{s}}_k = [\dot{s}_k^\top, \dot{u}_k^\top, \dot{y}_k^\top]^\top. \tag{5.19}$$

where $\dot{y}_k = \frac{\partial\bar{\phi}_k}{\partial t} = \frac{\partial\bar{\phi}_k}{\partial s_k}\dot{s}_k + \frac{\partial\bar{\phi}_k}{\partial u_k}\dot{u}_k$ and $\bar{u}_k = \dot{u}_k$ is the new control input. The new Hessian is constant since $\bar{\phi}_k$ is a linear function of $\bar{s}_k$ and $\bar{u}_k$. The dimension of the augmented system depends on $\bar{\phi}_k$ and the original number of inputs. Since constant Hessian can be

obtained both in linear and nonlinear objective cases, we can assume a constant Hessian in CMoN-RTI without loss of generality. This means that each block H_k in (2.23) and its sub-blocks Q_k, S_k, R_k in (5.7) are constants throughout on-line optimization. In the following, we assume a linear least square objective function $\phi_k, k = 0, \ldots, N$ for notation simplicity.

Exploit previous condensing blocks

Now let us combine CMoN-RTI and partial condensing. Take the fixed-time block updating logic Ω_1 in Algorithm (3.2) as an example. At every sampling instant, N_f out of N sensitivities are updated. The indexes for these sensitivities are labeled by $(e_1, \ldots, e_{N_f})$. [1] Once given the number of condensing block N_b or the number of consecutive stages in a block $N_{pc} = N/N_b$, the indexes of the condensing blocks which those sensitivities belong to can be computed. Assume

$$e_k = f_k N_{pc} + r_j,\ k = 1, \ldots, N_f,\ r_j = 1, \ldots, N_{pc} \tag{5.20}$$

where f_k is the quotient of e_k divided by N_{pc} and r_k is the remainder. The condensing algorithms are performed only for those blocks of indexes from f_1 to f_{N_f} and the updated sensitivities have indexes $r_j \neq N_{pc}$ in each condensing block. Notice that $f_i = f_j, i \neq j$ is possible. Hence, the maximum number of condensing blocks that should be re-computed is N_f.

For a condensing block $l = 0, \ldots N_b - 1$, the updating strategies from the previous sampling instant have two steps:

1. If some or all sensitivities of indexes from 0 to $N_{pc} - 1$ are updated, the condensing algorithms for this block require completely re-computation. Otherwise, the condensing results $G, \tilde{H}_k, \tilde{C}_k$ can be directly used.

2. The coupling relationship $\tilde{D}_k, \tilde{a}_k$ between two consecutive condensing blocks can be computed by multiplying sensitivities $A_{(l+1)N_{pc}-1}$ to the last row block of G in (5.15).

The complete algorithm is summarized in Algorithm 5.7.

[1] These blocks are originally ordered based on their CMoN values. Here their indexes are ordered monotonically.

Algorithm 5.7 CMoN-RTI using updating logic Ω_1 and partial condensing with complexity $\mathcal{O}(N_f N_{pc}^2)$

1: `// prepare on-line data`
2: Initialize $\mathbf{z}^i$ for $i > 0$
3: Formulate QP problem (2.15), excluding $B(\mathbf{z}^i)$
4: Compute CMoN K_k^i by (3.56) for all $k > 0$
5: `// Apply sensitivity updating logic and partial condensing`
6: Perform Algorithm (3.2)
7: Compute $f_j^i, j = 1, \ldots, N_f$ by (5.20)
8: **for** $l = 0, 1, \ldots, N_b - 1$ **do**
9: Compute $L_l^i, \tilde{g}_l^i, \tilde{c}_l^i$ by Algorithms 5.6, B.2 and B.3
10: **if** $l \in \{f_1^i, \ldots, f_{N_f}^i\}$ **then**
11: Compute G_l^i and $\tilde{H}_l^i, \tilde{C}_l^i$ in (5.15) and (5.16) by looping Algorithms 5.6, B.1 and B.3, until $N_{pc} - 1$
12: **else**
13: $G_l^i \leftarrow G_l^{i-1}, \tilde{H}_l^i \leftarrow \tilde{H}_l^{i-1}, \tilde{C}_l^i \leftarrow \tilde{C}_l^{i-1}$
14: Compute $\tilde{D}_l^i, a_l^i$ by multiplying $A_{(l+1)N_{pc}-1}^i$ to the last row block of G_l^i and L_l^i
15: **end if**
16: **end for**
17: `// Solve the QP problem`
18: Solve QP problem (5.16)

Partial Condensing with a Linear Complexity in Horizon Length

Notice in Algorithm 5.7, the 13th step computes the coupling relationship between two consecutive condensing blocks. The computational complexity of this step is $\mathcal{O}(N_{pc})$. Therefore, it is possible to achieve a linear growth rate of the computational complexity w.r.t. to the number of stages in a block. To make this possible, the first $N_{pc} - 1$ sensitivities in each block has to be unchanged hence no $\mathcal{O}(N_{pc}^2)$ computations are involved. Bearing this idea, a novel strategy named Linear Partial Condensing RTI (LPC-RTI) for determining the sensitivities that should be updated at each sampling instant is proposed.

Choose an appropriate number of condensing blocks $N_b < N$ and compute $N_{pc} = N/N_b > 1$. Define an index set

$$I_{LPC} = \{N_{pc} - 1, 2N_{pc} - 1, \ldots, N_b N_{pc} - 1\}. \tag{5.21}$$

In LPC-RTI, a sensitivity double (A_k, B_k) is updated at each sampling instant if $k \in I_{LPC}$. As a consequence, the dominating partial condensing computation reduces to the 13th step of Algorithm 5.7. There are several remarks regarding this strategy:

1. The sensitivities that should be updated are determined offline and are equally distributed over the prediction window. In this case, the nonlinearity of the system is not taken into account. As the number of condensing blocks increases, there are more sensitivities that are updated and the dynamics of the system is approximated in a higher accuracy.

2. The computationally complexity of partial condensing is $\mathcal{O}(N_b N_{pc}) = \mathcal{O}(N)$. Hence, the complexity is linear in N, independent on the choice of N_b. A trade-off between sensitivity computational cost and the accuracy of dynamics approximation can be expected by varying N_b among permissible values.

This strategy is summarized in Algorithm 5.8.

Algorithm 5.8 LPC-RTI: Linear Partial Condensing RTI with complexity $\mathcal{O}(N)$

// `prepare on-line data`
Initialize $\mathbf{z}^i$ for $i > 0$
Formulate QP problem (2.15), excluding $B(\mathbf{z}^i)$
// `Apply sensitivity updating logic and partial condensing`
for $l = 0, \dots N_b - 1$ **do**
  Compute $L_l^i, \tilde{g}_l^i, \tilde{c}_l^i$ by Algorithms 5.6, B.2 and B.3
  $G_l^i \leftarrow G_l^{i-1}, \tilde{H}_l^i \leftarrow \tilde{H}_l^{i-1}, \tilde{C}_l^i \leftarrow \tilde{C}_l^{i-1}$
  Compute $A^i_{(l+1)N_{pc}-1}$ and $B^i_{(l+1)N_{pc}-1}$
  Compute $\tilde{D}_l^i, a_l^i$ by multiplying $A^i_{(l+1)N_{pc}-1}$ to the last row block of G_l^i and L_l^i
end for
// `Solve the QP problem`
Solve QP problem (5.16)

In Figure 5.3, we show the average CPU time of Algorithm 5.7 for each sampling instant, and the speedup factor w.r.t. the standard partial condensing algorithms presented in Appendix B. The CPU time is measured in a closed-loop simulation starting at the same initial condition. All algorithms are coded in Matlab hence there maybe some software overheads. However, under the same criterion, the comparison among algorithms is fair and the trend shown in Figure 5.3 is demonstrative.

Intuitively, the computational time for block condensing decreases when a larger number of condensing blocks N_b is selected. This is because that the complexity of block condensing scales quadratically with the size of each block but only linearly with the number of condensing blocks. The speedup factor also grows with the number of condensing blocks, since more condensing blocks can be kept unchanged if only a small subset of sensitivities are updated. When $N_b = 2$, there is little speedup since in most cases both blocks contain the updated sensitivities hence are needed to be updated as well.

In Figure 5.4, the same results are depicted when applying Algorithm 5.8. In this case, the number of updated sensitivities is equivalent to the number of condensing blocks. For smaller N_b, the computational time for block condensing and sensitivity evaluation is less, however, the approximation of nonlinear dynamics is worse. Therefore, one may expect a trade-off between the computational efficiency and the optimization accuracy. As demonstrated in Algorithm 5.8, the computational time of LPC-RTI grows linearly with the horizon length N. Considerable speedup is observed when using LPC-RTI, with at least a factor of 2 and up to 14 as shown in the right sub-figure. In addition, the speedup is more significant for larger N, making LPC-RTI more appealing for problems with a long prediction horizon.

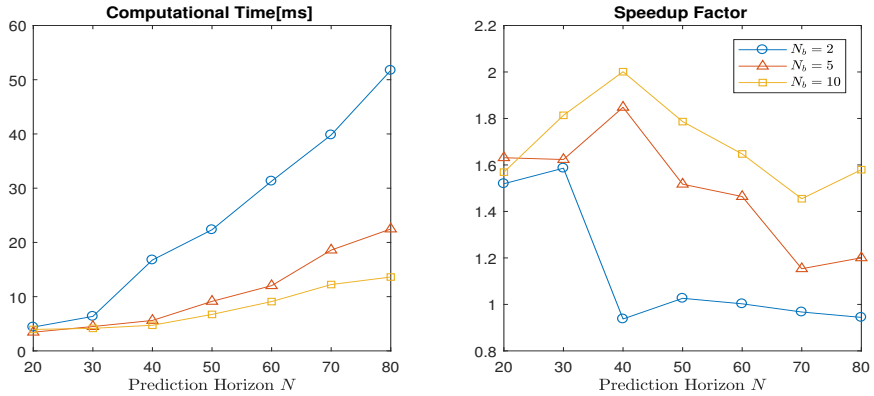


Figure 5.3: The average CPU time[ms] of Algorithm 5.7 for each sampling instant (left), and the speedup factor w.r.t. the standard partial condensing algorithms presented in Appendix B (right).

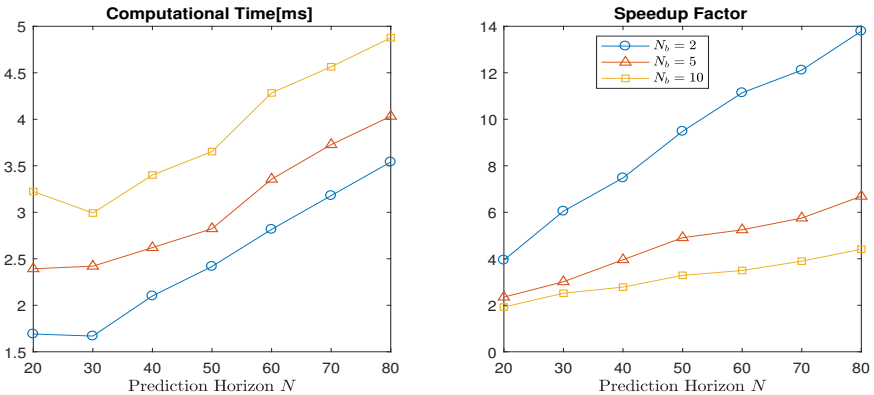


Figure 5.4: The average CPU time[ms] of Algorithm 5.8 for each sampling instant (left), and the speedup factor w.r.t. the standard partial condensing algorithms presented in Appendix B (right).

5.2 Solve sparse QP in CMoN-RTI

The computationally complexity for solving fully condensed QP problems is usually $\mathcal{O}(N^3)$ (Ferreau et al., 2014). Partial condensing results to an equality and inequality constrained QP problem (5.16), which can be solved by sparsity-exploiting solver with a complexity of $\mathcal{O}(N)$, or more specifically, $\mathcal{O}(N_b)$. The most expensive steps for solving the (partially) sparse QP problem are formulating and factorizing the KKT system. State-of-the-art sparsity-exploiting solver, e.g. qpDUNES, has a complexity of $\mathcal{O}(N_b N_{pc}^3)$ for factorizing the Hessian blocks for inner stage QP problems and a complexity of $\mathcal{O}(N_b)$ for factorization in outer Newton iterations (Kouzoupis et al., 2015b). ForcesPro has a complexity of $\mathcal{O}(N_b)$ for factorizing the KKT system (Domahidi, Zgraggen, Zeilinger, Morari, and Jones, 2012). However, the discussion by now does not taken into account the partially updated sensitivities when formulating the QP problem. In fact, either qpDUNES or ForcesPro require a complete re-factorization when part or all the sensitivities are updated. In this section, we show that in CMoN-RTI, the sparse QP problem can be solved via splitting methods with a complexity of $\mathcal{O}(N_f)$.

5.2.1 Splitting Methods

Operator splitting (OP) techniques can solve convex optimal control problems very fast with a moderate solution accuracy. The most popular OP method is probably the alternating direction method of multipliers (ADMM) (Boyd et al., 2011). In (O'Donoghue et al., 2013), a tailored ADMM algorithm is developed for solving optimal control problems. This algorithm breaks the original problem into two parts, a quadratic part which can be solved via on-line optimization, and a set of non-convex part which can be solved in parallel or sometimes analytically.

Consider the fully sparse QP problem (5.2). The partially condensed QP problem (5.16) can be seen as a shrunken problem of (5.2). Move the inequality constraint into objective we obtain

$$\min_{\Delta \mathbf{z}} \quad \psi(\Delta \mathbf{z}) + I_{\mathcal{C}}(\Delta \mathbf{z}) \tag{5.22a}$$

$$s.t. \quad \Delta \mathbf{z} \in \mathcal{D} \tag{5.22b}$$

where the quadratic objective is

$$\psi(\Delta \mathbf{z}) = \sum_{k=0}^{N} \frac{1}{2} \Delta z_k^\top H_k \Delta z_k + g_k^\top \Delta z_k, \tag{5.23}$$

and

$$I_{\mathcal{C}}(\Delta \mathbf{z}) = \begin{cases} 0, \ \Delta \mathbf{z} \in \mathcal{C} \\ \infty, \ \text{otherwise}, \end{cases} \tag{5.24}$$

is an indicator function over a closed convex set

$$\mathcal{C} = \{\Delta z_k | C_k \Delta z_k \leq -c_k, k = 0, \ldots, N\}. \tag{5.25}$$

System dynamics are imposed in the set

$$\mathcal{D} = \{\Delta \mathbf{z} | E_0 \Delta z_0 = \hat{x}_0 - s_0, E_{k+1} \Delta z_{k+1} = D_k z_k + a_k, \ k = 0, 1, \ldots, N-1\}. \tag{5.26}$$

Such a definition ensures the equivalence of problem (5.22) and (5.2), since the inequality constraints are imposed in the objective function and will be fulfilled after optimization. Splitting methods break problem (5.22) into two parts as

$$\min_{\Delta z, \Delta \hat{\mathbf{z}}} \quad \psi(\Delta \mathbf{z}) + I_{\mathcal{D}}(\Delta \mathbf{z}) + I_{\mathcal{C}}(\Delta \hat{\mathbf{z}}) \tag{5.27a}$$

$$s.t. \quad \Delta \mathbf{z} = \Delta \hat{\mathbf{z}}, \tag{5.27b}$$

where $\Delta \hat{\mathbf{z}}$ is a duplicate of the original optimization variable $\Delta \mathbf{z}$. This is the typical consensus form of ADMM algorithms and the equivalence between problem (5.22) and (5.27) is shown in (O'Donoghue et al., 2013; Boyd et al., 2011). The formulation (5.27) can be solved via a standard ADMM algorithm, which is summarized in Algorithm 5.9.

Algorithm 5.9 ADMM for solving problem (5.27)

1: Initialize $\Delta \hat{\mathbf{z}}^0$, ν^0 for iteration $i = 0$
2: **for** i=0,1,... **do**
3: $\quad \Delta \mathbf{z}^{i+1} = \text{argmin}(\psi(\Delta \mathbf{z}) + I_{\mathcal{D}}(\Delta \mathbf{z}) + \frac{\rho}{2} \|\Delta \mathbf{z} - \Delta \hat{\mathbf{z}}^i - \nu^i\|_2^2)$
4: $\quad \Delta \hat{\mathbf{z}}^{i+1} = \text{argmin}(I_{\mathcal{C}}(\Delta \hat{\mathbf{z}}) + \frac{\rho}{2} \|\Delta \hat{\mathbf{z}} - \Delta \mathbf{z}^{i+1} + \nu^i\|_2^2)$
5: $\quad \nu^{i+1} = \nu^i + \Delta \hat{\mathbf{z}}^{i+1} - \Delta \mathbf{z}^{i+1}$
6: **end for**

In Algorithm 5.9, $\rho > 0$ is an algorithm parameter and ν is the Lagrangian multiplier for the consensus constraint (5.27b). The second step in Algorithm 5.9 is separable across time, hence we can replace it by

$$\Delta \hat{z}_k^{i+1} = \text{argmin}(I_{\mathcal{C}_k}(\Delta \hat{z}_k) + \frac{\rho}{2} \|\Delta \hat{z}_k - \Delta z_k^{i+1} + \nu_k^i\|_2^2), \ k = 0, 1, \ldots, N-1. \tag{5.28}$$

This step also refers to *proximal minimization*. If $\mathcal{C}$ is a box and is separable across components, then the minimum of (5.28) can be computed analytically by using saturation functions. If $\mathcal{C}$

is not a simple set, the proximal minimization can be solved via a general numerical method, such as active-set and interior point method. The biggest advantage of ADMM methods is that the proximal minimization can be performed in parallel on multiple processors. The convergence of Algorithm 5.9 is shown in (O'Donoghue et al., 2013; Boyd et al., 2011).

5.2.2 CMoN-RTI using Partial Block Update

The most computationally expensive part of Algorithm 5.9 is the first step, which requires to solve the QP problem (5.2) without inequality constraints, written as

$$\min_{\Delta \mathbf{z}} \quad \sum_{k=0}^{N} \frac{1}{2} \Delta z_k^\top \hat{H}_k \Delta z_k + \hat{g}_k^\top \Delta z_k \tag{5.29a}$$

$$s.t. \quad E_0 \Delta z_0 = \hat{x}_0 - s_0, \tag{5.29b}$$

$$E_{k+1} \Delta z_{k+1} = D_k z_k + a_k,\ k = 0, 1, \ldots, N-1, \tag{5.29c}$$

where $\hat{H}_k = H_k + \rho \mathbb{I}_{n_x+n_u}$, $\hat{g}_k = g_k - \rho(\Delta \hat{z}_k + v_k)$. A compact form as (2.15) can be written as

$$\min_{\Delta \mathbf{z}} \quad \frac{1}{2} \Delta \mathbf{z}^\top \hat{H} \Delta \mathbf{z} + \hat{g}^\top \Delta \mathbf{z} \tag{5.30a}$$

$$s.t. \quad b(\mathbf{z}^i) + B(\mathbf{z}^i) \Delta \mathbf{z} = 0, \tag{5.30b}$$

where $\hat{H} = \text{blkdiag}(\hat{H}_0, \ldots, \hat{H}_N)$, $\hat{g} = [\hat{g}_0^\top, \ldots, \hat{g}_N^\top]^\top$. Problem (5.30) can be solved by solving its KKT system, expressed as

$$\begin{bmatrix} \hat{H} & B^\top \\ B & \end{bmatrix} \begin{bmatrix} \Delta \mathbf{z} \\ \lambda \end{bmatrix} = - \begin{bmatrix} \hat{g} \\ b \end{bmatrix}. \tag{5.31}$$

(O'Donoghue et al., 2013) suggest to use sparse $LDL^\top$ factorization to solve the linear system (5.31). While this method is quite efficient, there is a major shortcoming when applying it in NMPC or time-varying problems. In these cases, $LDL^\top$ factorization has to be re-computed at every sampling instant which significantly increases the computational effort. As a result, splitting methods or ADMM are usually employed for linear MPC problems, where the KKT system (5.31) is time-invariant and its factorization can be computed offline.

An alternative way is to eliminate the primal variable $\Delta \mathbf{z}$ by

$$\Delta \mathbf{z} = -\hat{H}^{-1}(B^\top \lambda + \hat{g}), \tag{5.32}$$

which results to the most compact form, often called *normal equations*,

$$Y\lambda = \beta, \tag{5.33}$$

where $Y = B\hat{H}^{-1}B^\top, \beta = b - B\hat{H}^{-1}\hat{g}$. Due to block banded structure of $\hat{H}$ and B, Y is symmetric block tri-diagonal. In the following, we show that in CMoN-RTI, Y can be efficiently factorized using partial block-wise Cholesky factorization in the scenario of NMPC.

Calculating Y

Y has a symmetric block-banded structure (Domahidi et al., 2012):

$$Y = \begin{bmatrix} Y_{11} & Y_{12} & & & & \\ Y_{12}^\top & Y_{22} & Y_{23} & & & \\ \vdots & \ddots & \ddots & & & \\ & & Y_{N-2,N-1}^\top & Y_{N-1,N-1} & Y_{N-1,N} \\ & & & Y_{N-1,N}^\top & Y_{N,N} \end{bmatrix}, \tag{5.34}$$

where

$$Y_{k,k} = D_{k-1}\hat{H}_{k-1}^{-1}D_{k-1}^\top + E_k\hat{H}_k^{-1}E_k^\top, \tag{5.35}$$
$$Y_{k,k+1} = -E_k\hat{H}_{k-1}^{-1}D_k^\top. \tag{5.36}$$

To compute Y, (Domahidi et al., 2012) suggest to use a rectangular factorization expressed by

$$\hat{H}_k = \hat{L}_k\hat{L}_k^\top, \tag{5.37}$$
$$V_k\hat{L}_k^\top = D_k, \tag{5.38}$$
$$W_k\hat{L}_k^\top = E_k, \tag{5.39}$$
$$Y_{k,k} = V_{k-1}V_{k-1}^\top + W_{k-1}W_{k-1}^\top, \tag{5.40}$$
$$Y_{k,k+1} = W_kV_k^\top, \tag{5.41}$$

where $\hat{L}_k$ is the Cholesky factor of $\hat{H}_k$ and V_k, W_k can be solved by matrix forward substitution. The Flops for computing (5.37) to (5.41) in two different scenarios is compared in Tabel 5.1: in an interior point method exploited by (Domahidi et al., 2012) and in the proposed splitting method with partially updated sensitivities when employing CMoN-RTI.

Table 5.1: Computational cost in Flops for computing (5.37) to (5.41) for all stages in (5.29)

Equation	(Domahidi et al., 2012)	Proposed
(5.37)	$\frac{2}{3}N(n_x+n_u)^3$	0
(5.38)	$N(n_x+n_u)^2 n_x$	$N_f(n_x+n_u)^2 n_x$
(5.39)	$N(n_x+n_u)^2 n_x$	0
(5.40)	$2N(n_x+n_u)n_x^2$	$N_f(n_x+n_u)n_x^2$
(5.41)	$2N(n_x+n_u)n_x^2$	$2N_f(n_x+n_u)n_x^2$

The proposed method can perform the steps (5.37) and (5.39) completely offline. This is because $\hat{H}_k$ is constant given a constant Hessian H_k and a constant splitting parameter ρ. The Cholesky factor $\hat{L}_k$ is a constant. In addition, $E_k = [\mathbb{I}_{n_x}\mathbb{O}]$ is a constant matrix, hence (5.39) as well as the second part of (5.40) can also be computed offline. However, using the original interior point method as in (Domahidi et al., 2012), $\hat{H}_k$ is time-varying or iteration varying even if the Hessian H_k is a constant, since $\hat{H}_k$ contains the Lagrangian multipliers associated with inequality constraints, which vary at every iteration. The splitting method eliminates inequality constraints when computing the normal equation (5.33).

On the other hand, in CMoN-RTI the sensitivities D_k are partially updated for $k = 0,1,\ldots,N-1$. Hence only N_f out N equations like (5.38) are needed to be solved. Specifically, for sensitivities $D_r, r>0$ not being updated, $Y_{r+1,r+1}$ and $Y_{r,r+1}$, i.e. the $(r+1)$th block column of Y does not need to be re-computed. For D_0 not being updated, Y_{11} does not change. As a result, the computational cost for (5.38), (5.40) and (5.41) are considerably saved, especially when $N_f \ll N$.

Factorizing Y

The Cholesky factor of Y is

$$L_Y = \begin{bmatrix} L_{11} & & & & \\ L_{21} & L_{22} & & & \\ & L_{32} & & & \\ \vdots & \vdots & \ddots & & \\ & & & L_{N,N-1} & L_N \end{bmatrix}, \tag{5.42}$$

where the blocks $L_{i,j}$ are computed by (Wang and Boyd, 2010)

$$Y_{11} = L_{11}L_{11}^{\top}, \tag{5.43}$$
$$Y_{i,i+1} = L_{i,i}L_{i+1,i}^{\top},\ i = 1,\ldots,N, \tag{5.44}$$
$$Y_{i,i} - L_{i,i-1}L_{i,i-1}^{\top} = L_{i,i}L_{i,i}^{\top},\ i = 2,\ldots,N. \tag{5.45}$$

Note that L_Y is computed sequentially. First, L_{11} is the Cholesky factor of Y_{11}. Second, L_{21} is computed by matrix forward substitution. Finally, L_{22} is the Cholesky factor of $Y_{22} - L_{21}L_{21}^{\top}$. These steps are continuously performed up to $i = N$. Therefore, computational cost usually cannot be saved even if part of the Y blocks are constants. However, as indicated in (Domahidi et al., 2012), the computational cost for factorizing Y is only half of that for computing Y. By employing the proposed method stated in the previous subsection, the overall computational cost for solving the normal equation (5.33) can be significantly reduced.

Remarks

We have shown that how the KKT matrix from previous sampling instants can be re-used by employing splitting methods and block factorization. The Flops needed to compute the KKT matrix has been reduced as shown in Table 5.1. At this stage, we make some remarks on the proposed scheme, comparing to the original interior point based algorithms.

1. Splitting methods usually require more iterations to achieve a given accuracy than interior point methods. In our case, in order to have a time-invariant Hessian, the algorithm parameter ρ is chosen constant. This may lead to a slow convergence rate or even cause convergence problems.

2. Interior point methods can usually achieve a given accuracy using just a few iterations, while the computation for each iteration is relatively expensive.

3. There are two kinds of ideas for solving the convex QP problem (5.2). One is represented by interior point method which take fewer expensive iterations. The other is represented by splitting methods which take more cheaper iterations. The proposed scheme employs even cheaper computations at each sampling instant than the standard splitting methods by exploiting the structure of the OCP and the partial sensitivity update schemes.

6

MATMPC: a MATLAB-based Nonlinear MPC package

After decades of development, many open source MPC software or packages are available for simulation and experiment purposes. The most popular packages include MATLAB Model Predictive Control Toolbox (Bemporad, Morari, and Ricker, 2010) and MPT3 (Herceg, Kvasnica, Jones, and Morari, 2013). An excellent book by (Wang, 2009) gives comprehensive coding instructions for MPC, which makes the implementation of MPC rather intuitive. However, there is not much nonlinear MPC software that on the one hand, is easy to use and debug, and on the other hand, is computationally efficient. The difficulty of upgrading MPC implementations to NMPC ones rises considerably mainly due to the difficulty of efficient linearization and the complexity of non-convex optimization. Among the available NMPC software, ACADO (Houska et al., 2011) automatically generates optimized c codes for the RTI scheme (Diehl et al., 2002) introduced in Chapter 2, using a user-friendly symbolic interface in MATLAB and C++. All types of objectives and constraints are supported in ACADO. VIATOC (Kalmari, Backman, and Visala, 2015) also generates c codes for box constrained problems using gradient projection methods to solve NLPs on-line. In addition to the code generation tools, other software exploit general NLP solvers to solve online optimization problems. For example, CasADi (Andersson, 2013) is mainly a symbolic differentiation tool and embeds many NLP and QP solvers such as IPOPT (Wächter and Biegler, 2006) and SNOPT (Gill et al.,

2005). Forces Pro (Domahidi and Jerez, 2014) is freely available for academic users, which employs CasADi for efficient derivative computations and generates state-of-the-art interior point methods to solve multi-stage NLPs that are usually observed in NMPC.

The problems of aforementioned software are manifold:

1. The NMPC algorithms are automatically generated which can not be modified or investigated.
2. It is difficult to debug software and algorithms without a professional knowledge in computer programming.
3. The type of supported algorithms and functions are limited.

The reason for these problems is that, the available software aims at providing ready-to-use interfaces for engineers who know little about mathematical algorithms. The task of users is to tune model and software parameters to make algorithms work. There are few things that an algorithm developer can do to improve or augment the software unless he/she participates in the developing team.

Therefore, the author of this thesis believes that it is necessary to develop a NMPC package that aims at helping algorithm developer. This package should be fully open-source and commented at the level of algorithms. To this end, we have developed MATMPC, a NMPC package completely based on MATLAB, which is probably the most frequently used coding environment by control engineers. Comparing with the aforementioned software, MATMPC has the following features.

- MATLAB codes are much easier for debugging and maintaining than C or C++ codes. MATLAB provides a user-friendly environment for algorithmic coding. It is easy and straightforward to announce new variables, to use matrix and vector operations and to check type/dimensions. There is no need to worry about memory management and pointers, which often cause problems in C/C++.
- Almost all numerical linear algebra routines can be found in MATLAB. Efficient numerical linear algebra are supported in MATLAB, e.g. matrix multiplication, concatenation and factorization. The users can employ these mature routines or overwrite them with their self-developed routines. In particular, sparse linear algebra routines are well supported in MATLAB, which can significantly improve the efficiency of algorithms for large-scale problems.
- MATLAB has C/C++ API that can improve the computational efficiency of algorithms. For real-time applications, C codes can be hand coded or automatically generated from MATLAB builtin M files. A knowledge of C/C++ programming is not necessary.

MATMPC has two working modes: one works in pure MATLAB which aims at algorithm developing and debugging, and the other employs mexFunction, which is coded by hand or generated automatically from M files, which aims at computational efficiency for real-time applications. MATMPC is freely available on https://github.com/chenyutao36/MATMPC.

6.1 MATLAB Mode

The MATLAB Mode works only in MATLAB environment, with all the routines built in MATLAB available. The structure of MATMPC is illustrated in Figure 6.1.

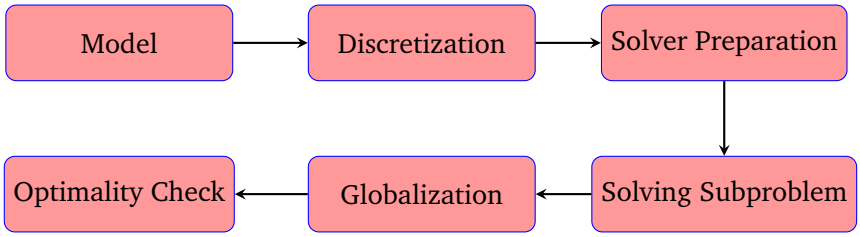


Figure 6.1: Structure of MATMPC.

Model

MATMPC takes a continuous-time dynamic model as default. The modelling language is based on CasADi (Andersson, 2013). Almost all kinds of expressions are supported. As a result, the objective function, the dynamic equation and the constraints are expressed by CasADi functions, which will be called later when running NMPC algorithms. An code example of the inverted pendulum 1.1 is shown.

```
%-------------------------------------------%
% This file contains the model
%-------------------------------------------%

%% Dimensions

nx=4;  % No. of states
nu=1;  % No. of controls
ny=5; % No. of outputs
nyN=4; % No. of outputs at the terminal point
np=0; % No. of model parameters
nc=2; % No. of constraints
ncN=1; % No. of constraints at the terminal point

%% Variables
```

```
16
17 import casadi.*
18
19 states   = SX.sym('states',nx,1);
20 controls = SX.sym('controls',nu,1);
21 params   = SX.sym('paras',np,1);
22 refs     = SX.sym('refs',ny,1);
23 refN     = SX.sym('refs',nyN,1);
24 Q        = SX.sym('Q',ny,ny);
25 QN       = SX.sym('QN',nyN,nyN);
26
27 %% Dynamics
28
29 M = 1;
30 m = 0.1;
31 l = 0.8;
32 g = 9.81;
33
34 p=states(1);
35 theta=states(2);
36 v=states(3);
37 omega=states(4);
38 u=controls(1);
39
40 a=-m*l*sin(theta)*omega^2+m*g*cos(theta)*sin(theta)+u;
41 b=-m*l*cos(theta)*omega^2+u*cos(theta)+(M+m)*g*sin(theta);
42 c=M+m-m*(cos(theta))^2;
43
44 x_dot=[v;omega;a/c;b/(l*c)];
45
46 f  = Function('f', {states,controls,params}, {x_dot},{'states','controls','
     params'},{'xdot'});
47
48 %% Objectives and constraints
49
50 h = [p;theta;v;omega;u];
51
52 hN = h(1:nyN);
53
54 h_fun=Function('h_fun', {states,controls,params}, {h},{'states','controls','
     params'},{'h'});
55 hN_fun=Function('hN_fun', {states,params}, {hN},{'states','params'},{'hN'});
56
57 ineq=[p;u];
58 ineqN=p;
```

```
ineq_fun=Function('ineq_fun', {states,controls,params}, {ineq},{'states','
    controls','params'},{'ineq'});
ineqN_fun=Function('ineqN_fun', {states,params}, {ineqN},{'states','params'},{'
    ineqN'});

lb_ineq=SX.sym('lb_ineq',length(ineq),1);
ub_ineq=SX.sym('ub_ineq',length(ineq),1);
lbN_ineq=SX.sym('lbN_ineq',length(ineqN),1);
ubN_ineq=SX.sym('ubN_ineq',length(ineqN),1);
```

Listing 6.1: MATLAB code example for the model of inverted pendulum

Discretization

The continuous-time optimization problem is discretized using direct multiple shooting (See Chapter 2) and direct collocation (Von Stryk, 1993). This involves a for-loop where functions generated by the model are repeatedly called. Take multiple shooting as an example. An numerical integrator has to be constructed with the ability of sensitivity computation. Users can employ CasADi for symbolically building such an integrator and generate a function for later use. An example of an explicit 4^{th} order Runge-Kutta integrator is shown below.

```
s  = 1; % No. of integration steps per shooting interval
DT = Ts/s; % Length of one integration step
X=states;
U=controls;
P=params;
z = [states;controls];

% explicit 4-th order Runge-Kutta integrator
for j=1:s
    k1 = f(X, U, P);
    k2 = f(X + DT/2 * k1, U, P);
    k3 = f(X + DT/2 * k2, U, P);
    k4 = f(X + DT * k3, U, P);
    X=X+DT/6*(k1 +2*k2 +2*k3 +k4);
end
F = Function('F', {z,params}, {X}, {'z','params'}, {'xf'}); % integration
    results
D= F.jacobian('z','xf'); % sensitivity w.r.t. initial states and controls
```

Listing 6.2: MATLAB code example for an explicit 4^{th} order Runge-Kutta integrator

Solver Preparation

Once the ingredients for a discretized NLP problem (or QP problem) are ready, one may need to prepare them for a specific solver. For a SQP method, any QP solvers can be employed, e.g. qpOASES (Ferreau et al., 2014), Ipopt (Wirsching et al., 2006) and Forces Pro (Domahidi and Jerez, 2014). However, they require different formulation of QP problems. For example, qpOASES solves a condensed QP problem while Forces Pro and qpDUNES (Frasch et al., 2015) directly solve a multi-stage one. Ipopt requires all inputs as sparse variables. Therefore, solver preparations are implemented in MATMPC. For condensing, full, partial (described in Chapter 5) and null-space (Diehl et al., 2010) condensing algorithms are implemented. Matrix multiplication and factorization are naturally supported in MATLAB hence these implementations are intuitive.

Solving Subproblem

To solve a discretized NLP problem like (2.10), an iterative algorithm is usually desired. In case SQP is used, QP subproblems can be solved by aforementioned solvers. If interior point method is used, users can either employ existing interior point solver or implement their own algorithms. Linear systems can be efficiently solved in MATLAB. First order methods can be efficiently implemented and can be run in parallel using MATLAB's parallel toolbox.

Globalization

Globalization techniques are desired to find local minimum from arbitrarily initial point. In MATMPC, efficient line search algorithms using merit functions (Nocedal and Wright, 2006) and filters (Wirsching et al., 2006) are implemented. These algorithms require function evaluations and derivative computations provided by CasADi and can be easily implemented in MATLAB. Users can also employ their own globalization algorithms written in MATLAB.

Optimality Check

The constraint residuals and the first order optimality condition (KKT value) are evaluated at the end of each iteration. In particular, KKT value, which is the first order derivative of the Lagrangian of the NLP problem, can be efficiently computed by backward mode of automatic differentiation provided by CasADi toolbox.

6.2 MEX Mode

A MATLAB Executable (often called MEX) is a way to call customized C or Fortran codes directly from MATLAB as if it is a MATLAB built-in function. Computational efficiency can be significantly improved by re-writing bottleneck computations as MEX. There are two ways of generating these MEX functions:

1. Writing customized MEX using the MATLAB gateway function
2. Automatically generating MEX from M files using MATLAB Coder

MEX Function

The gateway routine *mexFunction* is an interface of C code with MATLAB inputs and outputs, where a computational routine performs the computations users require. Replacing computationally extensive steps (e.g. for-loop) in M files by an equivalent MEX function is recommended to accelerate the computation.

The biggest advantage of using MEX is that MATLAB offers a C Matrix Library API that deals with matrix construction, deletion and access. This is extremely useful for developing optimization algorithms that employ considerable vector and matrix operations. In addition, struct and cell arrays are also supported by MEX.

Numerical linear algebra libraries like Lapack and Blas (Anderson, Bai, Bischof, Blackford, Demmel, Dongarra, Du Croz, Greenbaum, Hammarling, McKenney, et al., 1999) are included in MEX libraries and are employed by MATMPC when working in MEX mode. Other efficient implementation of linear algebra, e.g. openblas (Xianyi, Qian, and Saar, 2016), Intel MKL (Wang, Zhang, Shen, Zhang, Lu, Wu, and Wang, 2014) can also be used as dynamic libraries.

An example of coding Algorithm 5.1 is shown below, where the inputs are cell arrays.

```
...

/* compute G */
    for(i=0;i<N;i++){
        cell_element = mxGetCell(prhs[1], i);
        cell = mxGetPr(cell_element); /* Bi */
        Block_Fill(i,i,cell,nx,nu,G,N);

        for (j=i+1;j<N;j++){
            cell_element = mxGetCell(prhs[0], j);
            cell = mxGetPr(cell_element); /* Ai */
            Block_Access(j-1,i,Gi,nx,nu,G,N);
            dgemm(nTrans, nTrans, &nx, &nu, &nx, &one_d, cell, &nx, Gi, &nx, &
    zero, Ci, &nx);
```

```
            Block_Fill(j,i,Ci,nx,nu,G,N);
        }
    }

    /* compute L */
    for(i=0;i<N;i++){
        cell_element = mxGetCell(prhs[0], i);
        cell=mxGetPr(cell_element); /* Ai */
        memcpy(&L[i*nx],&ai[0], nx*sizeof(double));
        memcpy(&ai[0],&a[i*nx], nx*sizeof(double));
        memcpy(&Li[0],&L[i*nx], nx*sizeof(double));
        dgemv(nTrans,&nx,&nx,&one_d,cell,&nx,Li,&one_i,&one_d,ai,&one_i);
    }
    memcpy(&L[N*nx],&ai[0], nx*sizeof(double));

...
```

Listing 6.3: C code for condensing in MEX mode

Automatic Code Generation

MATLAB Coder generates readable and portable C and C++ code from MATLAB code. It supports most of the MATLAB language and a wide range of toolboxes. One can integrate the generated code into his/her projects as source code, static libraries, or dynamic libraries. One can also use the generated code within the MATLAB environment to accelerate computationally intensive portions of his/her MATLAB code. Therefore, algorithm developers can relieve themselves from coding or programming, but instead focusing on mathematics.

CasADi toolbox can also automatically generate C codes for its functions. MATMPC can call CasADi functions directly when working in MATLAB mode or their C counterparts when in MEX mode. Significant improvement can be observed by running C codes that implement automatic differentiation. These C codes can be compiled into MEX functions or dynamic libraries, which can be employed by external C applications.

6.3 Applications

MATMPC has been used for implementing and examining the proposed algorithms in this thesis. In addition, MATMPC has been used for simulations of many applications, including the benchmark problem inverted pendulum 1.1 and the chain of masses connected by linear 1.6 and nonlinear 4.33 springs. Applications for dynamic driving simulators are also presented (Details are presented in Chapter 7).

7

Applications to Driving Simulator and Motion Cueing

In this Chapter, we focus on non-trivial applications in automotive industry, based on fast NMPC algorithms introduced and proposed in previous chapters. The first application is to control a nine degree of freedom (DOF) dynamic driving simulator described in (Bruschetta et al., 2017c). The goal is to achieve real-time implementations of a NMPC-based motion cueing algorithm (MCA), for the tracking of driver perceived sensations. Implementations based on MATMPC is able to achieve a 100Hz control frequency. The second application is to implement a Multi-Sensory Cueing Algorithm (MSCA), proposed for a high performance driving simulator with active seat (Bruschetta et al., 2017b). The control frequency can be up to 200Hz.

7.1 Nine DOF Driving Simulator

The use of dynamic driving simulator has been increasingly popular in the automotive community, both in the research and industrial fields. Simulator platforms with different mechanical structures have been designed to target particular applications and markets. Such platforms are responsible for reproducing the driver sensations faithfully within their limited working space, while the mechanical constraints have to be satisfied to avoid hazardous

situations (Baseggio, Beghi, Bruschetta, Maran, and Minen, 2011; Beghi, Bruschetta, and Maran, 2012). The strategy for the motion control of platforms are hence called *moting cueing algorithms* (MCA).

MCAs based on linear MPC techniques are well developed for the control of a linear actuated six DOF platforms (Baseggio et al., 2011; Beghi et al., 2012). Human vestibular models and platform constraints are explicitly taken into account, and optimal control techniques are employed so that the tuning of the MCA is intuitive. Comparing with classical schemes based on washout filters, MPC-based MCAs allow for a better exploitation of the working space and a straightforward implementation of tilt coordination.

However, nonlinearities are introduced by using more complex actuators and exploiting more DOFs. Therefore, NMPC-based MCAs are good candidates for providing faithful driver perceptions while considering nonlinear dynamics and constraints. A nine DOF driving simulator platform DiM 150 has been described in (Bruschetta et al., 2017c). The mechanical structure of the simulator consists of a hexapod mounted on a tripod, which moves on a flat, stiff surface sliding on airpads. Such a structure allows for both low and high frequency of movements. The tripod is able to produce most of the low frequency longitudinal, lateral and yaw movements within a relatively large working space, while the hexapod provides high frequency longitudinal, lateral and yaw movements as well as pitch and roll rotations. See Figure 7.1 for a sketch of the device and Figure 7.2 for the real picture.

The complete dynamic model of DiM 150 consists of a human vestibular model that is characterized by a state-space realization (Beghi et al., 2013; Maran, 2013) and a reference transition model.

Human Vestibular Model

The human vestibular system is located in the inner ear and is composed by the semicircular canals and the otolith organs. The former can sense the angular rotations and the latter linear motion. We take the transfer functions given in (Baseggio et al., 2011) for semicircular and otolith as

$$W_{scc} = \frac{\hat{\omega}(s)}{\alpha(s)} = 5.73\frac{80s^2}{(1+80s)(1+5.73s)}, \tag{7.1}$$

$$W_{oth} = \frac{\hat{f}(s)}{f(s)} = 0.4\frac{1+10s}{(1+5s)(1+0.016s)}, \tag{7.2}$$

where $\hat{\omega}(s)$ is the sensed angular velocity, $\alpha(s)$ the acceleration stimulus, $\hat{f}(s)$ the sensed force stimulus and $f(s)$ the actual force stimulus. These transfer functions are converted to state-space realizations and the human vestibular model can be represented by $\Sigma_V(\hat{A},\hat{B},\hat{C},\hat{D})$,

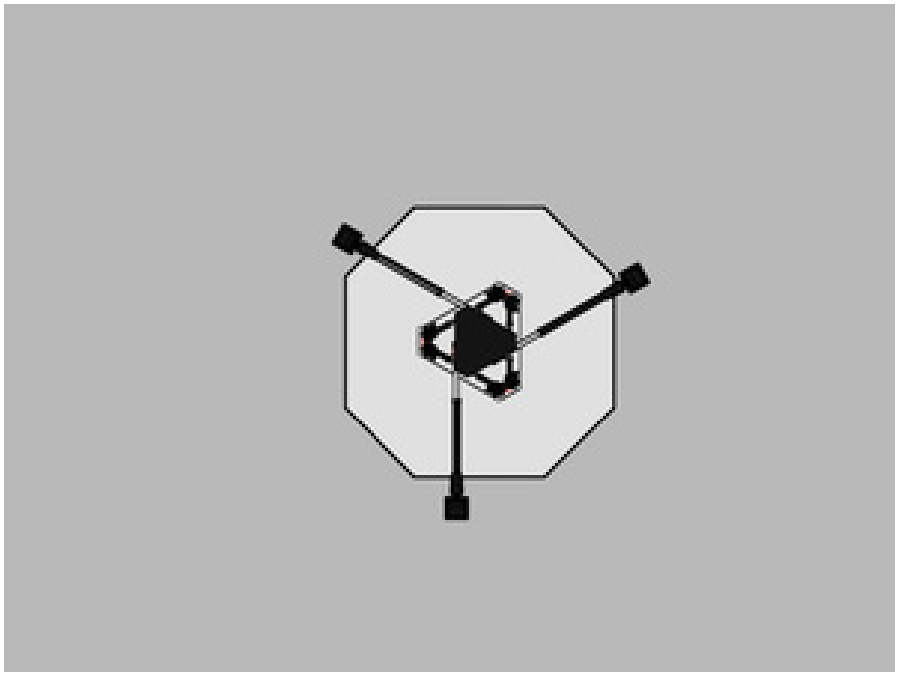


Figure 7.1: Top view sketch of the nine DOF driving simulator DiM 150

Figure 7.2: Real photo for the nine DOF driving simulator DiM 150

where

$$\hat{A} = \text{blkdiag}(\hat{A}_S, \hat{A}_O), \tag{7.3}$$
$$\hat{B} = \text{blkdiag}(\hat{B}_S, \hat{B}_O), \tag{7.4}$$
$$\hat{C} = \text{blkdiag}(\hat{C}_S, \hat{C}_O), \tag{7.5}$$
$$\hat{D} = \text{blkdiag}(\hat{D}_S, \hat{D}_O), \tag{7.6}$$

are state space matrices consisting of semicircular and otolith systems, respectively. The input and output of Σ_V are

$$u_V = [a_x, a_y, a_z, \dot{\psi}, \dot{\theta}, \dot{\phi}]^\top, \tag{7.7}$$
$$y_V = [\hat{a}_x, \hat{a}_y, \hat{a}_z, \hat{\dot{\psi}}, \hat{\dot{\theta}}, \hat{\dot{\phi}}]^\top, \tag{7.8}$$

where a_x, a_y, a_z are the actual longitudinal, lateral,and vertical accelerations and ψ, θ, ϕ are roll, pitch and yaw angles. The symbol $\hat{\cdot}$ represents the corresponding perceived signals. Since otoliths are not capable to discriminate between gravitational and longitudinal forces (Baseggio et al., 2011), the platform can tilt up to a certain angle to provide sustained accelerations. This strategy is called tilt coordination.

Reference Transition Model

There are four reference frames for the simulator platform:

1. The ground or the world frame $\{X_G, Y_G, Z_G\}$ that is fixed and static. Its origin is defined as O.
2. The tripod frame $\{X_T, Y_T, Z_T\}$ moves with its bottom disk on the $X-Y$ plane on the ground. The origin O_T is at the center of the bottom disk.
3. The hexapod frame $\{X_H, Y_H, Z_H\}$ moves with its top disk on the top of the tripod. The origin O_H is at the center of the top disk.
4. The drivers frame $\{X_D, Y_D, Z_D\}$ has a origin O_D located at the brain of the driver where the vestibular system lies in. This frame moves and rotates along with the hexapod frame.

Define the vector p_H^T as the translation vector between O_T and O_H, in the tripod coordinates (the reference frame is specified by the superscript). It follows that

$$p_H^G = p_T^G + R_T p_H^T, \tag{7.9}$$

where R_T is the 3×3 rotation matrix for the tripod using the classical $z-y-x$ convention. As a result, by taking twice differentiation of (7.9), the acceleration relationship is given by

$$\ddot{p}_H^G = \ddot{p}_T^G + R_T \ddot{p}_H^T + 2\omega_T \times R_T \dot{p}_H^T + \omega_T \times \omega_T \times R_T p_H^T + \dot{\omega}_T \times R_T p_H^T, \tag{7.10}$$

where ω_T is the instantaneous rotational velocity of the tripod and $\dot{R} = \omega \times R$. The symbol $\times$ denotes *cross product*. Since the tripod only has rotation around the vertical axis, $R_T = R_z(\phi_T)$.

Similarly, we have

$$\ddot{p}_D^G = \ddot{p}_H^G + R_{T,H} \ddot{p}_D^H + 2\omega_{T,H} \times R_{T,H} \dot{p}_D^H + \omega_{T,H} \times \omega_{T,H} \times R_{T,H} p_D^H + \dot{\omega}_{T,H} \times R_{T,H} p_D^H, \tag{7.11}$$

where p_D^H is a constant vector since the driver always moves along with the hexapod. The rotation matrix $R_{T,H}$ is a compound rotation consisting of the tripod yaw ϕ_T and the rotations of the hexapod. Hence,

$$R_{T,H} = R_T R_H = R_z(\phi_T) R_z(\phi_H) R_y(\theta_H) R_x(\psi_H). \tag{7.12}$$

Equation 7.11 can be simplified to

$$\ddot{p}_H^G = \ddot{p}_T^G + \omega_{T,H} \times \omega_{T,H} \times R_{T,H} p_D^H + \dot{\omega}_{T,H} \times R_{T,H} p_D^H, \tag{7.13}$$

where $\omega_{T,H} = \omega_T + \omega_H$. However, the acceleration and angular velocity perception has to be measured at the driver's brain in the drivers frame. This can be obtained by a reverse rotation given by

$$\ddot{p}_D^D = R_{T,H}^{-1} \ddot{p}_D^G, \tag{7.14}$$

where $R_{T,H}^{-1} = R_{T,H}^\top$.

Simplified Six DOF Model

In the original paper (Bruschetta et al., 2017c), the acceleration p_H^G and p_T^G are distinguished by imposing different tuning weights on them for solving the on-line optimal control problem. This strategy is not intuitive and does not reflect the fact that they are just low and high frequency signals. Therefore, we introduce a six DOF model where these two signals are separated by a low-pass and a high-pass filter, denoted as $f_{lp}(\cdot)$ and $f_{hp}(\cdot)$ respectively.

Define a universal acceleration vector $\ddot{p} = [\ddot{p}_x, \ddot{p}_y, \ddot{p}_z]^\top$. It follows that

$$\ddot{p}_{H_{x,y}}^G = f_{hp}(\ddot{p}_{x,y}), \tag{7.15}$$

$$\ddot{p}_{T_{x,y}}^G = f_{lp}(\ddot{p}_{x,y}). \tag{7.16}$$

Note that the vertical acceleration is provided by the hexapod hence $\ddot{p}_z = p^G_{H_z}$. This strategy is also applied to the separation of angular velocity of yaw ration between the tripod and the hexapod. We have

$$\dot{\phi}_H = f_{hp}(\dot{\phi}), \tag{7.17}$$
$$\dot{\phi}_T = f_{lp}(\dot{\phi}). \tag{7.18}$$

Using relationships (7.9) and (7.13) may still result to a complex dynamic model that is difficult for solver to search for an optimal solution. Several simplifications are thus introduced.

1. The angular velocity ω and the change rate of the Euler angles, e.g. $\dot{\phi}$ is usually not equivalent. However, when the Euler angles are sufficiently small, we have (Ardakani and Bridges, 2010)

$$\omega_H = [\dot{\phi}_H, \dot{\theta}_H, \dot{\psi}_H]^\top, \tag{7.19}$$
$$\omega_T = \dot{\phi}_T. \tag{7.20}$$

2. The centripetal term $\omega_T \times \omega_T \times R_T p^T_H$ in (7.9) is neglected. This is because the projection of $R_T p^T_H$ on the ground plane is sufficiently small given a small displacement p^T_H. The same criterion applies in (7.13).

3. The tangential acceleration $\dot{\omega}_T \times R_T p^T_H$ in (7.9) is neglected. This is because the tripod usually has a low frequency movement and its angular acceleration is sufficiently small. The tangential acceleration $\dot{\omega}_{T,H} \times R_{T,H} p^H_D$ is neglected by introducing a virtual sensor in the driver's brain hence p^H_D is 0 (Bruschetta et al., 2017c).

Finally, we obtain a complete dynamic model for the dynamic driving simulator with a state space

$$x_{dim} = [x_V^\top, p_H^{G^\top}, \dot{p}_H^{G^\top}, p_T^{G^\top}, \dot{p}_T^{G^\top}, E_H^\top, \omega_H^\top, E_T^\top, \omega_T^\top, s_f^\top]^\top, \tag{7.21}$$

where x_V is the state vector of the vestibular system, E_H, E_T are Euler angles of the hexapod and the tripod. The term s_f is the state vector of the high-pass and the low-pass filters. The dimension of x_{dim} is $30 + 6n_s$ where n_s is the order of the filters. The input and output of the model are given by

$$u_{dim} = [\ddot{p}^\top, \dot{\omega}^\top]^\top \in \mathbb{R}^6, \tag{7.22}$$
$$y_{dim} = [y_V^\top, p_H^{T^\top}, \dot{p}_H^{T^\top}, p_T^{G^\top}, \dot{p}_T^{G^\top}, E_H^\top, \omega_H^\top, E_T^\top, \omega_T^\top, u_{dim}^\top]^\top \in \mathbb{R}^{30}. \tag{7.23}$$

Note that the input vector of the vestibular system is given by

$$u_V = [R_{T,H}^\top(\ddot{p}_D^D + g), \omega_{T,H}^\top]^\top, \tag{7.24}$$

where g is the gravitational acceleration and is used for tilt coordination.

Actuator Constraints

Actuators for the hexapod and the tripod have to satisfy their length limits. However, due to the interaction among each leg, such constraints are not trivial. In (Bruschetta et al., 2017c), nonlinear constraints are presented for hexapod and tripod respectively. In Figure 7.3, the mechanical structure of the tripod and the hexapod is shown. Let A_i, B_i, C_j, D_j, $i = 1,\dots,6$, $j = 1,\dots,3$ be the coordinates of the actuator joints positions. Define $Q_{H,i}$ the leg vector from B_i to A_i and $Q_{T,j}$ the leg vector from C_j to D_j, we have

$$Q_{H,i} = R_H A_i + p_H^T - B_i, \tag{7.25}$$
$$Q_{T,j} = R_T C_j + p_T^G - D_j, \tag{7.26}$$

and the length of the actuators computed by $q_{H,i} = \|Q_{H,i}\|_2, q_{T,j} = \|Q_{T,j}\|_2$ should satisfy the length limits. Therefore, the nonlinear constraints can be formulated as

$$q(x_{dim}) = [q_{H,i}, q_{T,j}]^\top \in \mathbb{R}^9. \tag{7.27}$$

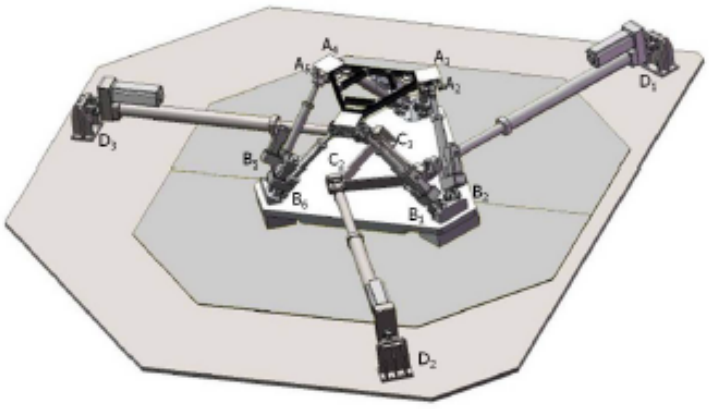


Figure 7.3: Mechanical structure of the DiM platform. A_i, B_i, C_j, D_j are the actuator joint positions; $i = 1,\dots,6$ indicate the hexapod actuators, and $j = 1,\dots,3$ indicate the tripod ones.

NMPC Formulation

The optimal control problem is formulated as

$$\min_{x_{dim}(t),u_{dim}(t)} \quad \int_0^T \|y_{dim}(t)-y_{ref}(t)\|_W^2 \mathrm{d}t + \|y_{dim}(T)-y_{ref}(T)\|_{W_T}^2 \tag{7.28}$$

$$s.t. \quad x_{dim}(0) = \hat{x}_0, \tag{7.29}$$

$$\dot{x}_{dim}(t) = f_{dim}(x_{dim}(t), u_{dim}(t)), \tag{7.30}$$

$$\underline{q} \le q(x_{dim}(t)) \le \bar{q}, \tag{7.31}$$

$$\underline{x} \le x_{dim}(t) \le \bar{x}, \tag{7.32}$$

$$\underline{u} \le u_{dim}(t) \le \bar{u} \tag{7.33}$$

where $\underline{q}, \bar{q}$ are lower and upper bounds for q, respectively and $\hat{x}_0$ is the state measurement or estimation at the current time $t = 0$. The lower and upper bounds for state variables can be found in (Bruschetta et al., 2017c).

The control frequency of the DiM application should be at least 100Hz, hence an explicit 4^{th} order RK integrator with integration length $t_I = 0.01$s is employed. The prediction horizon is chosen to be $T = 0.3$s. The cut-off frequency of the low-pass and high-pass filters is $f_c = 2$Hz, and the order for the filters is one. Simulation results show that such choices are sufficient for a successful separation of the signals, while higher order filters would increase the dimension of the state space hence make the optimal control problem more difficult to solve.

Simulation Results

The reference for the perceived signals are generated as described in (Bruschetta et al., 2017c). Indeed, the reference generation for MCA is not trivial and is important for the success of MCA. We omit this part since it is beyond the purpose of this thesis. In Figure 7.4, the tracking performance for perceived accelerations and angular velocities are shown. The most important three references, i.e. longitudinal, lateral accelerations and yaw angular velocities, are almost perfectly tracked, which leads to a faithful reproduce of the real driving sensation. At around $t = 5,21$s, strong longitudinal accelerations are required. The NMPC is able to optimally exploit tilt coordination to provide such accelerations, as can be seen in the subplot for the pitch angle velocity.

Figure 7.5 shows the displacements of the hexapod and the tripod in different DOFs, respectively. The motion of these two devices are clearly separated, i.e. the tripod is responsible to produce sustained and relatively large signals while the hexapod is responsible to produce

small but high frequency ones. It is clear that at around $t = 5,21$s, the hexapod is helping the tripod to produce sustained longitudinal accelerations by pitch tilt coordination.

Figures 7.6 and 7.7 show the actuator displacements for the hexapod and the tripod, respectively. During 50s of simulation, the actuators of the tripod do not stroke and the ones of the hexapod stroke at around $t = 43,46$s. This causes less accurate tracking of the perceived vertical acceleration as can be seen in Figure 7.4. However, NMPC is able to optimally produce the control movements while satisfying the complex actuator constraints.

The implementation is based on the MATMPC package described in Chapter 6. The real-time iterations scheme (Diehl et al., 2002) is employed using full condensing and the active-set solver qpOASES (Ferreau et al., 2014) with warm-start. All time critical steps are coded using MATLAB C language API, and compiled to be mexFunctions. The computational time for each sampling instant is around 7ms on a PC with Intel core i7-4790 running at 3.60GHz, Windows 10 platform, which satisfies the real-time requirement. However, when the actuator constraints are active, the computational time may go far beyond the real-time limit. Due to the nonlinearities in the constraints, neither block condensing plus qpDUNES (Kouzoupis et al., 2015b) [1] nor first order methods have potential to improve the computational efficiency. Future study may focus on building a map from the nonlinear actuator constraints to simple box bounds for state and control variables. Certainly there will be a trade-off between working space exploitation and the on-line computational efficiency.

7.1.1 Simulation with CMoN-RTI schemes

Here, an early stage comparison among three different RTI schemes is made, including the standard RTI (Diehl et al., 2002), LPC-RTI (Algorithm 5.8) and FTB-RTI (Algorithm 3.2). In order to increase the nonlinearity and complexity, a big yaw angle rotation reference is given to the DiM system. At every sampling instant, LPC-RTI scheme updates every one among three sensitivities within the prediction window and FTB-RTI scheme updates 20% of sensitivities with the biggest CMoN values. The sampling frequency is 200Hz.

In Figure 7.8, the outputs of the DiM system when employing the three schemes are shown. For the purpose of comparison, the reference signal is not presented. It can be seen that the yaw angle reach up to about 30 degree and has a change range up to 60 degree. However, LPC- and FTB-RTI can both grant acceptable control performance. The accelerations and angular velocities are well tracked, despite that LPC-RTI has a few tracking errors while compared to the standard RTI. For LPC-RTI, the deviation on displacements of the tripod is more obvious, while FTB-RTI has almost the same output to that of the standard RTI.

[1] Nonlinear constraints result in polytopic constraints in QP subproblems, which is not supported yet by qpDUNES

These results demonstrate that the proposed schemes in the thesis can not only be applied to toy examples such as the inverted pendulum (1.1) and the chain of masses (1.6), but also to real world systems like the DiM driving simulator. The high nonlinearities that are introduced by the big yaw angle rotations are well captured by updating only a small portion of the sensitivities.

Future research will be focused on implementations of the such models and algorithms via MATMPC, especially in MEX mode for real-time simulations.

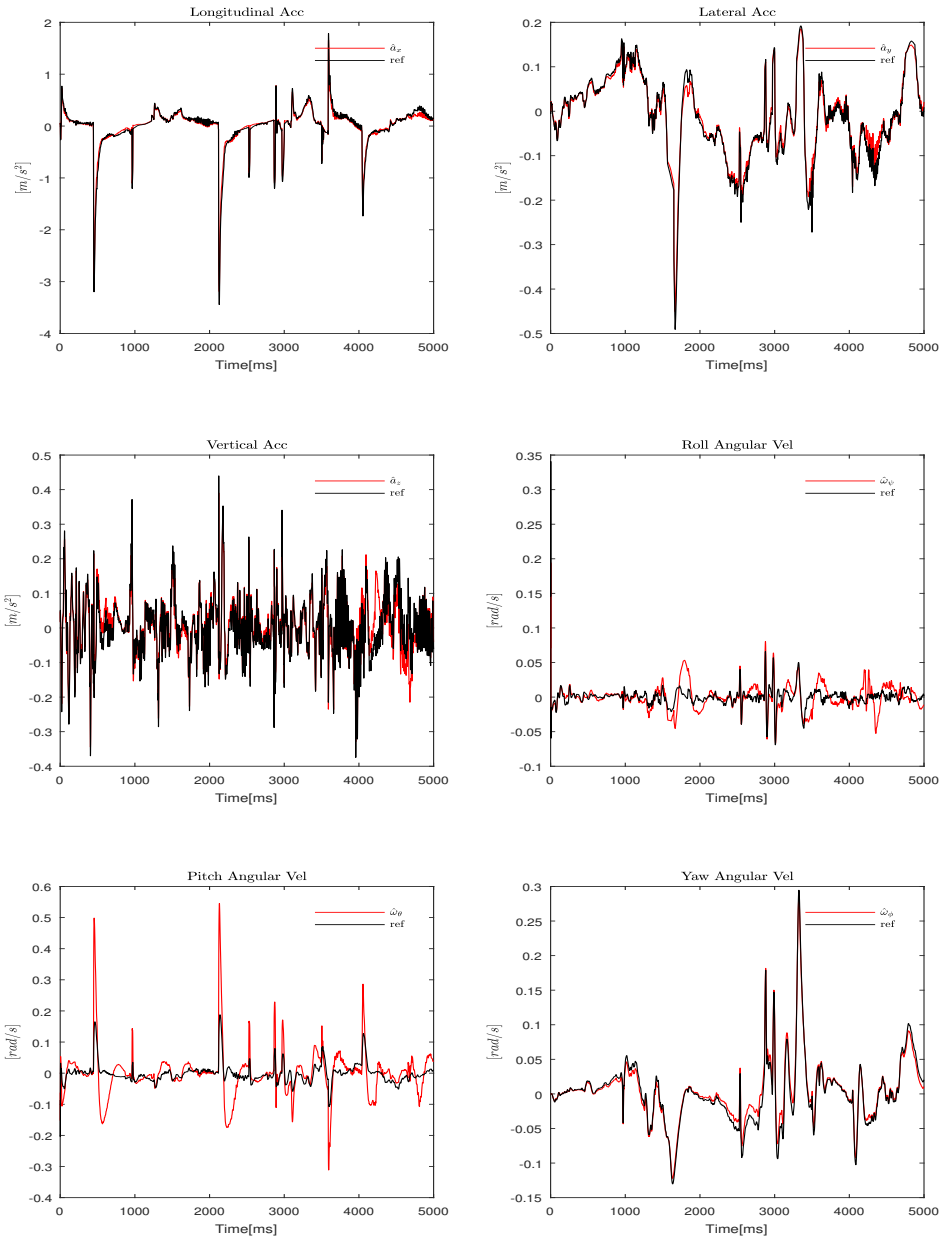


Figure 7.4: Tracking performance for perceived accelerations and angular velocities.

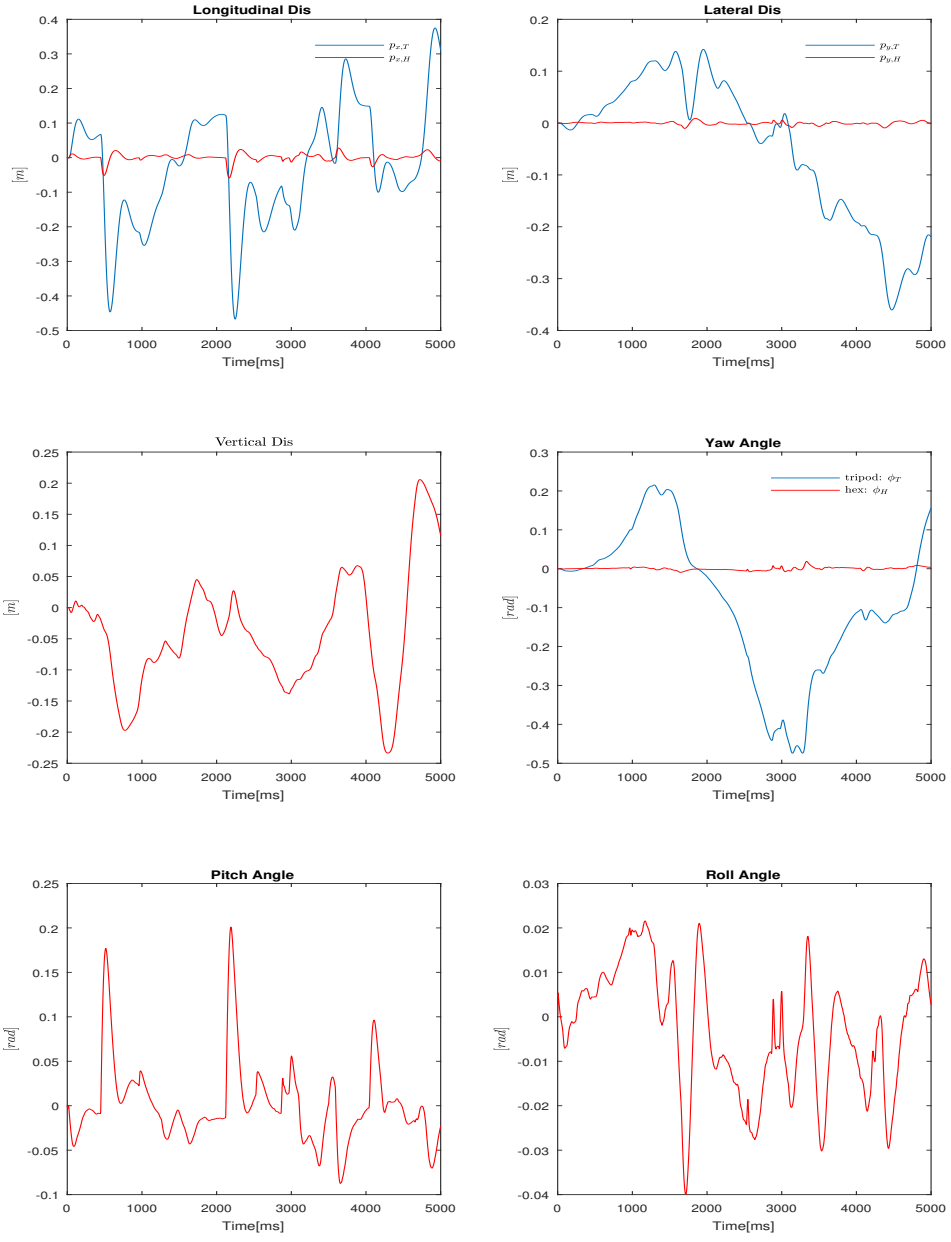


Figure 7.5: Displacements of the hexapod and the tripod in different DOFs, respectively.

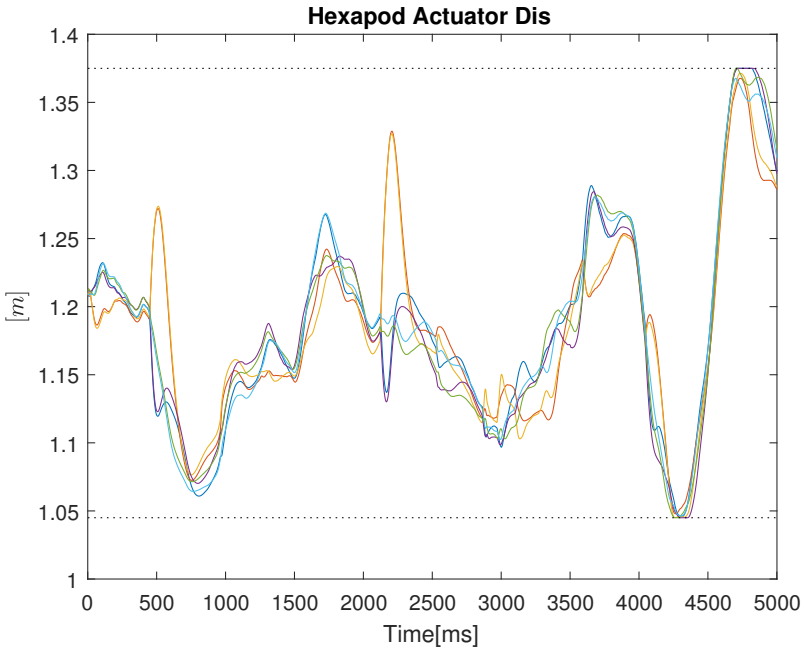


Figure 7.6: Actuator displacements for the hexapod with upper and lower bounds.

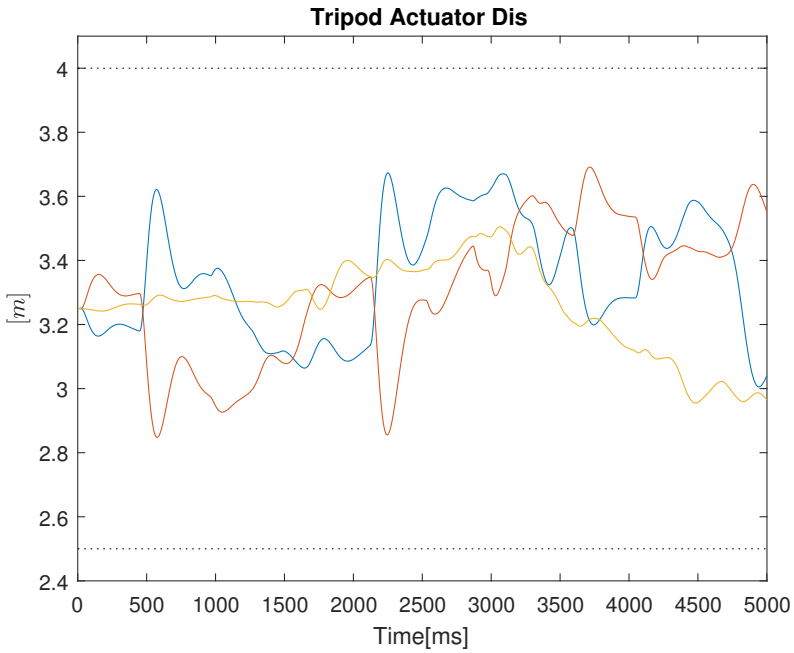


Figure 7.7: Actuator displacements for the tripod with upper and lower bounds.

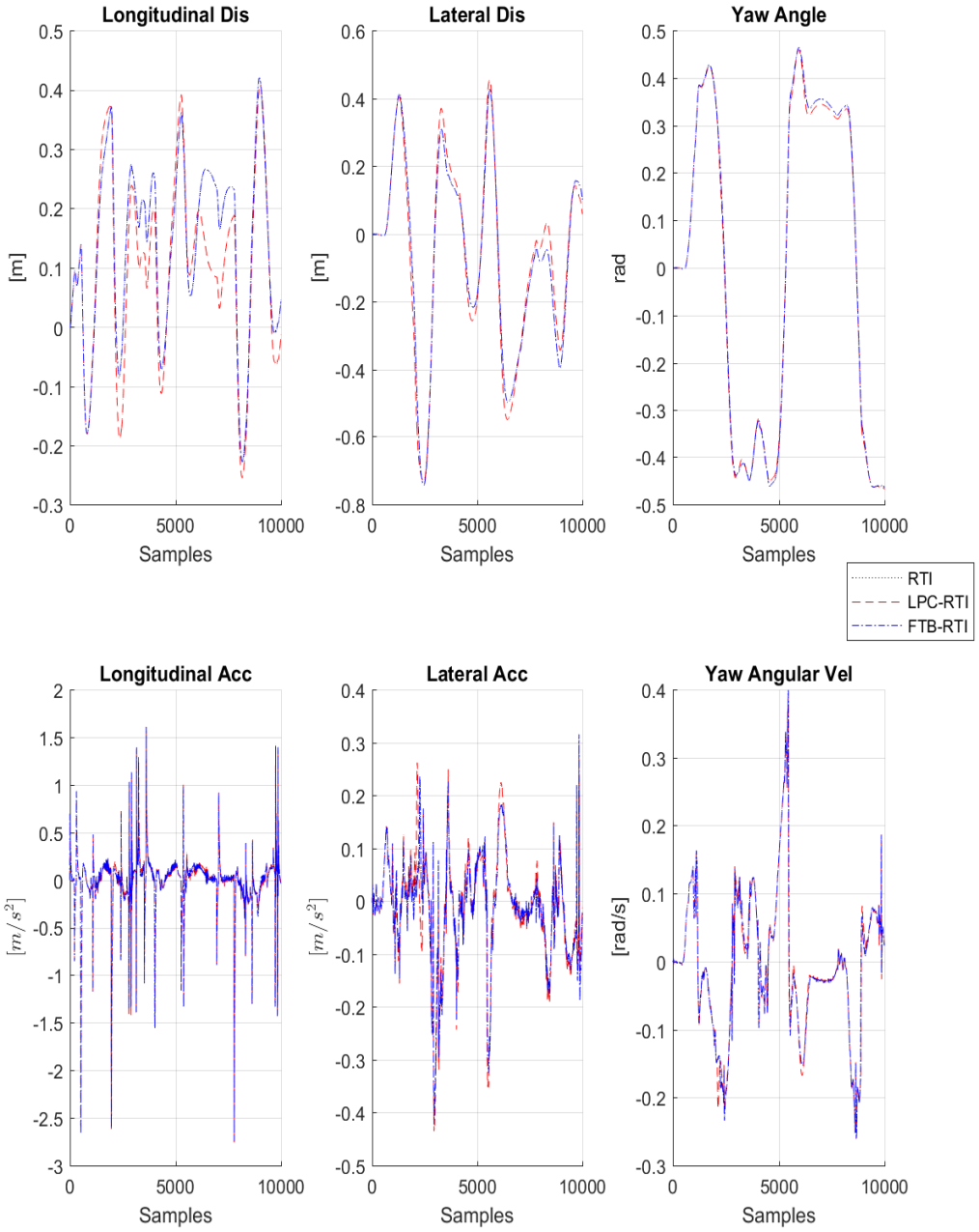


Figure 7.8: Comparison of outputs from the standard RTI, LPC-RTI and FTB-RTI schemes when applied to big yaw angle rotations. The top three figures are the longitudinal, lateral displacements and yaw angle of the tripod. The bottom three figures are longitudinal, lateral accelerations and yaw angular velocities of the tripod.

7.2 Multi-Sensory Cueing with Active Seat

The contents of this subsection are based on the paper (Bruschetta et al., 2017b). Due to physical-mechanical limitations, dynamic driving simulators are typically used to reproduce transient phenomena. The absence of the so-called "sustained accelerations" is one of the main factors that contributes to increase the conflict between real and virtual environment, which typically produces sickness in non-trained drivers. An interesting add-on that is capable of providing the driver with sustained accelerations is the Active Seat (AS) with integrated Active Belts (AB). Although the application in static simulators is almost straightforward, its usage in dynamic ones requires a specific algorithm, capable of managing, on-line, the interaction between platform acceleration and AS.

In the framework of Model Predictive Control (MPC), a Multi-Sensory Cueing Algorithm (MSCA) is proposed in (Bruschetta et al., 2017b) that computes coordinated references for the platform and for the AS/AB systems. A dynamic model of a seated human body is used to relate forces and contact pressures acting on the driver, then coupling it with the vestibular model and the platform motion. A scheme of the procedure is (Figure 7.9)

1. the vehicle translational accelerations and rotational velocities $\{a, v\}$ are computed by using a dynamical simulation engine;

2. signals $\{a, v\}$ are then pre-processed (according to given application/performance objectives);

3. the vestibular/pressure model is used to compute the reference for the controller ;

4. the platform displacements and the AS/AB pressures/tensions are computed by means of an NMPC controller, and then given as reference inputs to the motion controller.

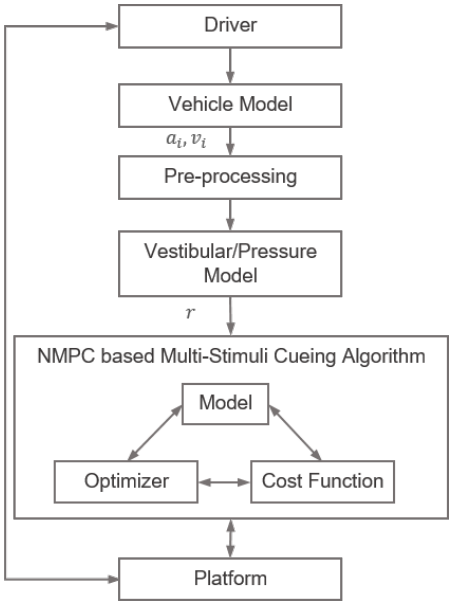


Figure 7.9: NMPC Scheme for MSCA

The advantages of such approach are manifold:

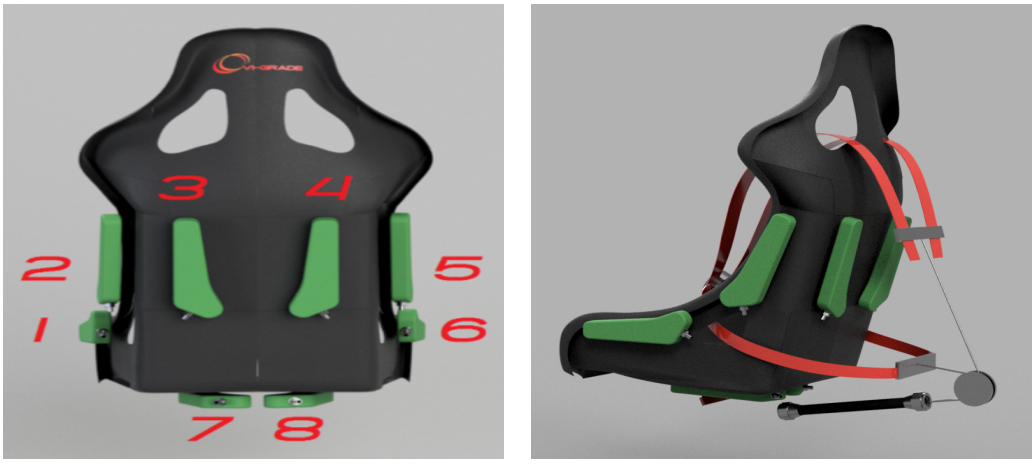
1. the platform motion, the AS and the AB are perfectly coordinated thanks to a coupled model;

2. the usage of an optimization based controller allows to have an haptic stimulus that is as closer as possible to the real one, improving the overall realism;

3. the sickness could be reduced even for untrained simulators' drivers, extending the class of potential simulator user;

4. the presence of a unique model makes the tuning procedure easier: the AS/SB system does not require to be re-tuned every time the motion cueing tuning is modified.

As the crucial factor of MSCA, the AS is a pneumatic system that consists of eight air bladders in specific locations which are inflated with air and controlled by means of proportional valves (see Figure 7.10). The AB are, more classically, five-point belts tensioned by means of electric motor.



(a) (b)

Figure 7.10: Active seat and active belts system

The working principle is quite intuitive: bladders 2 and 5 provide the pressure for the trunk of driver's body, 1 and 6 to the legs along the lateral direction, 3 and 4 to the back for positive longitudinal acceleration and, finally, 7 and 8 to the glutes along the vertical direction. The bladders are designed to have a distributed contact area and placed to act on the body similarly to what happens in reality. Proportional valves are used to have a progressive and continuous variation of pressure, which can be varied in the range [0-1.5] bar.

Model

The lateral dynamic of the body is characterized by means of a mass-spring-damper model, represented by the following differential equation:

$$m\ddot{d}_y + c(d_y)\dot{d}_y + k(d_y)d_y = ma_y \tag{7.34}$$

where d_y, $\dot{d}_y$ and $\ddot{d}_y$ represent respectively the position, velocity and acceleration of the center of mass of the trunk along the lateral direction; m is the driver mass which is subject to lateral acceleration; $c(d_y)$ is a nonlinear viscous damping coefficient; $k(d_y)$ is a nonlinear stiffness; ma_y is the external force that acts on the human body, caused by the lateral acceleration. The parameters are determined in the sequel. Let us define M the mass of the driver trunk, with arms and hands, which is assumed to be about the 67% of total body weight. The mass subject to the lateral acceleration is $m = 4/3M$ (Bruschetta et al., 2017b). The other two

parameters $k(d_y)$ and $c(d_y)$ are chosen as polynomial functions of the form

$$k(d_y) = k_p d_y^p + k_0, \tag{7.35}$$

$$c(d_y) = c_p d_y^p + c_0 \tag{7.36}$$

where k_0 and c_0 express the stiffness and the damping of the free trunk, p regulates the softness of the seat and k_p and c_p are directly related to the free space between trunk and seat bends. As an example a possible function for $k(d_y)$ is reported in Figure 7.11: driver is free to move in a range of [-0.02 0.02]m, whereas the seat is providing almost 500N at 0.06m.

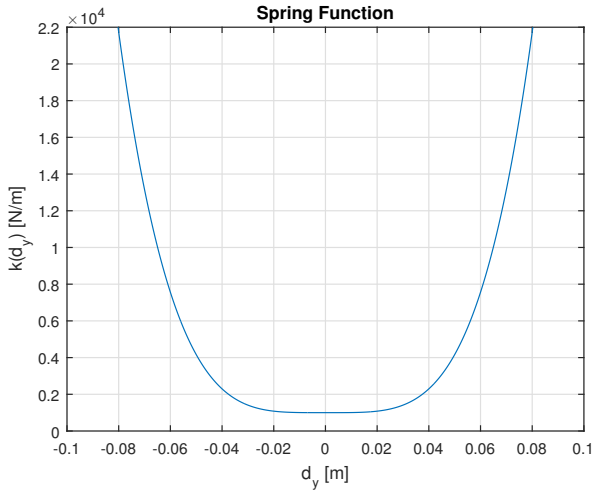


Figure 7.11: Spring function $k(d_y)$.

In addition, a dynamic friction model is used. The model is based on the average behaviour of the bristles, which is denoted here by z_m, and modelled as

$$\dot{z}_m = v - \frac{|v|}{h(v)} z_m, \tag{7.37}$$

where v is the relative velocity between the two surfaces. The friction force generated from the bending of the bristles is described as

$$F_f = \sigma_0 z + \sigma_1 \dot{z}_m. \tag{7.38}$$

The dynamic friction model is characterized by the function h and the parameters σ_0, σ_1. The function $h(v)$ is positive and depends on many factors such as material properties, lubrication,

temperature. A common parameterization of h that describes the Stribeck effect is

$$h(v) = F_d + (F_s - F_d)e^{(v/v_s)^2}, \tag{7.39}$$

where F_d is the dynamic friction level, F_s is the level of the static friction force and v_s is the Stribeck velocity.

To move the human body, the external force must overtake the static friction force F_s. We suppose an inclined backrest of angle α with respect to the vertical axis. The normal force F_N is then:

$$F_N = ma_x \cos(\alpha) + mg \sin(\alpha) \tag{7.40}$$

where a_x is the longitudinal acceleration. The dynamic friction force is

$$F_d = \mu_d F_N. \tag{7.41}$$

where F_N is the same given by (7.40).

The lateral model is completed by taking into account the tilt coordination. As the roll angle introduces a lateral acceleration perception, the pressure acting on driver is influenced by the seat inclination angle. A term is thus added to the external force acting on the driver, i.e.

$$F_{tilt} = mg \sin(\phi),$$

which, using a small-angle linearization, becomes

$$F_{tilt} = mg\phi. \tag{7.42}$$

Finally, the nonlinear system for lateral accelerations can be written as:

$$\begin{cases} \ddot{d}_y &= -\frac{c(d_y)}{m}\dot{d}_y - \frac{k(d_y)}{m}d_y - \frac{F_f(x_f)}{m} + a_y + g\phi \\ \dot{x}_f &= \dot{d}_y - \frac{|\dot{d}_y|}{h(\dot{d}_y, a_x)}x_f \\ y_{p,y} &= \frac{(k(d_y)-k_0)}{A}d_y + \frac{(c(d_y)-c_0)}{A}\dot{d}_y + \bar{u}_{p,y} \end{cases} \tag{7.43}$$

where ϕ is the roll angle and F_f is taken from (7.38) where $z_m = x_f$.

Similarly to the lateral system, longitudinal model is represented by the following spring/-damper model

$$\begin{cases} \ddot{d}_x &= -\frac{c_x}{m}\dot{d}_x - \frac{k_x}{m}d_x + a_x + g\theta \\ y_{p,x} &= \frac{k_x}{A}d_x + \bar{u}_{p,x} \end{cases} \tag{7.44}$$

where d_x is, as before, the longitudinal trunk displacement, θ is the pitch angle and $y_{p,x}$ the longitudinal pressure, which is the sum of the pressure induced by platform motion $\frac{k_x}{A}d_x$ and the pressure exerted by the AS/AB system on the driver $\bar{u}_{p,x}$. In particular positive values are used to activate the AS while negative values to tension the belts.

Incorporating the vestibular model into the AS/AB model, the state equation of the overall NMPC model results to be

$$\begin{cases} \ddot{d}_y &= -\frac{c(d_y)}{m}\dot{d}_y - \frac{k(d_y)}{m}d_y - \frac{F_f(x_f)}{m} + a_y + g\phi \\ \dot{x}_f &= \dot{d}_y - \frac{|\dot{d}_y|}{h(\dot{d}_y, a_x)}d_y \\ \ddot{d}_x &= -\frac{c}{m}\dot{d}_x - \frac{k}{m}d_x + a_x + g\theta \\ \dot{\mathbf{x}}_V &= A_V\mathbf{x}_V + B_V\mathbf{u}_V \end{cases} \tag{7.45}$$

and the input and output vectors are, respectively,

$$\begin{aligned} \mathbf{u} &= \begin{bmatrix} \bar{u}_{p,y} & \bar{u}_{p,x} & \mathbf{u}_V^T \end{bmatrix}^T \in \mathbb{R}^8 \\ \mathbf{y} &= \begin{bmatrix} y_{p,y} & y_{p,x} & \mathbf{y}_V^T \end{bmatrix}^T \in \mathbb{R}^{20}. \end{aligned} \tag{7.46}$$

NMPC Implementation

In order to satisfy the hard real-time constraint (200Hz sampling frequency or 5ms sample time), the RTI scheme (Diehl et al., 2002) is used. As a result, the following NLP problem is solved on-line at every sampling instant after multiple shooting discretization.

$$\begin{aligned} \min_{\mathbf{x},\mathbf{u}} \quad & \sum_{k=0}^{N-1} ||\mathbf{y}^k - \tilde{\mathbf{y}}^k||_W^2 + ||\mathbf{x}^N - \tilde{\mathbf{x}}^N||_{W_N}^2 \\ s.t. \quad & \mathbf{x}^{k+1} = \Xi(\mathbf{x}^k, \mathbf{u}^k) \\ & \underline{x} \le \mathbf{x}^k \le \bar{x} \\ & \underline{u} \le \mathbf{u}^k \le \bar{u} \end{aligned} \tag{7.47}$$

where $\tilde{\mathbf{y}}, \tilde{\mathbf{u}}$ are the reference and Ξ represents the integration of nonlinear dynamics described in (7.45). The corresponding QP problem reads

$$\begin{aligned} \min_{\Delta\mathbf{x},\Delta\mathbf{u}} \quad & \frac{1}{2}\begin{bmatrix}\Delta\mathbf{x}\\ \Delta\mathbf{u}\end{bmatrix}^T H \begin{bmatrix}\Delta\mathbf{x}\\ \Delta\mathbf{u}\end{bmatrix} + g^T \begin{bmatrix}\Delta\mathbf{x}\\ \Delta\mathbf{u}\end{bmatrix} \\ s.t. \quad & \Delta\mathbf{x}^{k+1} = A^k\Delta\mathbf{x}^k + B^k\Delta\mathbf{u}^k + c^k \\ & \underline{x} - \mathbf{x}^k \le \Delta\mathbf{x}^k \le \bar{x} - \mathbf{x}^k \\ & \underline{u} - \mathbf{u}^k \le \Delta\mathbf{u}^k \le \bar{u} - \mathbf{u}^k \end{aligned} \tag{7.48}$$

where $\Delta\mathbf{x} = \mathbf{x} - \mathbf{x}^k, \Delta\mathbf{u} = \mathbf{u} - \mathbf{u}^k$. The time-varying matrix H is the Hessian of the Lagrangian of (7.47) and is approximated using Gauss-Newton method here. Matrices A^k, B^k are the linearizations of system dynamics over the prediction horizon. The solving procedure of the (7.48) is challenging either in a sparse form (with $(N+1)n_x + Nn_u$ decision variables) or in a condensed form (with Nn_u decision variables). To further accelerate the on-line computation, the adjoint sensitivity strategy (Kirches et al., 2012) is used to exploit fixed A^k, B^k computed off-line. The dynamic model is thus slightly re-parameterized as described in Chapter 5.1.3. Specifically, the outputs $y_{p,x}, y_{p,y}$ are treated as differential states hence the output $\mathbf{y}$ is linearly dependent on all differential states and control variables. As a result, the Hessian H is time-invariant. Problem (2.15) can then be solved using Alternating Direction Method of Multipliers (ADMM) for optimal control problems (O'Donoghue et al., 2013), exploiting the matrix factorization cached off-line. Since the constraints are never active due to careful tuning of the parameters, the hard real-time requirement is satisfied by running simulations using MATMPC on a PC with Intel core i7 3.60Ghz.

Simulation Results

The MSCA is evaluated in the first part of the *Calabogie MotorSports Track* for the longitudinal dynamics, and in a double lane change maneuver for the lateral behaviour. In both cases a comparison between the compact DiM 150 and the greater DiM 700 is proposed (Figure 7.12). As the simulator size increases, the motion system can reproduce higher acceleration values and the AS/AB has to adapt in real time to the undergoing movement of the platform. Model parameters are given in (Bruschetta et al., 2017b).

In Figure 7.13 and 7.14 the perceived longitudinal acceleration and the displacement in the two simulators are reported. As expected in the DiM 700 a greater acceleration peak is achieved. In Figure 7.15 and 7.16 the excellent tracking performance of AS/AB are shown, together with the pressure induced by the platform and the one added by the AS/AB system. It is interesting to note that a perfect coordination between motion and AS/AB system is

obtained due to a coupled vestibular-pressure model. Moreover, the pressure induced by the simulator acceleration is significantly different in the two cases, smaller and shorter in the compact DiM 150. As a consequence the pressure request for the AS/AB is coordinated to have the same overall pressure. In both cases the need for a MSCA is evident, though, in the DiM 700, the required AS/AB pressure/tension plays a more relevant role.

In Figure 7.17 and 7.18, AS performances are shown. As in the longitudinal case, reference pressure signal is perfectly tracked, although in DiM 700 the 1:1 acceleration reproduction makes the AS unused, validating the correctness of the procedure. In DiM 150, the nonlinear dynamics of friction are evident, mostly when the reference acceleration crosses the value zero: it seems that a linearized model would not be representative enough of the friction phenomena.

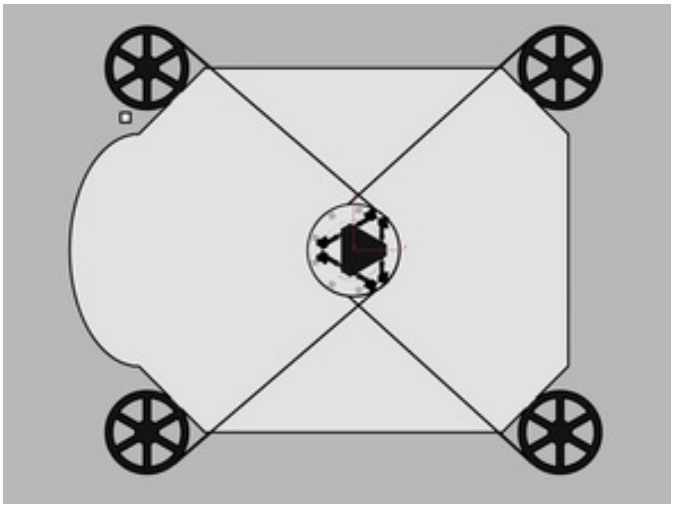


Figure 7.12: Top view sketch of DiM 700.

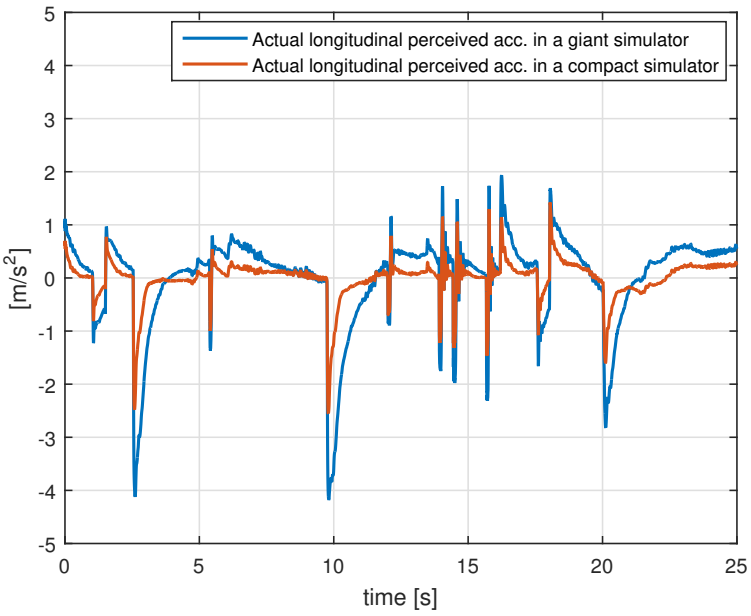


Figure 7.13: Perceived longitudinal acceleration in a compact vs giant simulator.

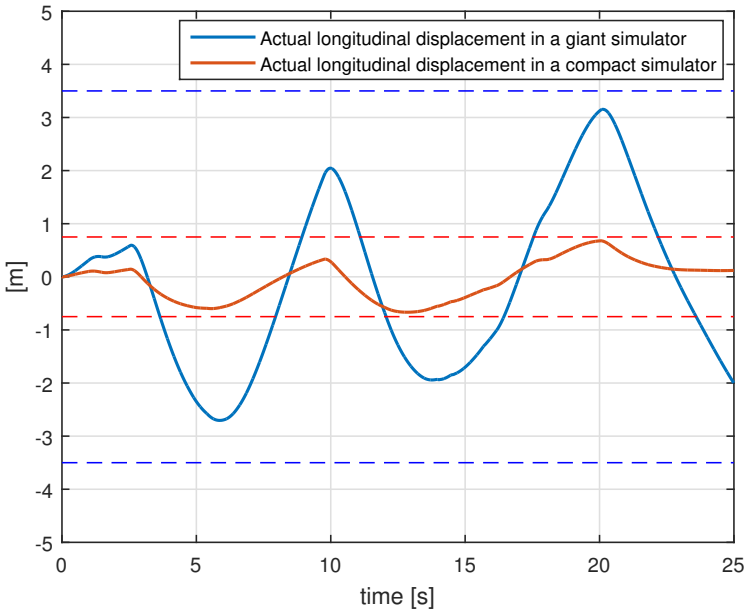


Figure 7.14: Longitudinal position displacement in a compact vs giant simulator.

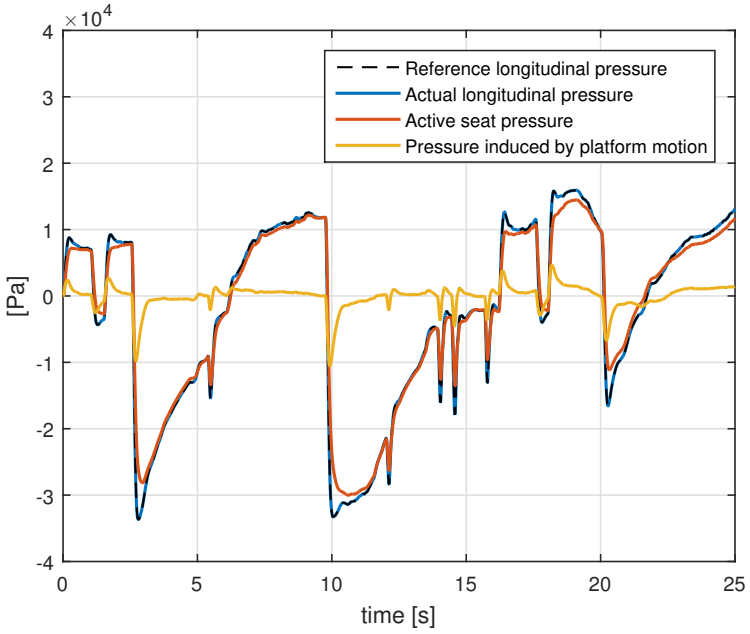


Figure 7.15: Longitudinal pressures in a compact simulator.

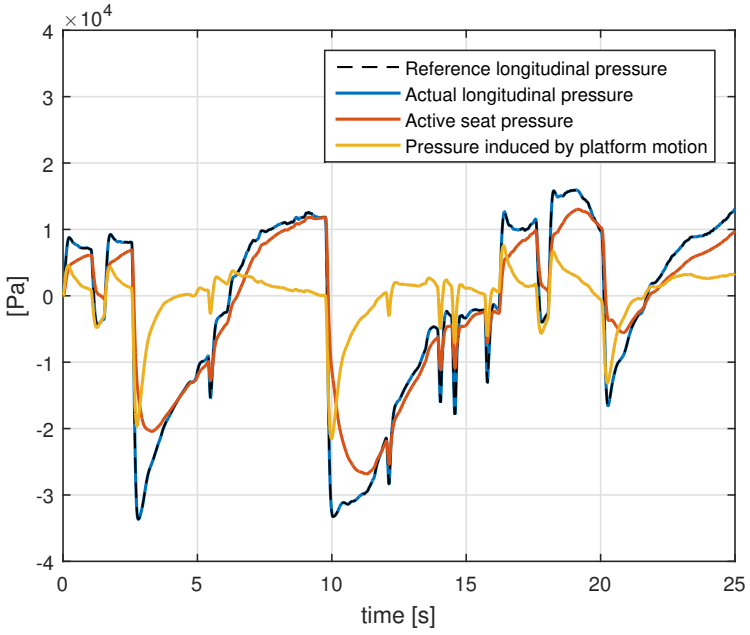


Figure 7.16: Longitudinal pressures in a giant simulator.

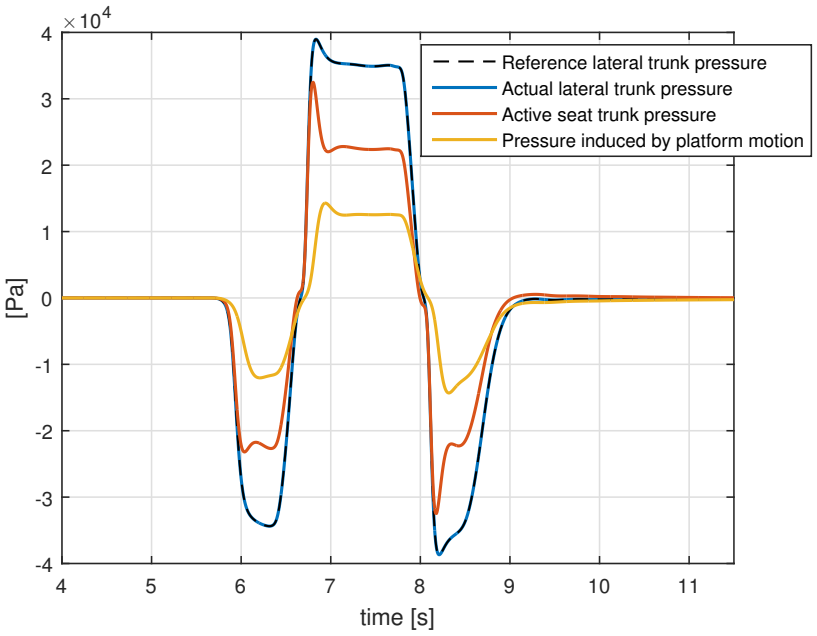


Figure 7.17: Lateral pressures in a compact simulator.

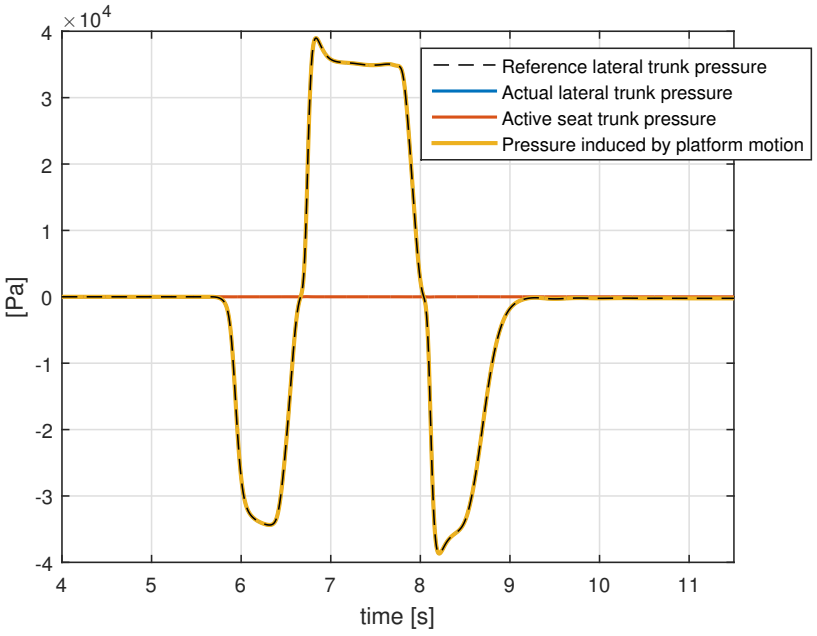


Figure 7.18: Lateral pressures in a giant simulator.

Conclusions and Outlook

In this thesis, we have presented efficient, tailored on-line optimization algorithms for fast nonlinear model predictive control using long prediction horizon. The effectiveness of the proposed algorithms have been demonstrated in several non-trivial simulation studies and numerical examples. Optimality and local convergence properties of the algorithms have also been proved. In addition, we implement NMPC algorithms in real-time for real-world applications.

Contributions

Partial sensitivity update (Chapter 3-4)

A partial sensitivity updating scheme for RTI-based NMPC algorithms has been proposed in order to reduce computational cost for sensitivity evaluation. This scheme exploits the multi-stage feature of the optimization problem due to multiple shooting and performs a block-wise update of the constraint Jacobian, which contains linearizations of state trajectories in each shooting interval. We define a novel Curvature-like Measure of Nonlinearity (CMoN) and apply it to access the degree of nonlinearity of each shooting interval. Based on the CMoN, two updating logic have been developed that result in different numerical and control performance. The first logic updates a fixed number of sensitivities with largest CMoN values. The second one introduces a constant threshold for CMoN to distinguish linear and nonlinear shooting intervals. The proposed scheme is shown to be able to reduce computational cost while maintain numerical and control performance as close as possible to that of the standard RTI scheme.

The optimality and local convergence properties have been investigated using theories in parametric programming. The partial updated, inexact Jacobian is considered as a perturbation parameter and its effects on the optimality of the solution has been quantified. As a consequence, an advanced tuning strategy is developed which converts the tuning of the threshold to that of the distance to optimum, i.e. the distance between the solutions of inexact and exact Jacobian QP problems. When considered in the full SQP scenario, the local

convergence of this advanced strategy has also been proved with a tunable convergence rate. Simulations studies and numerical examples have demonstrated the correctness of the theory.

Partial condensing and matrix factorization (Chapter 5)

Partial condensing algorithms for QP problems subject to partial sensitivity update have been developed. These algorithms overcome the shortcoming that full condensing need to be re-performed after partial sensitivity update and are capable of reducing the computational cost for condensing. In particular, an algorithm called LPC-RTI is proposed which has a linear computational complexity in prediction horizon length. Numerical examples show that a speedup by a factor of 10 is possible.

Alternatively, the QP problems can be solved directly in sparse form. The famous alternating direction method of multipliers (ADMM) is employed for time-varying QP problems in the scenario of RTI. We show that the KKT system for computing the primal variable can be partly updated thanks to partial sensitivity updating. Computational cost reduction has been proved by comparing floating point operations (flops) with classical interior point methods.

Implementations and applications (Chapter 6-7)

One of the contribution of this thesis is that a MATLAB-based NMPC package MATMPC has been developed that can be used to implement algorithms presented in this thesis. MATMPC is designed for the purpose of algorithm development hence it has two working modes. In MATLAB mode, all algorithms are implemented completely in MATLAB language hence it is quite easy for modification and debugging. Algorithm developers can focus only on mathematics and relieve themselves from professional coding. In MEX mode, we provide MEX functions for popular algorithms such as SQP, multiple shooting and (partial) condensing. MATLAB Coder can also be used to automatically generate C or MEX codes from M files. MEX mode enables MATMPC to achieve high performance when applied to real-time applications.

We use MATMPC for the application of a nine degree of freedom dynamic driving simulator. Real-time performance is achieved considering highly nonlinear dynamics and constraints using the standard RTI scheme. Therefore, algorithms are ready for real experiments on embedded platforms. A Multi-Sensory motion cueing algorithm is also tested using MATMPC running the adjoint RTI scheme and the ADMM solver. The real-time availability of the algorithm has been proved.

Outlook

Several future developments are expected to improve the results of this thesis. The theoretical properties of partial sensitivity updating can be further investigated. The stability of NMPC using such schemes is another interesting topic. In addition, partial update of the Hessian is expected to significantly reduce the computational cost, especially when exact Hessian is employed. This requires to evaluate the degree of nonlinearity of the linearized optimal control problem.

Efficient implementations of the algorithms proposed in this thesis are desired to improve MATMPC. In particular, the proposed algorithms are naturally suitable for pluralization hence a parallel implementation is desired. By using multiple processors or GPU, algorithms presented in this thesis promise to reduce considerably the on-line computational burden. An important problem that should be solved is that how to manage the data communication and synchronization.

The developed algorithms and implementations can be applied to real world problems such as the driving simulator presented in Chapter 7. Real-time NMPC is always our goal and will show its power in cutting edge industrial applications.

A

Proofs and Remarks for Chapter 4

A.1 Computation of M, N in (4.7) and (4.8)

For elements in M, we have

$$\nabla^2 \mathcal{L}_{QP} = H, \tag{A.1}$$

$$\nabla h_k = \nabla B_k, k = 1, \ldots, n_I, \tag{A.2}$$

$$\nabla d_j(\mathbf{p}) = \nabla G_j + P_{j,:}, j = 1, \ldots, n_E, \tag{A.3}$$

where $P_{j,:}$ is the jth row of P. For elements in N, we have

$$\nabla^2_{\mathbf{p}\Delta w}\mathcal{L} = \begin{bmatrix} O_1 & & & \\ \vdots & & & \\ O_1 & & & \\ \Lambda_{n_x+1} & & & \\ \vdots & & & \\ \Lambda_{2n_x} & & & \\ & \ddots & & \\ & & \Lambda_{Nn_x+1} & O_2 \\ & & \vdots & \vdots \\ & & \Lambda_{(N+1)n_x} & O_2 \end{bmatrix}, \tag{A.4}$$

$$\nabla_{\mathbf{p}} h_j = O_3 \in \mathbb{R}^{1\times n_p}, j = 0,\ldots,n_I, \tag{A.5}$$

$$\nabla_{\mathbf{p}} d = \text{blkdiag}(-\mathcal{W}_0,\ldots,-\mathcal{W}_{N-1}), \tag{A.6}$$

where $O_1 \in \mathbb{R}^{(n_x+n_u)\times(n_x+n_u)}, O_2 \in \mathbb{R}^{(n_x+n_u)\times n_x}, \Lambda_j = I_{n_x+n_u} \otimes \Delta\lambda_j$ and $\mathcal{W}_k = I_{n_x} \otimes \Delta w_k^\top$ with $\Delta w_k^\top \in \mathbb{R}^{1\times(n_x+n_u)}$.

A.2 Derivation of V^i_{pri} in (4.14)

From the updating logic Ω_2 by Algorithm 3.3, we can obtain

$$\|P_k^{i+1} q_k^i\| \le 2\eta_{pri}^{i+1} \|\nabla F_k(w_k^{old}) q_k^i\| \tag{A.7}$$

For the full Jacobian matrix, we have

$$\|P^{i+1}\mathbf{q}^i\| = \left\| \begin{bmatrix} O & & & & \\ P_0^{i+1} & O & & & \\ & P_1^{i+1} & & & \\ & & \ddots & O & \\ & & & P_{N-1}^{i+1} & O \end{bmatrix} \begin{bmatrix} q_0^i \\ q_1^i \\ \vdots \\ q_{N-1}^i \\ q_N^i \end{bmatrix} \right\| \tag{A.8}$$

$$= \left\| \begin{bmatrix} O \\ P_0^{i+1} q_0^i \\ \vdots \\ P_{N-1}^{i+1} q_{N-1}^i \end{bmatrix} \right\| = \sqrt{\sum_{k=0}^{N-1} \|P_k^{i+1} q_k^i\|^2} \tag{A.9}$$

$$\leq \sqrt{\sum_{k=0}^{N-1} 4(\eta_{pri}^i)^2 \|\nabla F_k(w_k^{old}) q_k^i\|^2} \tag{A.10}$$

$$= 2\eta_{pri}^i \sqrt{\sum_{k=0}^{N-1} \|\nabla F_k(w_k^{old}) q_k^i\|^2} \tag{A.11}$$

$$= 2\eta_{pri}^i \left\| \begin{bmatrix} \nabla F_0(w_0^{old}) q_0^i \\ \nabla F_1(w_1^{old}) q_1^i \\ \vdots \\ \nabla F_k(w_{N-1}^{old}) q_{N-1}^i \end{bmatrix} \right\| \tag{A.12}$$

$$= 2\eta_{pri}^i \|V_{pri}^i\|. \tag{A.13}$$

A similar derivation can be conducted for the dual inexactness in (4.19). The details are hence omitted.

A.3 Boundedness of $(\alpha^{i+1}, \beta^{i+1})$ in (4.15) and (4.20)

Given a fixed $c_1 > 0$ and $\rho^{i+1} > 0$, the larger $(\alpha^{i+1}, \beta^{i+1})$ are, the smaller the threshold upper bounds in (4.24) and (4.25). Hence, there always exist sufficiently large $(\alpha^{i+1}, \beta^{i+1})$ such that the upper bounds of primal and dual thresholds are smaller than or equal to the smallest nonzero CMoN value $\min(K_k^{i+1}), k = 0, \ldots, N-1$. As a result, all Jacobian blocks are updated and the DtO is zero and its tolerance is always respected. Since $\min(K_k^{i+1}) > 0$, $(\alpha^{i+1}, \beta^{i+1})$ can be finite. If there are zero CMoN values for some k, the corresponding shooting trajectories are linear, and the updating of the corresponding sensitivities does not affect the Jacobian approximation accuracy, hence the DtO.

A.4 Rationale of the practical implementation for Algorithm 4.1

Define $(\bar{\alpha}^{i+1}, \bar{\beta}^{i+1})$ as

$$\bar{\alpha}^{i+1} = \frac{\|\Delta \mathbf{w}(\mathbf{p}^{i+1})\|}{\|\mathbf{q}^i\|}, \, , \bar{\beta}^{i+1} = \frac{\|\Delta \lambda(\mathbf{p}^{i+1})\|}{\|\Delta \lambda(\mathbf{p}^i)\|}. \tag{A.14}$$

Then the following theorem holds.

Theorem A.4.1.

$$0 < c_\alpha = \frac{\bar{\alpha}^{i+1}}{\alpha^{i+1}} < \infty, \, , 0 < c_\beta = \frac{\bar{\beta}^{i+1}}{\beta^{i+1}} < \infty \, . \tag{A.15}$$

Proof. Assuming that $\bar{P}^{i+1}$ contains all the nonzero sensitivity blocks of P^{i+1}, then

$$\|\bar{P}^{i+1} \Delta \bar{\mathbf{w}}(\mathbf{p}^{i+1})\| = \|P^{i+1} \Delta \mathbf{w}(\mathbf{p}^{i+1})\|, \tag{A.16}$$

$$\|\bar{P}^{i+1} \bar{\mathbf{q}}^i\| = \|P^{i+1} \mathbf{q}^i\|, \tag{A.17}$$

$$\|\bar{P}^{i+1^\top} \Delta \bar{\lambda}(\mathbf{p}^{i+1})\| = \|P^{i+1^\top} \Delta \lambda(\mathbf{p}^{i+1})\|, \tag{A.18}$$

$$\|\bar{P}^{i+1^\top} \Delta \bar{\lambda}(\mathbf{p}^{i})\| = \|P^{i+1^\top} \Delta \lambda(\mathbf{p}^{i})\|, \tag{A.19}$$

where $\Delta \bar{\mathbf{w}}(\mathbf{p}^{i+1}), \bar{\mathbf{q}}^i, \Delta \bar{\lambda}(\mathbf{p}^{i+1}), \Delta \bar{\lambda}(\mathbf{p}^i)$ are sub-vectors of $\Delta \mathbf{w}(\mathbf{p}^{i+1}), \mathbf{q}^i, \Delta \lambda(\mathbf{p}^{i+1}), \Delta \lambda(\mathbf{p}^i)$ respectively, after removing elements corresponding to zero blocks in P^{i+1}. Then

$$c_\alpha = \frac{\bar{\alpha}^{i+1}}{\alpha^{i+1}} = \frac{\|\Delta \mathbf{w}(\mathbf{p}^{i+1})\|}{\|\bar{P}^{i+1} \Delta \bar{\mathbf{w}}(\mathbf{p}^{i+1})\|} \frac{\|\bar{P}^{i+1} \bar{\mathbf{q}}^i\|}{\|\mathbf{q}^i\|}, \tag{A.20}$$

$$c_\beta = \frac{\bar{\beta}^{i+1}}{\beta^{i+1}} = \frac{\|\Delta \lambda(\mathbf{p}^{i+1})\|}{\|\bar{P}^{i+1^\top} \Delta \bar{\lambda}(\mathbf{p}^{i})\|} \frac{\|\bar{P}^{i+1^\top} \Delta \bar{\lambda}(\mathbf{p}^{i})\|}{\|\Delta \lambda(\mathbf{p}^{i})\|}, \tag{A.21}$$

By performing singular value decomposition of $\bar{P}^{i+1}$, it is easy to prove that

$$0 < \frac{\sigma_{min}(\bar{P}^{i+1})}{\sigma_{max}(\bar{P}^{i+1})} \leq (c_\alpha, c_\beta) \leq \frac{\sigma_{max}(\bar{P}^{i+1})}{\sigma_{min}(\bar{P}^{i+1})} < \infty, \tag{A.22}$$

where $\sigma_{max}(\bar{P}^{i+1}) > 0, \sigma_{min}(\bar{P}^{i+1}) > 0$ are the largest and smallest singular value of $\bar{P}^{i+1}$ respectively, since $\bar{P}^{i+1}$ is nonsingular if $P^{i+1} \neq 0$. If $P^{i+1} = 0$, $(c_\alpha, c_\beta) = 1$. ∎

From theorem A.4.1, it follows that $(\bar{\alpha}^{i+1}, \bar{\beta}^{i+1})$ are bounded approximations of $(\alpha^{i+1}, \beta^{i+1})$. Note that $\bar{P}^{i+1}$ is block diagonal. As the number of updated Jacobian blocks increases, (c_α, c_β) become smaller and closer to unity, meaning that the approximations are more accurate the more sensitivities are updated. In the extreme case where all the Jacobian blocks are updated, $\bar{P}^{i+1} = 0$ and $(c_\alpha, c_\beta) = 1$, i.e. the approximation is perfect.

However, $(\bar{\alpha}^{i+1}, \bar{\beta}^{i+1})$ can only be exploited once the QP problem (4.1) is solved. Given an appropriate tolerance $\bar{e}^{i+1}$, a reasonable approximation is obtained by using (4.27). Two remarks are made regarding this approximation. First, information from previous sampling instants are often good approximations of that at the current sampling instant, when the system is approaching steady state and the controller is converging "on the fly" (Gros et al., 2016), e.g. both $\bar{\alpha}^{i}$ and $\bar{\alpha}^{i+1}$ are less than one. Second, note that $(\eta_{pri}^{i+1}, \eta_{dual}^{i+1})$ are considerably underestimated due to the conservative approach adopted to convert (**??**) into the inequality (4.12). Hence, c_2 can be tuned to modulate the conservativeness of (4.12).

A.5 Proof for Theorem 4.3.1 in Chapter 4

Let the locally *exact* solution initialized at y_p^i be

$$\Delta y_0^i = -\dot{F}^{-1}(y_p^i)F(y_p^i)\,. \tag{A.23}$$

Assume that at iteration i, the DtO is satisfied as

$$\|\Delta y_p^i - \Delta y_0^i\| = \|e^i\| \leq \bar{e}^i. \tag{A.24}$$

Let $\|d_y^i\| = \|y_p^i - y_*^i\|$ be the distance between the two sequences at the current iteration. Observe that

$$\dot{F}(y_p^i) = \dot{F}(y_*^i) + d_y^{i^\top}\ddot{F}(y_*^i) + \mathcal{O}(\|d_y^i\|^2), \tag{A.25}$$
$$F(y_p^i) = F(y_*^i) + \dot{F}(y_*^i)d_y^i + \mathcal{O}(\|d_y^i\|^2). \tag{A.26}$$

Assume that $\|d_y^i\|$ is sufficiently small and $\mathcal{O}(\|d_y^i\|^2)$ can be neglected, then by combining (4.40) and (A.23) it follows that

$$\Delta y_0^i - \Delta y_*^i = -\dot{F}^{-1}(y_*^i)(d_y^{i^\top}\ddot{F}(y_*^i)\Delta y_0^i) - d_y^i\,. \tag{A.27}$$

As a result,

$$\|\Delta y_0^i - \Delta y_*^i\| \leq g^i\|d_y^i\|\,, \tag{A.28}$$

where $g^i = \|\dot{F}^{-1}(y_*^i)(\Delta y_0^{i^\top} \ddot{F}(y_*^i)) + \mathcal{I}\|$. The distance between the two solutions at the current iteration is

$$\begin{aligned} \|\Delta y_p^i - \Delta y_*^i\| &\leq \|\Delta y_p^i - \Delta y_0^i\| + \|\Delta y_0^i - \Delta y_*^i\| && \text{(A.29)} \\ &\leq \underbrace{\bar{e}^i + g^i \|d_y^i\|}_{\|d_{\Delta y}^i\|}, && \text{(A.30)} \end{aligned}$$

and the distance between the two sequences at the next iteration is

$$\begin{aligned} \|d_y^{i+1}\| &:= \|y_p^{i+1} - y_*^{i+1}\| && \text{(A.31)} \\ &\leq \|d_y^i\| + \|\Delta y_p^i - \Delta y_*^i\| && \text{(A.32)} \\ &\leq \bar{e}^i + (1 + g^i)\|d_y^i\|. && \text{(A.33)} \end{aligned}$$

Since Algorithm 4.1 always starts from $\|d_y^0\| = 0$, $\|d_y^i\|, \forall i > 0$ is a linear combination of $(\bar{e}^0, \bar{e}^1, \ldots, \bar{e}^i)$. Therefore, it is always possible to choose a set of scalars $\{i \in \mathbb{N}^+ | \bar{e}^i \geq 0\}$, such that $\|d_y^i\| \approx 0$. Equivalently, the sequence $\{y_p\}$ can be sufficiently close to $\{y_*\}$ at every iteration.

Consider now the convergence properties of $\{y_p\}$. By assumption 2, it follows that

$$\begin{aligned} \|y_p^{i+1}\| &\leq \|d_{\Delta y}^{i+1}\| + \kappa_0 \|\Delta y_*^i\| && \text{(A.34)} \\ &\leq \|d_{\Delta y}^{i+1}\| + \kappa_0 \|d_{\Delta y}^i\| + \kappa_0 \|\Delta y_p^i\| && \text{(A.35)} \\ &= \kappa_1 + \kappa_0 \|\Delta y_p^i\| && \text{(A.36)} \end{aligned}$$

where $\kappa_1 = \|d_{\Delta y}^{i+1}\| + \kappa_0 \|d_{\Delta y}^i\|$. Since $\kappa_0 < 1$ and $\|d_{\Delta y}^{i+1}\|, \|d_{\Delta y}^i\|$ can be arbitrarily small, there exists a κ_2 satisfying $\kappa_0 \leq \kappa_2 < 1$ such that

$$\|y_p^{i+1}\| \leq \kappa_1 + \kappa_0 \|\Delta y_p^i\| \leq \kappa_2 \|\Delta y_p^i\|. \tag{A.37}$$

Therefore, the sequence $\{y_p\}$ is convergent and its convergence rate is at most identical to that of $\{y_*\}$. As proved in (Bock et al., 2007; Diehl et al., 2010), when $\{y_p\}$ does converge, it converges to the exact limit y_*^+ of the sequence $\{y_*\}$.

B

Partial Condensing Algorithms for Chapter 5

Following the idea of (Andersson, Frasch, Vukov, and Diehl, 2017), we write the coupling relationship in the mth condensing block by

$$A^{[m]}\Delta s^{[m]} = B^{[m]}\Delta z^{[m]} + a^{[m]} \Leftrightarrow \begin{bmatrix} \mathbb{I}_{n_x} & & & \\ -A_0 & \mathbb{I}_{n_x} & & \\ & \ddots & \ddots & \\ & & -A_{N_{pc}-2} & \mathbb{I}_{n_x} \end{bmatrix} \begin{bmatrix} \Delta s_0 \\ \vdots \\ \Delta s_{N_{pc}-1} \end{bmatrix}$$
$$= \begin{bmatrix} \mathbb{I}_{n_x} & & & \\ & B_0 & & \\ & & \ddots & \\ & & B_{N_{pc}-2} & \mathbb{O}_{n_x,n_u} \end{bmatrix} \begin{bmatrix} \Delta s_0 \\ \Delta u_0 \\ \vdots \\ \Delta u_{N_{pc}-1} \end{bmatrix} + \begin{bmatrix} 0 \\ a_0 \\ \vdots \\ a_{N_{pc}-2} \end{bmatrix} \tag{B.1}$$

In addition, the control variable has the following map

$$\Delta u^{[m]} = \tilde{I}\Delta z^{[m]} \Leftrightarrow \begin{bmatrix} u_0 \\ \vdots \\ u_{N_{pc}-1} \end{bmatrix} = \begin{bmatrix} \mathbb{O}_{n_u,n_x} & \mathbb{I}_{n_u} & & \\ & & \mathbb{I}_{n_u} & \\ & & & \ddots \\ & & & \mathbb{I}_{n_u} \end{bmatrix} \begin{bmatrix} \Delta s_0 \\ \Delta u_0 \\ \vdots \\ \Delta u_{N_{pc}-1} \end{bmatrix} \tag{B.2}$$

Hereafter, the $^{[m]}$ symbol is ignored. The original objective function for this block can be converted to

$$\begin{aligned}&\frac{1}{2}\Delta u^\top R\Delta u+\frac{1}{2}\Delta s^\top Q\Delta s+\Delta s^\top S\Delta u+\Delta u^\top S^\top\Delta s+\Delta u^\top g_u+\Delta s^\top g_s\\&=\frac{1}{2}\Delta z^\top H_c\Delta z+\Delta z^\top g_c\end{aligned}\tag{B.3}$$

where

$$L=A^{-1}a,\tag{B.4}$$
$$G=A^{-1}B,\tag{B.5}$$
$$H_c=\tilde{I}^\top R\tilde{I}+G^\top QG+G^\top S\tilde{I}+\tilde{I}^\top S^\top G,\tag{B.6}$$
$$g_c=\tilde{I}^\top g_u+G^\top(g_s+QL)+\tilde{I}^\top S^\top L.\tag{B.7}$$

The computation of G and L can be easily performed by implementing Algorithm 5.6 as presented in (Kouzoupis et al., 2015b). Note that G has the structure as in (5.15). The computation of g_c and H_c can be computed by exploiting the block structure of A and B, and by introducing $W=A^{-\top}QG$ and $w=A^{-\top}(g_s+QL)$,

$$g_c=\tilde{I}^\top g_u+B^\top w+\tilde{I}^\top S^\top L,\tag{B.8}$$
$$H_c=\tilde{I}^\top R\tilde{I}+B^\top W+\tilde{I}^\top S^\top G+(\tilde{I}^\top S^\top G)^\top.\tag{B.9}$$

Let's have a closer look into the components in H_c.

$$\tilde{I}^\top R\tilde{I}=\begin{bmatrix}\mathbb{O}_{n_x}&&&\\&R_0&&\\&&\ddots&\\&&&R_{N_{pc}-1}\end{bmatrix},\tag{B.10}$$

$$\tilde{I}^\top S^\top G=\begin{bmatrix}\mathbb{O}_{n_x,n_u}&\cdots&\mathbb{O}_{n_x,n_u}&\mathbb{O}_{n_x,n_u}\\S_0^\top G_{0,0}&\mathbb{O}_{n_u,n_u}&&\mathbb{O}_{n_u,n_u}\\S_1^\top G_{1,0}&S_1^\top G_{1,1}&&\vdots\\\vdots&\vdots&\ddots&\\S_{N_{pc}-1}^\top G_{N_{pc}-1,0}&S_{N_{pc}-1}^\top G_{N_{pc}-1,1}&\cdots&S_{N_{pc}-1}^\top G_{N_{pc}-1,N_{pc}-1}&\mathbb{O}_{n_u,n_u}\end{bmatrix}.\tag{B.11}$$

Assuming that

$$W = \begin{bmatrix} W_{0,0} & W_{0,1} & \cdots & W_{0,N_{pc}-1} & \mathbb{O}_{n_x,n_u} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ W_{N_{pc}-1,0} & W_{N_{pc}-1,1} & \cdots & W_{N_{pc}-1,N_{pc}-1} & \mathbb{O}_{n_x,n_u} \end{bmatrix}, \tag{B.12}$$

where $W_{i,j} \in \mathbb{R}^{n_x \times n_u}$, we have

$$B^\top W = \begin{bmatrix} W_{0,0} & & & & & \mathbb{O}_{n_x,n_u} \\ B_0^\top W_{1,0} & B_0^\top W_{1,1} & & & & \mathbb{O}_{n_u,n_u} \\ \vdots & \vdots & & & & \vdots \\ B_{N_{pc}-2}^\top W_{N_{pc}-1,0} & B_{N_{pc}-2}^\top W_{N_{pc}-1,1} & \cdots & B_{N_{pc}-2}^\top W_{N_{pc}-1,N_{pc}-2} & \mathbb{O}_{n_u,n_u} & \mathbb{O}_{n_u,n_u} \\ \mathbb{O}_{n_u,n_u} & & \cdots & & & \mathbb{O}_{n_u,n_u} \end{bmatrix}, \tag{B.13}$$

where the lower block triangular part of W can be computed by block-wise backward substitution when solving

$$A^\top W = QG. \tag{B.14}$$

The algorithm for computing H_c is given in Algorithm B.1, where only the lower block triangular part of H_c is computed.

Algorithm B.1 Calculation of $\tilde{H}_k = H$ in (5.16) with complexity $\mathcal{O}(N_{pc}^2)$

Initialize $H_{N_{pc},N_{pc}} \leftarrow R_{N_{pc}-1}$, $R_{-1} \leftarrow \mathbb{O}_{n_x}$, $B_{-1} \leftarrow \mathbb{I}_{n_x}$
for $i = 0, \ldots, N_{pc} - 1$ **do**
  $W_{N_{pc}-1,i} \leftarrow Q_{N_{pc}-1} G_{N_{pc}-1,i}$
  $H_{N_{pc},i} \leftarrow S_{N_{pc}-1}^\top G_{N_{pc}-1,i}$
  for $j = N_{pc} - 1, \ldots, i + 1$ **do**
    $H_{j,i} \leftarrow B_{j-1}^\top W_{j,i} + S_{j-1}^T G_{j-1,i}$
    $W_{j-1,i} \leftarrow A_{j-1}^\top W_{j,i} + Q_{j-1} G_{j-1,i}$
  end for
  $H_{i,i} \leftarrow R_{i-1} + B_{i-1}^\top W_{i,i}$
end for

Similarly, we follow the same procedure for computing g_c, which is given in Algorithm B.2.

Algorithm B.2 Calculation of $\tilde{g}_k$ in (5.16) with complexity $\mathcal{O}(N_{pc})$

Initialize $w_{N_{pc}-1} \leftarrow g_s^{N_{pc}-1} + Q_{N_{pc}-1} L_{N_{pc}-1}$, $\tilde{g}_k^{N_{pc}} \leftarrow g_u^{N_{pc}-1} + S_{N_{pc}-1}^\top L_{N_{pc}-1}$
for $i = N_{pc}-1, \ldots, 1$ **do**
  $g_k^i \leftarrow g_u^{i-1} + S_{i-1}^\top L_{i-1} + B_{i-1}^\top w_i$
  $w_{i-1} \leftarrow g_s^{i-1} + A_{i-1}^\top w_i + Q_{i-1} L_{i-1}$
end for
$\tilde{g}_k^0 \leftarrow w_0$

The inequality constraints can be re-written as

$$C^s \Delta s + C^u \Delta u \leq -c \tag{B.15}$$
$$\Leftrightarrow \quad (C^s G + C^u \tilde{I}) \Delta z \leq -(c + C^s L) \tag{B.16}$$
$$\Leftrightarrow \quad C^c \Delta z \leq -c_c, \tag{B.17}$$

where

$$C^c = \begin{bmatrix} C_{0,0}^c & C_{0,1}^c & & & \\ C_{1,0}^c & C_{1,1}^c & C_{1,2}^c & & \\ \vdots & \vdots & \ddots & & \\ C_{N_{pc}-1,0}^c & C_{N_{pc}-1,1}^c & \cdots & C_{N_{pc}-1,N_{pc}-1}^c & C_{N_{pc}-1,N_{pc}}^c \end{bmatrix}, \tag{B.18}$$

$$c^c = \begin{bmatrix} c_0^c \\ \vdots \\ c_{N_{pc}-1}^c \end{bmatrix}, \tag{B.19}$$

are computed according to Algorithm B.3.

Algorithm B.3 Calculation of $\tilde{C}_k = C^c$ and $\tilde{c}_k = c^c$ in (B.18) and (B.19) with complexity $\mathcal{O}(N^2)$.

for $i = 0, \ldots, N_{pc}-1$ **do**
  $C_{i,0}^c \leftarrow C_i^s G_{i,0}$
  $C_{i,i+1}^c \leftarrow C_i^u$
  for $j = i+1, \ldots, N_{pc}-1$ **do**
    $C_{j,i+1}^c \leftarrow C_j^s G_{j,i+1}$
  end for
  $c_i^c \leftarrow c_i + C_i^s L_i$
end for

References

Albersmeyer J. Adjoint-based algorithms and numerical methods for sensitivity generation and optimization of large scale dynamic systems. 2010.

Albersmeyer J., Beigel D., Kirches C., Wirsching L., Bock H. G., and Schlöder J. P. Fast nonlinear model predictive control with an application in automotive engineering. In *Nonlinear Model Predictive Control*, pages 471–480. Springer, 2009.

Albin T., Frank F., Ritter D., Abel D., Quirynen R., and Diehl M. Nonlinear mpc for combustion engine control: A parameter study for realizing real-time feasibility. In *Control Applications (CCA), 2016 IEEE Conference on*, pages 311–316. IEEE, 2016.

Allgöwer F. and Zheng A. *Nonlinear model predictive control*, volume 26. Birkhäuser, 2012.

Anderson E., Bai Z., Bischof C., Blackford L. S., Demmel J., Dongarra J., Du Croz J., Greenbaum A., Hammarling S., McKenney A., and others . *LAPACK Users' guide*. SIAM, 1999.

Andersson J. *A General-Purpose Software Framework for Dynamic Optimization*. PhD thesis, Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10, 3001-Heverlee, Belgium, October 2013.

Andersson J. A., Frasch J. V., Vukov M., and Diehl M. A condensing algorithm for nonlinear mpc with a quadratic runtime in horizon length. In *Optimization Online*, 2017.

Ardakani H. A. and Bridges T. Review of the 3-2-1 euler angles: a yaw-pitch-roll sequence. *Department of Mathematics, University of Surrey, Guildford GU2 7XH UK, Tech. Rep*, 2010.

Axehill D. Controlling the level of sparsity in mpc. *Systems & Control Letters*, 76:1–7, 2015.

Baseggio M., Beghi A., Bruschetta M., Maran F., and Minen D. An mpc approach to the design of motion cueing algorithms for driving simulators. In *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pages 692–697. IEEE, 2011.

Bates D. M. and Watts D. G. Relative curvature measures of nonlinearity. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–25, 1980.

Beghi A., Bruschetta M., and Maran F. A real time implementation of mpc based motion cueing strategy for driving simulators. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 6340–6345. IEEE, 2012.

Beghi A., Bruschetta M., and Maran F. A real-time implementation of an mpc-based motion cueing strategy with time-varying prediction. In *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, pages 4149–4154. IEEE, 2013.

Bemporad A., Morari M., Dua V., and Pistikopoulos E. N. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

Bemporad A., Morari M., and Ricker N. L. Model predictive control toolbox userâĂŹs guide. *The mathworks*, 2010.

Binder T., Blank L., Bock H. G., Bulirsch R., Dahmen W., Diehl M., Kronseder T., Marquardt W., Schlöder J. P., and von Stryk O. Introduction to model based optimization of chemical processes on moving horizons. In *Online optimization of large scale systems*, pages 295–339. Springer, 2001.

Bock H. G., Diehl M., Kostina E., and Schlo'der J. P. Constrained optimal feedback control of systems governed by large differential algebraic. *Real-Time PDE-constrained optimization*, 3:1, 2007.

Bolognani S., Bolognani S., Peretti L., and Zigliotto M. Design and implementation of model predictive control for electrical motor drives. *IEEE Transactions on industrial electronics*, 56(6):1925–1936, 2009.

Boyd S., Parikh N., Chu E., Peleato B., and Eckstein J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

Bruschetta M., Maran F., and Beghi A. A fast implementation of mpc-based motion cueing algorithms for mid-size road vehicle motion simulators. *Vehicle System Dynamics*, 55(6): 802–826, 2017a.

Bruschetta M., Cunico D., Chen Y., Beghi A., and Minen D. An mpc based multi-sensory cueing algorithm (msca) for a high performance driving simulator with active seat. In *Driving Simulation Conference 2017 EuropeVR*. Driving Simulation Association, September 6-8 2017b.

Bruschetta M., Maran F., and Beghi A. A nonlinear, mpc-based motion cueing algorithm for a high-performance, nine-dof dynamic simulator platform. *IEEE Transactions on Control Systems Technology*, 25(2):686–694, 2017c.

Byrd R., Nocedal J., and Waltz R. Knitro: An integrated package for nonlinear optimization. *Large-scale nonlinear optimization*, pages 35–59, 2006.

Camacho E. F. and Alba C. B. *Model predictive control*. Springer Science & Business Media, 2013.

Chen Y., Cuccato D., Bruschetta M., and Beghi A. A fast nonlinear model predictive control strategy for real-time motion control of mechanical systems. In *Advanced Intelligent Mechatronics (AIM), 2017 IEEE International Conference on*, pages 1780–1785. IEEE, 2017a.

Chen Y., Davide C., Bruschetta M., and Beghi A. An inexact sensitivity updating scheme for fast nonlinear model predictive control based on a curvature-based measure of nonlinearity. Accepted for publication in *Decision and Control, 2017 56th IEEE Conference on*, 2017b.

Chiang N.-Y., Huang R., and Zavala V. M. An augmented lagrangian filter method for real-time embedded optimization. *IEEE Transactions on Automatic Control*, 2017.

Cutler C. R. and Ramaker B. L. Dynamic matrix control?? a computer control algorithm. In *Joint automatic control conference*, number 17, page 72, 1980.

Daniel J. W. Stability of the solution of definite quadratic programs. *Mathematical Programming*, 5(1):41–53, 1973.

Desoer C. and Wang Y.-T. Foundations of feedback theory for nonlinear dynamical systems. *IEEE Transactions on Circuits and Systems*, 27(2):104–123, 1980.

Di Cairano S. and Brand M. On a multiplicative update dual optimization algorithm for constrained linear mpc. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pages 1696–1701. IEEE, 2013.

Diehl M. *Real-time optimization for large scale nonlinear processes*. PhD thesis, Heidelberg University, 2001.

Diehl M., Bock H. G., Schlöder J. P., Findeisen R., Nagy Z., and Allgöwer F. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, 12(4):577–585, 2002.

Diehl M., Ferreau H. J., and Haverbeke N. Nonlinear model predictive control. *Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation*, pages 391–417, 2009.

Diehl M., Findeisen R., Allgöwer F., Bock H. G., and Schlöder J. P. Nominal stability of real-time iteration scheme for nonlinear model predictive control. *IEE Proceedings-Control Theory and Applications*, 152(3):296–308, 2005.

Diehl M., Walther A., Bock H. G., and Kostina E. An adjoint-based sqp algorithm with quasi-newton jacobian updates for inequality constrained optimization. *Optimization Methods & Software*, 25(4):531–552, 2010.

Domahidi A. and Jerez J. FORCES Professional. embotech GmbH (http://embotech.com/FORCES-Pro), July 2014.

Domahidi A., Zgraggen A. U., Zeilinger M. N., Morari M., and Jones C. N. Efficient interior point methods for multistage problems arising in receding horizon control. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 668–674. IEEE, 2012.

Du J., Song C., and Li P. A gap metric based nonlinearity measure for chemical processes. In *American Control Conference, 2009. ACC'09.*, pages 4440–4445. IEEE, 2009.

Eker S. A. and Nikolaou M. Linear control of nonlinear systems: Interplay between nonlinearity and feedback. *AIChE journal*, 48(9):1957–1980, 2002.

El-Sakkary A. The gap metric: Robustness of stabilization of feedback systems. *IEEE Transactions on Automatic Control*, 30(3):240–247, 1985.

Enns R. H. *It's a nonlinear world*. Springer Science & Business Media, 2010.

Ferreau H. J., Almer S., Verschueren R., Diehl M., Frick D., Domahidi A., Jerez J. L., Stathopoulos G., and Jones C. Embedded optimization methods for industrial automatic control. In *Proceedings of the IFAC World Congress*, 2017.

Ferreau H. J., Kirches C., Potschka A., Bock H. G., and Diehl M. qpoases: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363, 2014.

Fiacco A.-V. Introduction to sensitivity and stability analysis in nonlinear programming. 1984.

Frasch J. V., Sager S., and Diehl M. A parallel quadratic programming method for dynamic optimization problems. *Mathematical Programming Computation*, 7(3):289–329, 2015.

Frasch J. V., Wirsching L., Sager S., and Bock H. G. Mixed—level iteration schemes for nonlinear model predictive control. *IFAC Proceedings Volumes*, 45(17):138–144, 2012.

Garcia C. E. and Morari M. Internal model control. a unifying review and some new results. *Industrial & Engineering Chemistry Process Design and Development*, 21(2):308–323, 1982.

Garcia C. E., Prett D. M., and Morari M. Model predictive control: theory and practice-a survey. *Automatica*, 25(3):335–348, 1989.

Gill P. E., Murray W., and Saunders M. A. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM review*, 47(1):99–131, 2005.

Griewank A. and Walther A. On constrained optimization by adjoint based quasi-newton methods. *Optimization Methods and Software*, 17(5):869–889, 2002.

Griewank A. and Walther A. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

Gros S., Zanon M., Quirynen R., Bemporad A., and Diehl M. From linear to nonlinear mpc: bridging the gap via the real-time iteration. *International Journal of Control*, pages 1–19, 2016.

Guay M., Dier R., Hahn J., and McLellan P. Effect of process nonlinearity on linear quadratic regulator performance. *Journal of process control*, 15(1):113–124, 2005.

Guay M., McLellan P., and Bacon D. Measurement of nonlinearity in chemical process control systems: The steady state map. *The Canadian Journal of Chemical Engineering*, 73 (6):868–882, 1995.

Gulan M., Salaj M., Abdollahpouri M., and Rohal-Ilkiv B. Real-time mhe-based nonlinear mpc of a pendubot system. *IFAC-PapersOnLine*, 48(23):422–427, 2015.

Han S.-P. Superlinearly convergent variable metric algorithms for general nonlinear programming problems. *Mathematical Programming*, 11(1):263–282, 1976.

Helbig A., Marquardt W., and Allgöwer F. Nonlinearity measures: definition, computation and applications. *Journal of Process Control*, 10(2):113–123, 2000.

Herceg M., Kvasnica M., Jones C., and Morari M. Multi-Parametric Toolbox 3.0. In *Proc. of the European Control Conference*, pages 502–510, Zürich, Switzerland, July 17–19 2013. http://control.ee.ethz.ch/~mpt.

Hock W. and Schittkowski K. Test examples for nonlinear programming codes. *Journal of Optimization Theory and Applications*, 30(1):127–129, 1980.

Houska B., Ferreau H. J., and Diehl M. An auto-generated real-time iteration algorithm for nonlinear mpc in the microsecond range. *Automatica*, 47(10):2279–2285, 2011.

Kalmari J., Backman J., and Visala A. A toolkit for nonlinear model predictive control using gradient projection and code generation. *Control Engineering Practice*, 39:56–66, 2015.

Kirches C., Wirsching L., Bock H., and Schlöder J. Efficient direct multiple shooting for nonlinear model predictive control on long horizons. *Journal of Process Control*, 22(3): 540–550, 2012.

Kirches C., Wirsching L., Sager S., and Bock H. G. Efficient numerics for nonlinear model predictive control. In *Recent Advances in Optimization and its Applications in Engineering*, pages 339–357. Springer, 2010.

Kouzoupis D., Ferreau H. J., Peyrl H., and Diehl M. First-order methods in embedded nonlinear model predictive control. In *Proceedings of the European Control Conference (ECC)*, 2015a.

Kouzoupis D., Quirynen R., Frasch J., and Diehl M. Block condensing for fast nonlinear mpc with the dual newton strategy. *IFAC-PapersOnLine*, 48(23):26–31, 2015b.

Kouzoupis D., Zanelli A., Peyrl H., and Ferreau H. J. Towards proper assessment of qp algorithms for embedded model predictive control. In *Control Conference (ECC), 2015 European*, pages 2609–2616. IEEE, 2015c.

Kühl P., Ferreau J., Albersmeyer J., Kirches C., Wirsching L., Sager S., Potschka A., Schulz G., Diehl M., Leineweber D. B., and others . Muscod-ii users manual. *University of Heidelberg*, 2007.

La Scala B. F., Mallick M., and Arulampalam S. Differential geometry measures of nonlinearity for filtering with nonlinear dynamic and linear measurement models. In *Optical Engineering+ Applications*, pages 66990C–66990C. International Society for Optics and Photonics, 2007.

Leineweber D. Efficient reduced sqp methods for the optimization of chemical processes described by large sparse dae models. 1999.

Li X. R. Measure of nonlinearity for stochastic systems. In *Information Fusion (FUSION), 2012 15th International Conference on*, pages 1073–1080. IEEE, 2012.

Lindscheid C., Haßkerl D., Meyer A., Potschka A., Bock H., and Engell S. Parallelization of modes of the multi-level iteration scheme for nonlinear model-predictive control of an industrial process. In *Control Applications (CCA), 2016 IEEE Conference on*, pages 1506–1512. IEEE, 2016.

Mallick M. and La Scala B. F. Differential geometry measures of nonlinearity for the video tracking problem. In *Defense and Security Symposium*, pages 62350P–62350P. International Society for Optics and Photonics, 2006.

Maran F. Model-based control techniques for automotive applications. *Ph.D. Dissertation*, 2013.

Mattingley J. and Boyd S. Cvxgen: A code generator for embedded convex optimization. *Optimization and Engineering*, 13(1):1–27, 2012.

Mayne D. Nonlinear model predictive control: Challenges and opportunities. In *Nonlinear model predictive control*, pages 23–44. Springer, 2000.

Mayne D. Q., Rawlings J. B., Rao C. V., and Scokaert P. O. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

Mehrotra S. On the implementation of a primal-dual interior point method. *SIAM Journal on optimization*, 2(4):575–601, 1992.

Mittelmann H. D. Decision tree for opimization software, 2016. URL http://plato.asu.edu/guide.html.

Nesterov Y. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Nikolaou M. and Misra P. Linear control of nonlinear processes: recent developments and future directions. *Computers & chemical engineering*, 27(8):1043–1059, 2003.

Niu R., Varshney P. K., Alford M., Bubalo A., Jones E., and Scalzo M. Curvature nonlinearity measure and filter divergence detector for nonlinear tracking problems. In *Information Fusion, 2008 11th International Conference on*, pages 1–8. IEEE, 2008.

Nocedal J., Wächter A., and Waltz R. A. Adaptive barrier update strategies for nonlinear interior methods. *SIAM Journal on Optimization*, 19(4):1674–1693, 2009.

Nocedal J. and Wright S. *Numerical optimization*. Springer Science & Business Media, 2006.

O'Donoghue B., Stathopoulos G., and Boyd S. A splitting method for optimal control. *IEEE Transactions on Control Systems Technology*, 21(6):2432–2442, 2013.

Ohtsuka T. A continuation/gmres method for fast computation of nonlinear receding horizon control. *Automatica*, 40(4):563–574, 2004.

Powell M. J. A fast algorithm for nonlinearly constrained optimization calculations. In *Numerical analysis*, pages 144–157. Springer, 1978.

Qin S. J. and Badgwell T. A. An overview of nonlinear model predictive control applications. *Nonlinear model predictive control*, pages 369–392, 2000.

Qin S. J. and Badgwell T. A. A survey of industrial model predictive control technology. *Control engineering practice*, 11(7):733–764, 2003.

Quirynen R. Automatic code generation of implicit runge-kutta integrators with continuous output for fast embedded optimization. Master's thesis, KU Leuven, 2012.

Quirynen R. *Numerical Simulation Methods for Embedded Optimization*. PhD thesis, KU Leuven, 2017.

Quirynen R., Vukov M., Zanon M., and Diehl M. Autogenerating microsecond solvers for nonlinear mpc: A tutorial using acado integrators. *Optimal Control Applications and Methods*, 36(5):685–704, 2015.

Rao A. V. A survey of numerical methods for optimal control. *Advances in the Astronautical Sciences*, 135(1):497–528, 2009.

Ricker N. L. and Lee J. Nonlinear model predictive control of the tennessee eastman challenge process. *Computers & Chemical Engineering*, 19(9):961–981, 1995.

Schweickhardt T. and Allgower F. Quantitative nonlinearity assessment-an introduction to nonlinearity measures. *Computer Aided Chemical Engineering*, 17:76–95, 2004.

Serban R. and Hindmarsh A. C. Cvodes: the sensitivity-enabled ode solver in sundials. In *ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 257–269. American Society of Mechanical Engineers, 2005.

Tan W., Marquez H. J., Chen T., and Liu J. Analysis and control of a nonlinear boiler-turbine unit. *Journal of process control*, 15(8):883–891, 2005.

Ullmann F. Fiordos: A matlab toolbox for c-code generation for first order methods. *MS thesis*, 2011.

Vanderbei R. J. and Shanno D. F. An interior-point algorithm for nonconvex nonlinear programming. *Computational Optimization and Applications*, 13(1):231–252, 1999.

Von Stryk D. M. O. Numerical solution of optimal control problems by direct collocation. In *Optimal Control*, pages 129–143. Springer, 1993.

Vukov M., Gros S., Horn G., Frison G., Geebelen K., Jørgensen J. B., Swevers J., and Diehl M. Real-time nonlinear MPC and MHE for a large-scale mechatronic application. *Control Engineering Practice*, 45:64–78, 2015.

Vukov M., Domahidi A., Ferreau H. J., Morari M., and Diehl M. Auto-generated algorithms for nonlinear model predictive control on long and on short horizons. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pages 5113–5118. IEEE, 2013.

Wächter A. and Biegler L. T. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1): 25–57, 2006.

Wang E., Zhang Q., Shen B., Zhang G., Lu X., Wu Q., and Wang Y. Intel math kernel library. In *High-Performance Computing on the Intel® Xeon PhiâĎć*, pages 167–188. Springer, 2014.

Wang L. *Model predictive control system design and implementation using MATLAB®*. Springer Science & Business Media, 2009.

Wang Y. and Boyd S. Fast model predictive control using online optimization. *IEEE Transactions on Control Systems Technology*, 18(2):267–278, 2010.

Wirsching L., Albersmeyer J., Kühl P., Diehl M., and Bock H. An adjoint-based numerical method for fast nonlinear model predictive control. In *Proceedings of the 17th IFAC World Congress, Seoul, Korea*, volume 17, pages 1934–1939. Citeseer, 2008.

Wirsching L., Bock H. G., and Diehl M. Fast nmpc of a chain of masses connected by springs. In *Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, 2006 IEEE*, pages 591–596. IEEE, 2006.

Works P. A comparison of interior point and sqp methods on optimal control problems. 2002.

Xianyi Z., Qian W., and Saar W. Openblas: an optimized blas library.(2016), 2016.

Zanelli A., Domahidi A., Jerez J., and Morari M. Forces nlp: An efficient implementation of interior-point methods for multistage nonlinear nonconvex programs. *International Journal of Control*, pages 1–17, 2017.

Zanelli A., Quirynen R., and Diehl M. An efficient inexact nmpc scheme with stability and feasibility guarantees. *IFAC-PapersOnLine*, 49(18):53–58, 2016.

Zavala V. M. and Biegler L. T. The advanced-step nmpc controller: Optimality, stability and robustness. *Automatica*, 45(1):86–93, 2009.