IMT School for Advanced Studies, Lucca
Lucca, Italy

Coordinate-Descent Augmented Lagrangian Methods for Interpretative and Adaptive Model Predictive Control

PhD Program in Computer Science and Systems Engineering

XXXIV Cycle

By

Liang Wu

2023

The dissertation of Liang Wu is approved.

PhD Program Coordinator: Alberto Bemporad, IMT School for Advanced Studies Lucca

Advisor: Prof. Alberto Bemporad, IMT School for Advanced Studies Lucca, Italy

The dissertation of Liang Wu has been reviewed by:

Gabriele Pannocchia, Department of Civil and Industrial Engineering of the University of Pisa

Dip Goswami, Department of Electrical Engineering at Eindhoven University of Technology

IMT School for Advanced Studies Lucca
2023

To my wife, daughter, and parents.

Contents

List of Figures

List of Tables

Acknowledgements

First of all, I would like to express my deepest gratitude to my advisor Prof. Alberto Bemporad, who gave me the dream opportunity to pursue my Ph.D. at IMT. His wise guidance, profound expertise, and inspiring work ethic have been instrumental in achieving the results of this thesis and have had a permanent positive impact on my personal and professional development. Secondly, I sincerely thank IMT for giving me the Erasmus exchange opportunity and Siemens Process System Enterprise (Siemens PSE) for accepting me and completing my 11-month internship. During this 11-month internship at Siemens PSE, my supervisor Maarten Nauta and Costas Pantelides granted me the assurance and motivation I needed to push forward. The thing I am most grateful for is that Bemporad and Costas both wrote me reference letters, which is the reason why I successfully applied for a postdoc at MIT. Next, I would like to sincerely thank Prof. Pannocchia and Prof. Goswami for assessing the thesis and providing useful feedback.

Thanks to all current and former members of the IMT colleagues, I enjoyed a friendly, pleasant, and motivating work environment. I want to thank Laura Ferrarotti, Nilay Saraf, Sampath Kumar Mulagaleti, Niraj Rathod, Surya Soman, Stefano Menchetti, Virginio Clemente, Mirko Hu, Na Liu, and Mengjia Zhu for the friendship, for amazing conversations overall sort of subjects, and for playing Pingpong and basketball. I also want to thank Chinmay Siwach, Deison Preve, Pavan Kumar, and Francis John for all the relaxing time (river and beach) we spent together and for enhancing our shared experience.

Finally, my most enormous thanks are to my wife. During the challenging Covid-19 period, when I was stuck in China, you married me and always gave me the most extensive support in my research, encouraging me to do what I wanted. And you sacrificed a lot, kept company with me, and even gave me the most adorable daughter, Wanwan Wu. Her moving smile, bouncing little hands, restless feet, and shaking little head constantly soothe my inner anxiety and make me believe I am the happiest person in the world.

Vita

April 06, 1993	Born, Tongcheng County (Hubei Province), China
2011-2015	B.E. in Chemical Engineering Final mark: Distinction Qinghai University, China
2015-2018	MSc. in Chemical Engineering Specialization: Computational Multiphase Flow Tianjin University, China
2018-2023	PhD Program in Computer Science and Systems Engineering Research Unit: Dynamical Systems, Control, and Optimization IMT School for Advanced Studies Lucca, Italy

Publications

1. L. Wu and A. Bemporad, "A Simple and Fast Coordinate-Descent Augmented Lagrangian Solver for Model Predictive Control," in IEEE Transactions on Automatic Control, doi: 10.1109/TAC.2023.3241238.
2. L. Wu and M. Nauta, "A rapid-prototype MPC tool based on gPROMS platform," submitted, available in arXiv preprint, arXiv::2209.00092.
3. L. Wu and A. Bemporad, "A construction-free coordinate descent augmented Lagrangian method for embedded linear MPC based on ARX models," accepted by IFAC WC 2023, available in arXiv preprint, arXiv:2207.06098.
4. L. Wu, "Equivalence of SS-based MPC and ARX-based MPC," submitted, available in arXiv preprint, arXiv:2209.00107.
5. L. Wu, "An interpretative and adaptive MPC for nonlinear systems," submitted, available in arXiv preprint, arXiv:2209.01513.

Abstract

Model predictive control (MPC) of nonlinear systems suffers a trade-off between model accuracy and real-time computational burden. This thesis presents an interpretative and adaptive MPC (IA-MPC) framework for nonlinear systems, which is related to the widely used approximation method based on successive linearization MPC and Extended Kalman Filtering (SL-MPC-EKF). First, we introduce a solution algorithm for linear MPC that is based on the combination of Coordinate Descent and Augmented Lagrangian (CDAL) ideas. The CDAL algorithm enjoys three features: (i) it is *construction-free*, in that it avoids explicitly constructing the quadratic programming (QP) problem associated with MPC; (ii) is *matrix-free*, as it avoids multiplications and factorizations of matrices; and (iii) is *library-free*, as it can be simply coded without any library dependency, 90-lines of C-code in our implementation. We specialize the algorithm for both state-space formulations of MPC and formulations based on AutoRegressive with eXogenous terms models (CDAL-ARX). The thesis also presents a rapid-prototype MPC tool based on the gPROMS platform, in which the qpOASES and CDAL algorithm was integrated. In addition, based on an equivalence between SS-based and ARX-based MPC problems we show, we investigate the relation between the proposed IA-MPC and the classical SL-MPC-EKF method. Finally, we test and show the effectiveness of the proposed IA-MPC framework on four typical nonlinear MPC benchmark examples.

Chapter 1

Introduction

1.1 Motivation and research objective

Model Predictive Control (MPC) is an advanced technique to control multi-input multi-output systems subject to constraints [32]. MPC has been widely used in diverse industrial areas, such as process [88], power electronics [33], aerospace [25], etc. The core idea of MPC is to predict the evolution of the controlled system through a dynamical model, solve an optimization problem over a finite time horizon, only implement the control input at the current time, and then repeat the optimization at the next sample step [88, 12]. After major developments in the field of MPC over the past three decades, MPC for linear plants described by the linear state-space model has made significant progress in theoretical stability analysis and real-time numerical algorithm implementation [56]. However, most industrial plants are nonlinear, and their constrained nonlinear MPC (NMPC) formulation is a non-convex optimization problem that encounters practical difficulties in terms of computational complexity and algorithm implementation, such as solving it within sampling time on embedded platforms. It is known that a trade-off exists between the described accuracy of the nonlinear model and the online NMPC computation cost: the more accurate/complicated the model, the greater the online computational cost.

This thesis presents an interpretative and adaptive MPC (IA-MPC) framework for nonlinear systems. In the IA-MPC framework, a linear state-space model is first obtained by performing the linearization of a first-principle-based model at the initial point. And then, an equivalent autoregressive with extra input (ARX) model is obtained via the SS-to-ARX transformation. This novel acquisition of ARX model allows us to keep the interpretability of a first-principle-based model and the adaptivity of the ARX model simultaneously. In online closed-loop control, the ARX model parameters are feedback corrected by using the Extended Kalman Filter (EKF) algorithm; and the corresponding ARX-based MPC problems are efficiently solved by the proposed CDAL-ARX algorithm with construction-free, matrix-free, and library-free features. The effectiveness of our IA-MPC method was illustrated by four nonlinear typical benchmarks.

1.2 Thesis outline and contributions

The thesis is structured into two parts: the first part (Chapters 2-4) focuses on the SS-based MPC and ARX-based MPC problem formulations and their solving algorithms, whereas the following part (Chapters 5-6) focuses the equivalence of the SS-based MPC and ARX-based MPC problem and the IA-MPC framework. The content in this thesis is mainly based on the work published in [115, 116, 114, 113, 112]. Chapter 2 presents a simple and fast coordinate descent Augmented Lagrangian (CDAL) solver for SS-based MPC problems. Chapter 3 presents the intern work at Siemens Process System Enterprise, in which the CDAL algorithm and the open-source qpOASES solver [29] both have been integrated into the professional process modeling platform, gPROMS. Chapter 4 presents the extension of the CDAL solver, that is, developing a CDAL-ARX algorithm for ARX-based MPC problems. Chapter 5 presents the equivalence relationship between the SS-based MPC and ARX-based MPC problem. Chapter 6 presents how the widely used successive linearization MPC with EKF framework inspires our IA-MPC based on the equivalence of SS-based MPC and ARX-based MPC problems. Our

IA-MPC framework combines the EKF algorithm to update the ARX model parameter and the CDAL-ARX algorithm to solve the corresponding ARX-based MPC problem.

The outline of the chapters, which are meant to be as self-contained as possible, is the following:

- Chapter 2, A Simple and Fast Coordinate Descent Augmented Lagrangian Solver for MPC:

  This chapter presents a novel Coordinate Descent Augmented Lagrangian (CDAL) solver for linear parameter-varying MPC problems. At each iteration, an augmented Lagrangian (AL) subproblem is solved by coordinate descent (CD), exploiting the structure of the MPC problem. The CDAL solver enjoys three main properties: (i) it is *construction-free*, in that it avoids explicitly constructing the quadratic programming (QP) problem associated with MPC; (ii) is *matrix-free*, as it avoids multiplications and factorizations of matrices; and (iii) is *library-free*, as it can be simply coded without any library dependency, 90-line of C-code in our implementation. To favor convergence speed, CDAL employs a reverse cyclic rule for the CD method, the accelerated Nesterov's scheme for updating the dual variables, a simple diagonal preconditioner, and an efficient coupling scheme between the CD and AL methods. We show that CDAL competes with other state-of-the-art methods, both in the case of unstable linear time-invariant and linear parameter-varying prediction models.

  The content of this Chapter and this abstract are reprinted from:

  L. Wu and A. Bemporad, "A Simple and Fast Coordinate-Descent Augmented Lagrangian Solver for Model Predictive Control," in IEEE Transactions on Automatic Control, doi: 10.1109/TAC.2023.3241238.

- Chapter 3, A rapid-prototype MPC tool based on gPROMS:

This chapter presents a rapid-prototype MPC tool based on the gPROMS platform, with support for the whole MPC design workflow. The gPROMS-MPC tool can not only directly interact with a first-principle-based gPROMS model for closed-loop simulations but also utilizes its mathematical information to derive simplified control-oriented models, basically via linearization techniques. It can inherit the interpretability of the first-principle-based gPROMS model, unlike the PAROC framework [87] in which the control-oriented models are obtained from black-box system identification based on gPROMS simulation data. The gPROMS-MPC tool allows users to choose when to linearize, such as at each sampling time (successive linearization) or some specific points to obtain one or multiple good linear models. The gPROMS-MPC tool implements our previous construction-free CDAL and the online parametric active-set qpOASES algorithms to solve sparse or condensed MPC problem formulations, respectively, for possible successive linearization or high state-dimension cases. The CDAL algorithm is also matrix-free and library-free, thus supporting embedded C-code generation. After many example validations of the tool, here we only show one example to investigate the performance of different MPC schemes.

The content of this Chapter and this abstract are reprinted from:

L. Wu and M. Nauta, "A rapid-prototype MPC tool based on gPROMS platform," submitted, available in arXiv preprint, arXiv::2209.00092.

- Chapter 4, A construction-free coordinate descent augmented Lagrangian method for embedded linear MPC based on ARX models:

  This chapter proposes a construction-free algorithm for solving linear MPC problems based on autoregressive with exogenous terms (ARX) input-output models. The solution algorithm relies on a coordinate-descent augmented Lagrangian (CDAL) method previously proposed by the authors, which we adapt here to exploit

the special structure of ARX-based MPC. The CDAL-ARX algorithm enjoys the construction-free feature, in that it avoids explicitly constructing the quadratic programming (QP) problem associated with MPC, which would eliminate construction costs when the ARX model changes/adapts online. For example, the ARX model parameters are linear-varying with scheduling signals, or recursively adapted from streaming input-output data with cheap computation cost, which make the ARX model widely used in adaptive control. Moreover, the implementation of the resulting CDAL-ARX algorithm is matrix-free and library-free, and hence amenable for deployment in industrial embedded platforms. We show the efficiency of CDAL-ARX in two numerical examples, also in comparison with MPC implementations based on other general-purpose quadratic programming solvers

The content of this Chapter and this abstract are reprinted from:

L. Wu and A. Bemporad, "A construction-free coordinate descent augmented Lagrangian method for embedded linear MPC based on ARX models," accepted by IFAC WC 2023, available in arXiv preprint, arXiv:2207.06098.

- Chapter 5, Equivalence of SS-based MPC and ARX-based MPC:

  Two kinds of control-oriented models used in MPC are the state-space (SS) model and the input-output ARX model. The SS model has interpretability when obtained from the modeling paradigm, and the ARX model is black-box but adaptable. This chapter aims to introduce the interpretability into the ARX model and thus proposes the modeling paradigm for the acquisition of ARX model. By first linearizing the first-principle-based models, interpretative SS models can be acquired and then transformed into equivalent ARX models based on the SS-to-ARX transformation theory. This chapter presents the Cayley-Hamilton, Observer-Theory, and Kalman Filter based SS-to-ARX transformation, showing that choosing the ARX model order should depend on the process noise to achieve

a good closed-loop performance rather than the fitting criteria in data-driven ARX identification paradigm. The resulted interpretative ARX model can be adopted in the adaptive MPC framework by adding an online updating scheme for the ARX model. An AFTI-16 MPC example is used to illustrate the equivalence of SS-based MPC and ARX-based MPC problems and to investigate the robustness of different SS-to-ARX transformations to noise.

The content of this Chapter and this abstract are reprinted from:

L. Wu, "Equivalence of SS-based MPC and ARX-based MPC," submitted, available in arXiv preprint, arXiv:2209.00107.

- Chapter 6, An interpretative and adaptive MPC for nonlinear systems:

  Model predictive control (MPC) for nonlinear systems suffers a trade-off between the model accuracy and real-time computational burden. One widely used approximation method is the successive linearization MPC with the EKF (SL-MPC-EKF) method, in which the EKF algorithm is to handle unmeasured disturbances and unavailable full states information. Inspired by this, an interpretative and adaptive MPC (IA-MPC) method, is presented in this chapter. In our IA-MPC method, a linear state-space model is firstly obtained by performing the linearization of a first-principle-based model at the initial point, and then this linear state-space model is transformed into an equivalent ARX model. This interpretative ARX model is then updated online by the EKF algorithm, which is modified as a decoupled one without matrix-inverse operator. The corresponding ARX-based MPC problem are solved by our previous construction-free, matrix-free and library-free CDAL-ARX algorithm. This simple library-free C-code implementation would significantly reduce the difficulty in deploying nonlinear MPC on embedded platforms. The performance of the IA-MPC method is tested against the nonlinear MPC with EKF and SL-MPC-EKF

method in four typical nonlinear benchmark examples, which show the effectiveness of our IA-MPC method.

The content of this Chapter and this abstract are reprinted from:

L. Wu, "An interpretative and adaptive MPC for nonlinear systems," submitted, available in arXiv preprint, arXiv:2209.01513.

Concluding remarks that highlight the contributions of this thesis and notes on relevant open problems for future research are included in Chapter 7.

Chapter 2

A Simple and Fast Coordinate-Descent Augmented-Lagrangian Solver for MPC

2.1 Introduction

Apart from small-scale linear time-invariant (LTI) MPC problems whose explicit MPC control law can be obtained [14], deploying an MPC controller in an electronic control unit requires an embedded Quadratic Programming (QP) solver. In the past decades, the MPC community has made tremendous research efforts to develop embedded QP algorithms, based on interior-point methods [108, 111], active-set algorithms [28, 8], gradient projection methods [81], the alternating direction method of multipliers (ADMM) [17, 103], and other techniques [58, 40, 7, 97, 98].

A demanding requirement for industrial MPC applications is code simplicity, for easily being verified, validated, and maintained on embedded platforms. In this respect, the interior-point and active-set methods require more complicated arithmetic operations in their algorithm implementations when compared to first-order optimization methods

like gradient projection and ADMM. The first-order optimization methods are quite appealing in embedded MPC since their embedded implementations could only involve additions and multiplications (no divisions, square roots, etc.). However, most of the proposed approaches require that the MPC-to-QP transformation is explicitly constructed for consumption by the solver, such as for preconditioning, estimating the Lipschitz constant of the cost gradient, and factorizing matrices. This may not be an issue for linear time-invariant (LTI) MPC problems, in which the MPC-to-QP construction and other operations on the problem matrices can be done offline. But for some linear parameter-varying (LPV) or for linear time-varying MPC problems in which the linear dynamic model, cost function, and/or constraints change at run time, an explicit online MPC-to-QP construction increases the complexity of the embedded code and computation time. Avoiding an explicit MPC-to-QP construction can be called a construction-free property of an MPC solver. The barrier interior-point FastMPC solver [108] and the active-set based BVLS solver [98] are construction-free; they directly use the model and weight matrices to define the MPC problem without constructing a QP problem. However, their complicated implementations are not matrix-free as involving Cholesky or QR factorizations arithmetic operations during iterations. The well-known simple and efficient first-order method OSQP [103] is not construction-free and matrix-free when applied to solve LPV-MPC problems, as it requires that matrix factorizations are computed and cached on each sampling time. The OSQP utilizes its own LDL^T solver to perform matrix factorizations, thus being library-free.

2.1.1 Contribution

By combining the coordinate descent (CD) and augmented Lagrangian (AL) methods, in this chapter, we develop a construction-free, matrix-free, and library-free solver for LTI and LPV MPC problems that are particularly suitable for embedded industrial deployment.

Coordinate descent has received extensive attention in recent years

due to its application to machine learning [42, 20, 95]. In this chapter, we will exploit the special structure arising from linear MPC formulations when applying CD. In [96, 70, 67], the authors also use AL to solve linear MPC problems with input and state constraints using the fast gradient method [71] to solve the associated subproblems. The Lipschitz constant of the cost gradient and convexity parameters [96] are needed to achieve convergence, and computing them requires, in turn, the Hessian matrix of the subproblem, hence constructing the QP problem. As the Hessian matrix of the AL subproblem is close to a block diagonal matrix, this suggests the use of the CD method to solve such a QP subproblem, due to the fact that CD does not require any problem-related parameter. Moreover, only small matrices are involved in running the CD method, namely the matrices of the linear prediction model and the weight matrices. As a result, the proposed CDAL algorithm does not require the QP construction phase and is extremely simple to implement. In addition, each update of the optimization vector has a computation cost per iteration that is quadratic with the state and input dimensions and linear with the prediction horizon.

To improve the convergence speed of CDAL, we propose four techniques: a reverse cyclic rule for CD, Nesterov's acceleration [71], preconditioning, and efficient coupling between CD and AL. While the use of a reverse cyclic rule in CD still preserves convergence, when the MPC problem is solved by warm-starting it from the shifted previous optimal solution, the gap between the initial guess and the new optimal solution is mainly caused by the last block of variables, and computing the last block at the beginning tends to reduce the overall number of required iterations to converge, as we will verify in the numerical experiments reported in this chapter. We employ Nesterov's acceleration scheme for updating the dual vector to improve computation speed and a heuristic preconditioner that simply scales the state variables. In addition, an efficient coupling scheme between CD and AL methods is proposed to reduce the computation cost of each CD iteration. To analyze the role of each component of CDAL and its computational performance with respect to other solvers (FastMPC, μAO-MPC, OSQP, and MATLAB's

`quadprog`), we conduct numerical experiments on an ill-conditioned problem of LTI-MPC control of an open-loop unstable AFTI-16 aircraft, and on LPV-MPC control of a continuously stirred tank reactor (CSTR).

2.1.2 Notation

$H \succ 0$ ($H \succeq 0$) denotes positive definiteness (semi-definiteness) of a square matrix H, H' (or z') denotes the transpose of a matrix H (or vector z), $H_{i,j}$ denotes the element of a matrix H on the ith row and the jth column, $H_{i,\cdot}, H_{\cdot,j}$ denote the ith row vector, and jth column vector of matrix H, respectively. For a vector z, $\|z\|_2$ denotes the Euclidean norm of z, $z_{\neq i}$ the subvector obtained from z by eliminating its ith component z_i.

2.2 Model Predictive Control

Consider the following MPC formulation for tracking problems

$$
\begin{aligned}
\min \quad & \frac{1}{2}\sum_{t=0}^{T-1} \|W_y\,(y_{t+1} - r_{t+1})\|_2^2 + \|W_u\,(u_t - u_t^r)\|_2^2 \\
& \qquad + \|W_{\Delta u}\Delta u_t\|_2^2 \\
\text{s.t.} \quad & x_{t+1} = Ax_t + Bu_t,\ t = 0,\dots,T-1 \\
& y_{t+1} = Cx_{t+1},\ t = 0,\dots,T-1 \\
& u_t = u_{t-1} + \Delta u_t,\ t = 0,\dots,T-1 \\
& x_{\min} \le x_t \le x_{\max},\ t = 1,\dots,T \\
& u_{\min} \le u_t \le u_{\max},\ t = 0,\dots,T-1 \\
& \Delta u_{\min} \le \Delta u_t \le \Delta u_{\max}, t = 0,\dots,T-1 \\
& x_0 = \bar{x}_0, u_{-1} = \bar{u}_{-1}
\end{aligned} \tag{2.1}
$$

in which $x_t \in \mathbb{R}^{n_x}$ is the state vector, $u_t \in \mathbb{R}^{n_u}$ the input vector, $\Delta u_t = u_t - u_{t-1}$ the vector of input increments, $y_t \in \mathbb{R}^{n_y}$ the output vector, r_t and u_t^r are the output and input set-points, and $\bar{x}_0$ and $\bar{u}_{-1}$ denote the current state and the previous input vectors, respectively. We assume

that $W_y = W_y' \succeq 0$, $W_u = W_u' \succeq 0$, $W_{\Delta u} = W_{\Delta u}' \succ 0$. The formulation (2.1) could be extended to include time-varying bounds on x and u along the prediction horizon, linear equality constraints or box constraints on the terminal state x_T for guaranteed closed-loop convergence, as well as affine prediction models. To simplify the notation, in the sequel we consider the following reformulation of (2.1)

$$\begin{aligned}
\min \quad & \frac{1}{2}\sum_{t=1}^{T} \hat{x}_t'(\hat{C}'\hat{W}\hat{C})\hat{x}_t - \hat{x}_t'(\hat{C}'\hat{W}\hat{r}_t) + \frac{1}{2}\hat{u}_{t-1}'W_{\Delta u}\hat{u}_{t-1} \\
\text{s.t.} \quad & \hat{x}_{t+1} = \hat{A}x_t + \hat{B}\hat{u}_t,\ t = 0,\dots,T-1 \\
& \hat{x}_{\min} \le \hat{x}_t \le \hat{x}_{\max},\ t = 1,\dots,T \\
& \hat{u}_{\min} \le \hat{u}_t \le \hat{u}_{\max},\ t = 0,\dots,T-1 \\
& \hat{x}_0 = \left[\begin{smallmatrix} \bar{x}_0 \\ \bar{u}_{-1} \end{smallmatrix}\right]
\end{aligned} \tag{2.2}$$

where $\hat{x}_t = \left[\begin{smallmatrix} x_t \\ u_{t-1} \end{smallmatrix}\right] \in \mathcal{R}^{\hat{n}_x}$, $\hat{n}_x = n_x + n_u$, $\hat{u}_t = \Delta u_t \in \mathcal{R}^{n_u}$, $\hat{A} = \left[\begin{smallmatrix} A & B \\ 0 & I \end{smallmatrix}\right] \in \mathbb{R}^{\hat{n}_x \times \hat{n}_x}$, $\hat{B} = \left[\begin{smallmatrix} B \\ I \end{smallmatrix}\right] \in \mathbb{R}^{\hat{n}_x \times n_u}$, $\hat{C} = \left[\begin{smallmatrix} C & 0 \\ 0 & I \end{smallmatrix}\right]$, $\hat{W} = \left[\begin{smallmatrix} W_y & 0 \\ 0 & W_u \end{smallmatrix}\right]$, $\hat{r}_t = \left[\begin{smallmatrix} r_t \\ u^r_{t-1} \end{smallmatrix}\right]$. The vector z of variables to optimize is

$$z = \left[\begin{array}{ccccccc} \hat{u}_0' & \hat{x}_1' & \hat{u}_1' & \dots & \hat{u}_{T-1}' & \hat{x}_T' \end{array}\right]' \in \mathbb{R}^{T(\hat{n}_x + n_u)}$$

The inequality constraints on state and input variables, whose number is $2T(\hat{n}_x + n_u)$, are

$$\underline{z} \le z \le \bar{z} \Leftrightarrow \left\{ \begin{array}{l} \hat{x}_{\min} \le \hat{x}_t \le \hat{x}_{\max}, \forall t = 1,\dots,T \\ \hat{u}_{\min} \le \hat{u}_t \le \hat{u}_{\max}, \forall t = 0,\dots,T-1 \end{array} \right.$$

where $\hat{x}_{\min} = \left[\begin{smallmatrix} x_{\min} \\ u_{\min} \end{smallmatrix}\right]$, $\hat{x}_{\max} = \left[\begin{smallmatrix} x_{\max} \\ u_{\max} \end{smallmatrix}\right]$, $\hat{u}_{\min} = \Delta u_{\min}$ and $\hat{u}_{\max} = \Delta u_{\max}$. At each sample step, the MPC problem (2.1) can be recast as the following quadratic program (QP)

$$\begin{aligned}
\min \quad & \frac{1}{2}z'Hz + h'z \\
\text{s.t.} \quad & \underline{z} \le z \le \bar{z} \\
& Gz = g
\end{aligned} \tag{2.3}$$

where $H = H' \succeq 0$, $H \in \mathbb{R}^{n_z \times n_z}$, $n_z = T(\hat{n}_x + n_u)$, $h \in \mathbb{R}^{n_z}$, $G \in \mathbb{R}^{T\hat{n}_x \times n_z}$, and $g \in \mathbb{R}^{T\hat{n}_x}$ are defined as

$$
\begin{aligned}
H &= \begin{bmatrix} R & 0 & \dots & 0 & 0 \\ 0 & Q & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & R & 0 \\ 0 & 0 & \dots & 0 & Q \end{bmatrix}, \quad \begin{aligned} R &= W_{\Delta u} \\ Q &= \hat{C}' \hat{W} \hat{C} \end{aligned} \\
G &= \begin{bmatrix} \hat{B} & -I & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & \hat{A} & \hat{B} & -I & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & \hat{A} & \hat{B} & -I \end{bmatrix} \\
h &= \begin{bmatrix} -\hat{C}' \hat{W} \hat{r}_1 \\ -\hat{C}' \hat{W} \hat{r}_2 \\ \vdots \\ -\hat{C}' \hat{W} \hat{r}_T \end{bmatrix}, \; g = \begin{bmatrix} -\hat{A} \hat{x}_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}
\end{aligned}
$$

Clearly matrix G is full row-rank. Note that $A, B, C, W_y, W_u, W_{\Delta u}$ and the upper and lower bounds on x, u, and Δu in (2.1) may change at each controller execution.

2.3 Algorithm

2.3.1 Augmented Lagrangian Method

We solve the convex quadratic programming problem (2.3) by applying the augmented Lagrangian method. The bound-constrained Lagrangian function $\mathcal{L} : \mathcal{Z} \times \mathbb{R}^{T \times \hat{n}_x} \to \mathbb{R}$ is given by

$$\mathcal{L}(z, \Lambda) = \frac{1}{2} z' H z + z' h + \Lambda' (Gz - g)$$

Where $\mathcal{Z} = \{\underline{z} \leq z \leq \bar{z}\}$ and $\Lambda \in \mathbb{R}^{T\hat{n}_x}$ is the vector of Lagrange multipliers associated with the equality constraints in (2.3). The dual problem of (2.3) is

$$\max_{\Lambda \in \mathbb{R}^{T\hat{n}_x}} \phi(\Lambda) \tag{2.4}$$

where $\phi(\Lambda) = \min_{z\in\mathcal{Z}} \mathcal{L}(z,\Lambda)$. Assuming that Slater's constraint qualification holds, the optimal value of the primal problem (2.3) and of its dual (2.4) coincide. However, $\phi(\Lambda)$ is not differentiable in general [16], so that any subgradient method for solving (2.4) would have a slow convergence rate. Under the AL framework, the augmented Lagrangian function

$$\mathcal{L}_\rho(z,\Lambda) = \frac{1}{2}z'Hz + z'h + \Lambda'(Gz-g) + \frac{\rho}{2}\|Gz-g\|^2 \tag{2.5}$$

is used instead, where the parameter $\rho > 0$ is a penalty parameter. The corresponding augmented dual problem is defined as:

$$\max_{\Lambda\in\mathbb{R}^{T\hat{n}_x}} \phi_\rho(\Lambda) \tag{2.6}$$

where $\phi_\rho(\Lambda) = \min_{z\in\mathcal{Z}} \mathcal{L}_\rho(z,\Lambda)$ is differentiable provided that $H + \rho G'G \succ 0$. The dual problem (2.4) and the augmented dual problem (2.6) share the same optimal solution [15, see chapter 2 subsection 2.2], and most important $\phi_\rho(\Lambda)$ is concave and differentiable, with gradient [16, 73] $\nabla\phi_\rho(\Lambda) = Gz^*(\Lambda) - g$, where $z^*(\Lambda)$ denotes the optimal solution of the inner problem $\min_{z\in\mathcal{Z}} \mathcal{L}_\rho(z,\Lambda)$ for a given Λ. Moreover, the gradient mapping $\nabla\phi_\rho : \mathbb{R}^{T\times\hat{n}_x} \to \mathbb{R}^{T\times\hat{n}_x}$ is Lipschitz continuous, with a Lipschitz constant given by $L_\phi = \rho^{-1}$ [55].

Let $F_\rho(z;\Lambda^k) = \frac{1}{2}z'H_Az + (h_A^k)'z$, where $h_A^k = \frac{1}{\rho}h + G'\Lambda^k - G'g$, and $H_A = \frac{1}{\rho}H + G'G$ has the block-sparse structure

$$H_A = \begin{bmatrix} \phi_1 & \phi_2 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ \phi_2' & \phi_3 & \phi_4 & \phi_5 & 0 & \dots & 0 & 0 & 0 \\ 0 & \phi_4' & \phi_1 & \phi_2 & 0 & \dots & 0 & 0 & 0 \\ 0 & \phi_5' & \phi_2' & \phi_3 & \phi_4 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & \phi_3 & \phi_4 & \phi_5 \\ 0 & 0 & 0 & 0 & 0 & \dots & \phi_4' & \phi_1 & \phi_2 \\ 0 & 0 & 0 & 0 & 0 & \dots & \phi_5' & \phi_2' & \phi_6 \end{bmatrix}$$

and $\phi_1 = \frac{1}{\rho}R + \hat{B}'\hat{B}$, $\phi_2 = -\hat{B}'$, $\phi_3 = \frac{1}{\rho}Q + \left(I + \hat{A}'\hat{A}\right)$, $\phi_4 = \hat{A}'\hat{B}$, $\phi_5 = -\hat{A}'$, $\phi_6 = \frac{1}{\rho}Q + I$. Since G is full rank, the matrix $H_A \succ 0$. According to

[15], the AL algorithm can be formulated in scaled form as follows:

$$z^{k+1} = \operatorname*{argmin}_{z\in\mathcal{Z}} F_\rho(z;\Lambda^k) \tag{2.7a}$$

$$\Lambda^{k+1} = \Lambda^k + (Gz^{k+1} - g) \tag{2.7b}$$

Which involves the minimization step of the primal vector z and the update step of the dual vector Λ. As shown in [15], the convergence of AL can be assured for a large range of values of ρ. Typically, the larger the penalty parameter, the faster the AL algorithm is to converge, but the more difficult (2.7a) is to solve, due to a larger condition number of the Hessian matrix of subproblem (2.7a). The convergence rate of the AL algorithm (2.7) is $O(1/k)$ according to [39]. To improve the speed of the AL method, [47] proposed an accelerated AL algorithm, whose iteration-complexity is $O(1/k^2)$ for linearly constrained convex programs, by using Nesterov's acceleration technique. The accelerated AL algorithm is summarized in Algorithm 1.

Algorithm 1 Accelerated augmented Lagrangian method [47]

Input: Initial guess $z^0 \in \mathcal{Z}$ and Λ^0; maximum number N_{out} of iterations; parameter $\rho > 0$.

1. Set $\alpha_1 \leftarrow 1$; $\hat{\Lambda}^0 \leftarrow \Lambda^0$;
2. **for** $k = 1, 2, \cdots, N_{\text{out}}$ **do**
   - 2.1. $z^k \leftarrow \operatorname{argmin}_{z\in\mathcal{Z}} F_\rho(z;\hat{\Lambda}^{k-1})$;
   - 2.2. $\Lambda^k \leftarrow \hat{\Lambda}^{k-1} + (Gz^k - g)$;
   - 2.3. **if** $\|\Lambda^k - \hat{\Lambda}^{k-1}\|_2^2 \le \epsilon$, **stop**;
   - 2.4. $\alpha_{k+1} \leftarrow \frac{1+\sqrt{1+4\alpha_k^2}}{2}$;
   - 2.5. $\hat{\Lambda}^k \leftarrow \Lambda^k + \frac{\alpha_k - 1}{\alpha_{k+1}}(\Lambda^k - \Lambda^{k-1})$;
3. **end**.

For solving the strongly convex box-constrained QP (2.7a), the fast gradient projection method was used in [96, 67]. Inspired by the fact that

the Gauss-Seidel method in solving block tridiagonal linear systems is efficient [1], in this chapter, we propose the use of the cyclic CD method to make full use of block sparsity and avoid the explicit construction of matrix H_A. Note that in the gradient projection method or fast gradient projection method [67], the Lipschitz constant parameter deriving from matrix H_A needs to be calculated or estimated to ensure convergence. Therefore, for linear MPC problems that change at runtime, such methods would be less preferable than a cyclic CD. In this chapter, by making full use of the structure of the subproblem, we will implement a cyclic CD method that requires fewer computations, as we will detail in the next section.

2.3.2 Coordinate Descent Method

The idea of the CD method is to minimize the objective function along only one coordinate direction at each iteration while keeping the other coordinates fixed [63, 110]. In [64, 65], the authors showed that the CD method is convergent in convex differentiable minimization problems, and the rate of convergence is at least linear. We first give a brief introduction of the CD method to solve (2.7a). Under the assumption that the set of optimal solutions is nonempty and that the objective function F_ρ is convex, continuously differentiable, and strictly convex with respect to each coordinate, the CD method proceeds iteratively for $k = 0, 1, \ldots,$ as follows:

$$\text{choose } i_k \in \{1, 2, \ldots, n_z\} \tag{2.8a}$$

$$z_{i_k}^{k+1} = \underset{z_{i_k} \in \mathcal{Z}}{\operatorname{argmin}} F_\rho(z_{i_k}, z_{\neq i_k}^k; \hat{\Lambda}^k) \tag{2.8b}$$

where with a slight abuse of notation we denote by $F_\rho(z_{i_k}, z_{\neq i_k}^k; \hat{\Lambda}^k)$ the value $F_\rho(z; \hat{\Lambda}^k)$ when $z_{\neq i_k} = z_{\neq i_k}^k$ is fixed. The convergence of the iterations in (2.8) for $k \to \infty$ depends on the rule used to choose the coordinate index i_k. In [65], the authors show that the *almost cyclic rule* and *Gauss-Southwell rule* guarantee convergence. Here we use the almost cyclic rule, that provides convergence according to the following lemma:

Lemma 1 ([65]) *Let $\{z^k\}$ be the sequence of coordinate-descent iterates* (2.8)*, where every coordinate index is iterated upon at least once on every N successive iterations, $N \geq n_z$. The sequence $\{z^k\}$ converges at least linearly to the optimal solution z^* of problem* (2.7a)*.*

In this chapter, we will use the *reverse cyclic rule*

$$i_k = n_z - (k \bmod n_z)$$

To exploit the fact that the shifted previous optimal solution is used as a warm start. The chosen rule clearly satisfies the assumptions of Lemma 1 for convergence. The implementation of one pass through all n_z coordinates using reverse cyclic CD is reported in Procedure 2. In Procedure 2, the Lagrangian variable $\hat{\Lambda} \in \mathbb{R}^{T \times \hat{n}_x}$ is divided into $\{\hat{\lambda}_0, \ldots, \hat{\lambda}_{T-1}\}$, where each $\hat{\lambda}_{t-1} \in \mathbb{R}^{\hat{n}_x}$. For a given symmetric $M \in \mathbb{R}^{n_s \times n_s} \succeq 0$, $d \in \mathbb{R}^{n_s}$, the operator $\mathrm{CCD}_{[\underline{s},\bar{s}]}\{M, d\}$ used in Procedure 2 represents one pass iteration of the reverse cyclic CD method through all n_s coordinates $s_{n_s}, \ldots, s_1$ for the following box-constrained QP

$$\min_{s \in [\underline{s},\bar{s}]} \frac{1}{2} s' M s + s' d \tag{2.9}$$

that is to execute the following n_s iterations

$$\begin{aligned} &\text{for } i = n_s, \ldots, 1 \\ &\quad s_i \leftarrow \left[s_i - \tfrac{1}{M_{i,i}} (M_{i,.} s + d_i) \right]_{\underline{s}_i}^{\bar{s}_i} \\ &\text{end} \end{aligned} \tag{2.10}$$

where $[s_i]_{\underline{s}_i}^{\bar{s}_i}$ is the projection operator

$$[s_i]_{\underline{s}_i}^{\bar{s}_i} = \begin{cases} \bar{s}_i & \text{if} \quad s_i \geq \bar{s}_i \\ s_i & \text{if} \quad \underline{s}_i < s_i < \bar{s}_i \\ \underline{s}_i & \text{if} \quad s_i \leq \underline{s}_i \end{cases} \tag{2.11}$$

Note that in Procedure 2, Steps 2, 3, 4.1, and 4.2 all involve the same operator CCD. In Procedure 3, we exemplify an efficient way to evaluate such an operator for Step 4.2 of Procedure 2, as the approach is similar for evaluating Steps 2, 3, and 4.1, where σ records the sum of squared coordinate variations.

Procedure 2 Full pass of reverse cyclic coordinate descent on all block variables

Input: $\hat{\Lambda} = \{\hat{\lambda}_0, \ldots, \hat{\lambda}_{T-1}\}$, $U = \{\hat{u}_0, \cdots, \hat{u}_{T-1}\}$, $X = \{\hat{x}_0, \hat{x}_1, \cdots, \hat{x}_T\}$; MPC settings $\hat{A}$, $\hat{B}$, Q, R, $\hat{u}_{\min}$, $\hat{u}_{\max}$, $\hat{x}_{\min}$, $\hat{x}_{\max}$; parameter $\rho > 0$.

1. $\sigma \leftarrow 0$;
2. $\{\hat{x}_T, \sigma\} \leftarrow \underset{\hat{x}_T \in [\hat{x}_{\min}, \hat{x}_{\max}]}{\text{CCD}} \{\frac{1}{\rho}Q + I, -\hat{\lambda}_{T-1} - \hat{A}\hat{x}_{T-1} - \hat{B}\hat{u}_{T-1} - \hat{C}'\hat{W}\hat{r}_T, \sigma\}$;
3. $\{\hat{u}_{T-1}, \sigma\} \leftarrow \underset{\hat{u}_{T-1} \in [\hat{u}_{\min}, \hat{u}_{\max}]}{\text{CCD}} \{\frac{1}{\rho}R + \hat{B}'\hat{B}, \hat{B}'(\hat{\lambda}_{T-1} + \hat{A}\hat{x}_{T-1} - \hat{x}_T), \sigma\}$;
4. **for** $t = T-2, T-3, \ldots, 0$ **do**
   4.1. $\{\hat{x}_{t+1}, \sigma\} \leftarrow \underset{\hat{x}_{t+1} \in [\hat{x}_{\min}, \hat{x}_{\max}]}{\text{CCD}} \{\frac{1}{\rho}Q + I + \hat{A}'\hat{A}, -(\hat{\lambda}_t + \hat{A}\hat{x}_t + \hat{B}\hat{u}_t) + \hat{A}'(\hat{\lambda}_{t+1} + \hat{B}\hat{u}_{t+1} - \hat{x}_{t+2}) - \hat{C}'\hat{W}\hat{r}_t, \sigma\}$;
   4.2. $\{\hat{u}_t, \sigma\} \leftarrow \underset{\hat{u}_t \in [\hat{u}_{\min}, \hat{u}_{\max}]}{\text{CCD}} \{\frac{1}{\rho}R + \hat{B}'\hat{B}, \hat{B}'(\hat{\lambda}_t + \hat{A}\hat{x}_t - \hat{x}_{t+1}), \sigma\}$;
5. **end**.

Output: $\hat{U}$, $\hat{X}$, σ.

2.3.3 Preconditioning

Preconditioning is a common heuristic for improving the computational performance of first-order methods. The optimal design of preconditioners has been studied for several decades, but such a computation is often more complex than the original problem and may become prohibitive if it must be executed at runtime. Diagonal scaling is heuristic preconditioning that is very simple and often beneficial [34, 105]. In this chapter, we propose to make the change of state variables $\bar{x} = E\hat{x}$, where $E \in \mathcal{R}^{\hat{n}_x \times \hat{n}_x}$ is a diagonal matrix whose ith entry is

$$E_{i,i} = \sqrt{Q_{i,i} + \hat{A}'_{\cdot,i}\hat{A}_{\cdot,i}} \tag{2.12}$$

Procedure 3 Evaluation of CCD in Step 4.2 of Procedure 2

Input: $\hat{\lambda}_t$, $\hat{u}_t$, $\hat{x}_t$, $\hat{x}_{t+1}$; MPC settings $\hat{A}$, $\hat{B}$, R, $\hat{u}_{\min}$, $\hat{u}_{\max}$; parameter $\rho > 0$; update amount $\sigma \geq 0$.

1. $V \leftarrow \hat{\lambda}_t + \hat{A}\hat{x}_t + \hat{B}\hat{u}_t - \hat{x}_{t+1}$;
2. **for** $i = n_u, \ldots, 1$ **do**
   2.1. $s \leftarrow \frac{1}{\rho} R_{i,\cdot}\hat{u}_t + (\hat{B}_{\cdot,i})'V$;
   2.2. $\theta \leftarrow \left[\hat{u}_{t,i} - \frac{s}{\frac{1}{\rho}R_{ii} + (\hat{B}'\hat{B})_{ii}}\right]_{\hat{u}_{min,i}}^{\hat{u}_{max,i}}$;
   2.3. $\Delta \leftarrow \theta - \hat{u}_{t,i}$;
   2.4. $\sigma \leftarrow \sigma + \Delta^2$;
   2.5. $\hat{u}_{t,i} \leftarrow \theta$;
   2.6. $V \leftarrow V + \Delta\hat{B}_{\cdot,i}$;
3. **end**.

Output: $\hat{u}_t, \sigma$.

and replace the prediction model $\hat{x}_{t+1} = \hat{A}\hat{x}_t + \hat{B}\hat{u}_t$ by

$$\bar{x}_{t+1} = \bar{A}\bar{x}_t + \bar{B}\hat{u}_t$$

where $\bar{A} = E\hat{A}E^{-1}$ and $\bar{B} = E\hat{B}$. The weight matrix Q and constraints $[\hat{x}_{\min}, \hat{x}_{\max}]$ are scaled accordingly by setting $\bar{Q} = E^{-1}QE^{-1}$ and $\bar{x}_{\min} = E^{-1}\hat{x}_{\min}$, $\bar{x}_{\max} = E^{-1}\hat{x}_{\max}$.

2.3.4 Efficient coupling scheme between CD and AL method

We are now ready to couple CD and AL to solve the posed MPC problem (2.1) efficiently. We first note that updating $\hat{u}_t$ and $\hat{x}_{t+1}$ for all t involves computing a similar temporary vector V in Procedure 3. As V is, in fact, the next update of the dual vector Λ in Algorithm 1, we modify Procedure 3 as shown in Procedure 4. The overall solution method described in the previous subsections is summarized in Algorithm 5, which

Procedure 4 Modified Procedure 3 to efficiently couple CD and AL

Input: λ_t, $\hat{u}_t$; MPC settings $\hat{A}$, $\hat{B}$, R, $\hat{u}_{\min}$, $\hat{u}_{\max}$; parameter $\rho > 0$; update amount $\sigma \geq 0$.

1. **for** $i = n_u, \ldots, 1$ **do**
   1.1. $s \leftarrow \frac{1}{\rho} R_{i,\cdot} \hat{u}_t + (\hat{B}_{\cdot,i})' \lambda_t$;
   1.2. $\theta \leftarrow \left[\hat{u}_{t,i} - \frac{s}{\frac{1}{\rho} R_{ii} + (\hat{B}' \hat{B})_{ii}} \right]_{\hat{u}_{min,i}}^{\hat{u}_{max,i}}$;
   1.3. $\Delta \leftarrow \theta - \hat{u}_{t,i}$;
   1.4. $\sigma \leftarrow \sigma + \Delta^2$;
   1.5. $\hat{u}_{t,i} \leftarrow \theta$;
   1.6. $\lambda_t \leftarrow \lambda_t + \Delta \cdot \hat{B}_{\cdot,i}$;
2. **end**.

Output: $\hat{u}_t, \lambda_t, \sigma$.

we call CDAL. Note that the main update of the Lagrangian variables in Algorithm 5 is placed early in Step 3.1, unlike in Algorithm 1, due to the use of the proposed efficient coupling scheme. The AL (outer) iterations are executed for maximum N_{out} iterations, and the CD (inner) iterations for at most N_{in} iterations. The tolerances ϵ_{out} and ϵ_{in} are used to stop the outer and inner iterations, respectively. Algorithm 5 is matrix-free and library-free, and we could implement it in 90 lines of C code.

2.4 Numerical Examples

We test the performance of the CDAL solver against other solvers in two numerical experiments. The first one is the ill-conditioned AFTI-16 control problem [48, 11] based on LTI-MPC, used in the Model Predictive Control Toolbox for MATLAB [13]. The main goals of this experiment include investigating whether our proposed simple heuristic preconditioner, the reverse cyclic rule, and Nesterov's acceleration scheme are

Algorithm 5 Accelerated reverse cyclic CDAL algorithm for linear (or linearized) MPC

Input: primal/dual warm-start $U = \{\hat{u}_0, \hat{u}_1, \cdots, \hat{u}_{T-1}\}$, $X = \{\hat{x}_0, \hat{x}_1, \cdots, \hat{x}_T\}$, $\Lambda^{-1} = \Lambda^0 = \{\lambda_0, \lambda_1, \cdots, \lambda_{T-1}\}$; MPC settings $\{\hat{A}, \hat{B}, \hat{C}, W_y, W_u, W_{\Delta u}, \Delta u_{\min}, \Delta u_{\max}, u_{\min}, u_{\max}, x_{\min}, x_{\max}\}$; Algorithm settings $\{\rho, N_{\text{out}}, N_{\text{in}}\, \epsilon_{\text{out}}, \epsilon_{\text{in}}\}$

1. Obtain preconditioned $\bar{X} = \{\bar{x}_0, \cdots, \bar{x}_T\}$, $\bar{A}$, $\bar{B}$, $\bar{Q}$, $\bar{x}_{\min}, \bar{x}_{\max}$ according to Section 2.3.4
2. $\alpha_1 \leftarrow 1$; $\hat{\Lambda}^0 \leftarrow \Lambda^0$;
3. **for** $k = 1, 2, \cdots, N_{\text{out}}$ **do**
   - 3.1. **for** $t = 0, \ldots, T-1$ **do**
     - 3.1.1. $\lambda_t^k = \hat{\lambda}_t^{k-1} + \bar{A}\bar{x}_t + \bar{B}\hat{u}_t - \bar{x}_{t+1}$;
   - 3.2. **for** $k_{in} = 1, 2, \cdots, N_{\text{in}}$ **do**
     - 3.2.1. $U, \bar{X}, \sigma \leftarrow$ Procedure 2 with use of Procedure 4;
     - 3.2.2. **if** $\sigma \leq \epsilon_{\text{in}}$ **break** the loop;
   - 3.3. **if** $\|\Lambda^k - \hat{\Lambda}^{k-1}\|_2^2 \leq \epsilon_{\text{out}}$ **stop**;
   - 3.4. $\alpha_{k+1} \leftarrow \frac{1+\sqrt{1+4\alpha_k^2}}{2}$;
   - 3.5. $\hat{\Lambda}^k \leftarrow \Lambda^k + \frac{\alpha_k - 1}{\alpha_{k+1}}(\Lambda^k - \Lambda^{k-1})$;
4. Recover X from $\bar{X}$
5. **end**.

Output: U, X, Λ

helpful, and provide a detailed comparison with other solvers. The second experiment demonstrates the benefits of the construction-free property in LPV-MPC of a CSTR [100], in which the prediction model is obtained by linearizing a nonlinear model of the process at each sample step. The reported simulation results were obtained on a MacBook Pro with 2.7 GHz 4-core Intel Core i7 and 16GB RAM. Algorithm 5 is executed in MATLAB via a C-mex interface.

2.4.1 AFTI-16 Benchmark Example

The open-loop unstable linearized AFTI-16 aircraft model reported in [48, 11] is

$$\begin{cases} \dot{x} = \begin{bmatrix} -0.0151 & -60.5651 & 0 & -32.174 \\ -0.0001 & -1.3411 & 0.9929 & 0 \\ 0.00018 & 43.2541 & -0.86939 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} x \\ \qquad + \begin{bmatrix} -2.516 & -13.136 \\ -0.1689 & -0.2514 \\ -17.251 & -1.5766 \\ 0 & 0 \end{bmatrix} u \\ y = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x \end{cases}$$

The model is sampled using zero-order hold every 0.05 s. The input constraints are $|u_i| \leq 25^\circ, i = 1, 2$, the output constraints are $-0.5 \leq y_1 \leq 0.5$ and $-100 \leq y_2 \leq 100$. The control goal is to make the pitch angle y_2 track a reference signal r_2. In designing the MPC controller we take $W_y = \mathrm{diag}([10,10])$, $W_u = 0$, $W_{\Delta u} = \mathrm{diag}([0.1, 0.1])$, and the prediction horizon is $T = 5$.

To investigate the effects of the three techniques (the reverse cyclic rule, acceleration, and preconditioning) that we have introduced to improve the efficiency of the CDAL algorithm, we performed closed-loop simulations on eight schemes with fixed $\rho = 1$. These are: 0-CDAL, the basic scheme, without acceleration and reverse cyclic rule; R-CDAL, the scheme with the Reverse cyclic rule; A-CDAL, the Accelerated scheme; AR-CDAL, the Accelerated scheme with the Reverse cyclic rule, and their respective schemes with preconditioner, namely P-0-CDAL, P-R-CDAL, P-A-CDAL, and finally CDAL, that includes all the proposed

techniques. The stopping criteria are defined by $\epsilon_{\text{in}} = 10^{-6}$, $\epsilon_{\text{out}} = 10^{-4}$, and N_{out}, N_{in} are set to the large enough value 5000 in order to guarantee good-quality solutions. The computational load associated with the above schemes is listed in Table 2.1, in which the last column, "cost", represents the closed-loop performance, and adopts the average MPC cost $\frac{1}{T}\sum_{t=0}^{T-1} \|W_y\left(y_{t+1} - r_{t+1}\right)\|_2^2 + \left\|W_u\left(u_{t+1} - u_{t+1}^r\right)\right\|_2^2 + \|W_{\Delta u}\Delta u_t\|_2^2$ over the closed-loop simulation time T, showing that it is almost the same for all schemes. The associated closed-loop trajectories are reported in Figure 2.1, which shows that the pitch angle correctly tracks the reference signal from 0° to 10° and then back to 0° and that both the input and output constraints are satisfied.

Since each MPC execution requires different numbers of inner and outer iterations, the average ("avg") and maximum ("max") number of iterations (or CPU time) are computed over the entire closed-loop execution. It can be observed that the maximum and average number of inner-loop iterations of R-CDAL are smaller than that of 0-CDAL (especially the maximum number), while their outer-loop iterations are almost the same, which shows that the reverse cyclic rule provides a significant improvement. Although A-CDAL has fewer outer-loop iterations, it has more inner-loop iterations than 0-CDAL on average. Therefore it does not result in a significant reduction in total computation time. We can see that AR-CDAL achieves fewer iterations both in the inner loop and outer loop and has better average and worst-case computation performance. It can also be seen from Table 2.1 that preconditioning significantly reduces the number of outer-loop iterations.

Next, we investigate the effect on computation efficiency of parameter ρ, that we expect to tend to trade off feasibility versus optimality. In particular, we expect larger values of ρ to favor feasibility, i.e., provide more inner-loop iterations and fewer outer-loop iterations, and vice versa. The computational performance results obtained by performing closed-loop simulations using the final **CDAL** algorithm for different values of ρ between 0.01 and 1 are listed in Table 2.2. When the parameter value is between 0.01 and 0.1, the CDAL algorithm has a very similar computational burden.

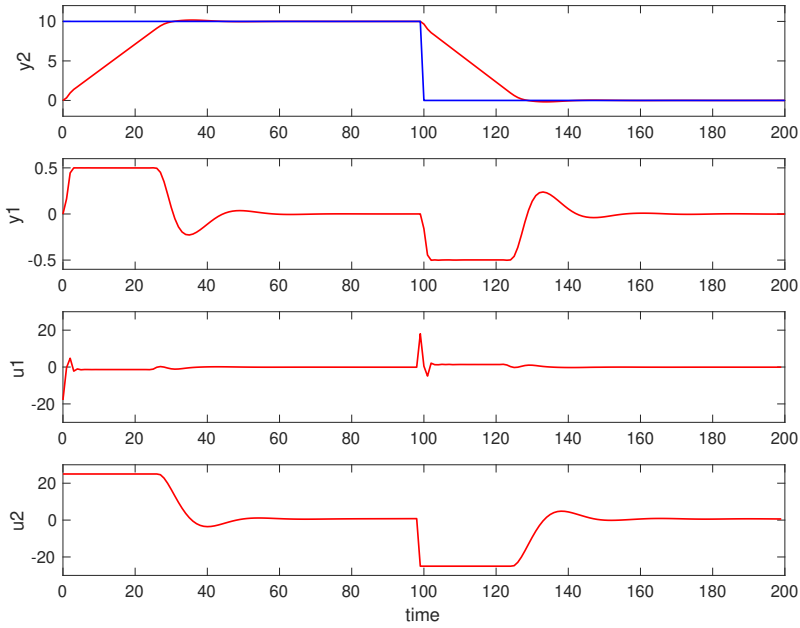


Figure 2.1: Linear AFTI-16 closed-loop performance

Table 2.1: Computational performance of different schemes

method	sum of inner iters		outer iters		time (ms)		cost
	avg	max	avg	max	avg	max	
0-CDAL	8577	79615	339	2104	4.9	55.3	42.3
R-CDAL	7298	72693	340	2103	4.3	53.2	42.5
A-CDAL	7437	57026	45	297	4.0	41.1	42.5
AR-CDAL	6207	51884	44	205	3.8	39.5	42.5
P-0-CDAL	3467	13386	33	171	2.1	11.4	42.5
P-R-CDAL	1757	13430	33	171	1.0	10.9	42.5
P-A-CDAL	3299	12161	13	60	1.7	9.7	42.5
CDAL	1543	12508	13	60	0.85	9.5	42.5

To further illustrate the efficiency of CDAL, Table 2.2 also lists the results obtained by using other solvers. Here the fastMPC solver is also a construction-free solver which provides a free C-mex code. We also made comparisons with the μAO-MPC solver v1.0.0-beta [119], which is based on an augmented Lagrangian method together with Nesterov's gradient method. The μAO-MPC differs from CDAL in the way the subproblems are solved, and the outer loop not involving an acceleration scheme. The state-of-the-art first-order method for QP, the OSQP solver v0.6.2 [103], and MATLAB's built-in QP solver (quadprog) are also used for comparison. For a fair comparison, each solver setting is chosen to at least ensure each share the same objective cost and constraint violation. When the parameter ρ of the CDAL is 0.01, the CDAL is faster than the other solvers. Regarding the μAO-MPC, OSQP, and quadprog solver, we split between QP problem construction time (including the required matrix factorizations) and pure solution time. Note that in this case, the controller is LTI-MPC, and hence the MPC problem construction and matrix factorizations required by these non-construction-free solvers can be performed offline. On the other hand, in the case of LPV-MPC problems, the total computation time would be spent online, and the embedded code would also include routines for problem construction and matrix factorization functions. Instead, CDAL does not require any construction or factorizations, thus making the solver very lean and fast also in a time-varying MPC setting, as investigated next.

2.4.2 Nonlinear CSTR Example

To illustrate the performance of CDAL when the linear MPC formulation (2.1) changes at runtime, we consider the control of the CSTR system [100], described by the continuous-time nonlinear model

$$\begin{array}{rcl} \frac{dC_A}{dt} & = & C_{A,i} - C_A - k_0 e^{\frac{-EaR}{T}} C_A \\ \frac{dT}{dt} & = & T_i + 0.3T_c - 1.3T + 11.92 k_0 e^{\frac{-EaR}{T}} C_A \\ y & = & C_A \end{array} \tag{2.13}$$

where C_A is the concentration of reagent A, T is the temperature of the reactor, and $C_{A,i}$ is the inlet feed stream concentration, which is assumed

Table 2.2: Computational load of CDAL with different values of ρ and comparison with other solvers

Solver	solver setting	time (ms) avg	time (ms) max	cost
CDAL	$\rho = 1$	0.85	9.5	42.561
	$\rho = 0.5$	0.72	7.1	42.590
	$\rho = 0.2$	0.53	4.2	42.612
	$\rho = 0.1$	0.47	3.8	42.619
	$\rho = 0.05$	0.42	3.3	42.618
	$\rho = 0.01$	0.41	3.2	42.618
fastMPC	$maxit = 5, k = 0.1$	0.54	4.2	42.627
μAO-MPC	$\mu = 0.05$	7.0*	68.1*	42.627
	in_iter=100,ex_iter=100	8**	69**	
OSQP	$N = 5000, \epsilon = 10^{-6}$	0.6*	10.1*	42.627
		1.5**	13.8**	
quadprog	default	10.3*	20.6*	42.622
		11**	22**	

* : pure solution time, without including matrix factorization
** : total time (MPC construction + solution)

to have the constant value 10.0 kgmol/m^3. The process disturbance comes from the inlet feed stream temperature T_i, which has slow fluctuations represented by $T_i = 298.15 + 5\sin(0.05t)\ K$. The manipulated variable is the coolant temperature T_c. The constants $k_0 = 34930800$ and $EaR = -5963.6$ (in MKS units).

The initial state of the reactor is at a low conversion rate, with $C_A = 8.57$ kgmol/m^3, $T = 311$ K. The control objective is to adjust the reactor state to a high reaction rate with $C_A = 2$ kgmol/m^3. The MPC controller manipulates the coolant temperature T_c to track a concentration reference as well as reject the measured disturbance T_i. Due to its nonlinearity, the model in (2.13) is linearized online at each sampling step:

$$\frac{dx}{dt} \approx f(x_t, u_{t-1}, p) + \left.\frac{\partial f}{\partial x}\right|_{x_t, u_{t-1}, p} (x - x_t) + \left.\frac{\partial f}{\partial u}\right|_{x_t, u_{t-1}, p} (u - u_{t-1})$$

where $f(x, u, p)$ is the mapping defined in (2.13) for $x = [C_A\ T]'$, $u = T_c$, $p = [C_{A,i}\ T_i]'$. By setting $A_c = \left.\frac{\partial f}{\partial x}\right|_{x_t, u_{t-1}, p}, B_c = \left.\frac{\partial f}{\partial u}\right|_{x_t, u_{t-1}, p}, e_c = f(x_t, u_{t-1}, p) - A_c x_t - B_c u_{t-1}$, we get the following linearized continuous-time model

$$\frac{d}{dt}x = A_c x + B_c u + e_c$$

We use the forward Euler method with sampling time $T_s = 0.5$ minutes to obtain the following discrete-time model

$$x_{t+1} = A_d x_t + B_d u_t + e_d$$

where $A_d = I + T_s A_c, B_d = T_s B_c, e_d = T_s e_c$. Although held constant over the prediction horizon, clearly matrices A_d, B_d and the offset term e_d change at runtime, which makes the controller an LPV-MPC. Regarding the performance index, we choose weights $W_y = 1$, $W_u = 0$, $W_{\Delta u} = 0.1$. The physical limitation of the coolant jacket is that its rate of change ΔT_c is subject to the constraint $[-1, 1]$ K when considering the sampling time $T_s = 0.5$ minutes. The prediction horizon is $T = 10$ steps.

We again compare CDAL with fastMPC, μAO-MPC, OSQP, and quadprog solvers in the LPV-MPC setting described above. CDAL is run with $\epsilon_{\text{in}} = 10^{-6}$, $\epsilon_{\text{out}} = 10^{-4}$, $\rho = 0.01$, and $N_{\text{out}} = N_{\text{in}} = 5000$. For a fair

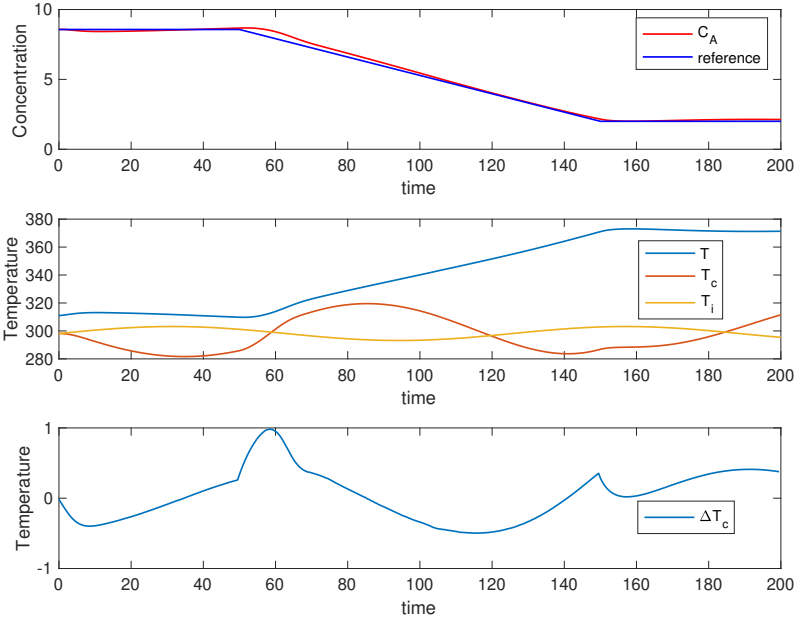


Figure 2.2: Nonlinear CSTR closed-loop performance

comparison, each solver setting is chosen to at least ensure each share the same objective cost and constraint violation. The closed-loop simulation results of CDAL and other solvers almost coincide and are plotted in Figure 2.2, from which it can be seen that C_A tracks the reference signal well, and the fluctuation of T_i is effectively suppressed.

The computational load and closed-loop performance associated with CDAL and other solvers are reported in Table 2.3. In this successive linearization-based MPC example, the problem-construction time is comparable with the problem-solving time for the results of non-construction-free solvers. If we only compare the solution time, CDAL is faster than other solvers except for OSQP, but indeed the MPC construction time must be included for comparison, which leads to CDAL being faster

than OSQP. Because of the construction-free, matrix-free, and library-free features, CDAL has an advantage in industrial embedded deployment when the optimization problem associated with MPC is constructed online, and this operation has a cost that is comparable to the solution time.

Table 2.3: Computational performance of CDAL and other solvers

Solver	solver setting	time (ms) avg	time (ms) max	cost
CDAL	$\rho, \epsilon_{in}, \epsilon_{out} = 0.01, 10^{-6}, 10^{-4}$	0.3	0.6	0.02202
fastMPC	maxit=5,$k = 0.1$	0.5	7.2	0.030170
μAO-MPC	$\mu = 0.01$	1.4*	10.1*	0.02202
	in_iter=100,ex_iter=10	2.1**	15.2**	
OSQP	default	0.15*	0.37*	0.02219
		0.6**	5.5**	
quadprog	default	1.6*	9.7*	0.02219
		1.8**	13.3**	

* : solution time
** : MPC construction time + solution time

2.5 Conclusion

This chapter has proposed a construction-free, matrix-free, and library-free MPC solver, based on a cyclic coordinate-descent method in the augmented Lagrangian framework. We showed that the method is efficient and competes with other existing methods, thanks to the use of a reverse cyclic rule, Nesterov's acceleration, a simple heuristic preconditioner, and an efficient coupling scheme. Compared to many QP solution methods proposed in the literature, CDAL avoids constructing the QP problem, which makes it particularly appealing for some scenarios in which its online construction is required and has a comparable computation time to solving itself.

Chapter 3

A rapid-prototype MPC tool based on gPROMS platform

3.1 Introduction

Developing a novel chemical or pharmaceutical process can take decades and hundreds of millions of dollars, and first-principle-based modeling or digital-twin technology can speed up development and reduce costs. The gPROMS' digital-twin technology [102] provides powerful and easy-to-use process modeling capabilities, rich physical property databases and model libraries, integrated experimental design, parameter estimation, and optimization capabilities.

During the process design phase, the process sometimes requires a controller to maintain stability or simultaneously is explored together with the controller design. The process often appears as a multi-input multi-output (MIMO) plant, and using a multi-PID controller scheme would be cumbersome or complicated for relatively inexperienced process engineers. Model predictive control (MPC) is developed for controlling MIMO plants subject to constraints [69], and MPC has been widely used in diverse industrial areas, such as process [88], and aerospace [25],

power electronics [33], etc. In the design workflow of an MPC controller for a physical plant, firstly, with the help of professional process modeling software such as gRPOMS, Aspen Plus [5], or general-purpose modeling software MATLAB/Simulink [24], a high-fidelity model based on the first-principle is established. The first-principle-based model could be as complex as possible, like introducing distributed parameter model to describe spatial-time relationships, which is usually a large mixed system of integral, partial differential, and algebraic equations (IPDAEs) [77]. The first-principle-based model can be calibrated from the experimental data with the use of parameter estimation or validated. The main objective of developing a first-principle-based model is to explore the optimal design space of the physical process, and another objective is to validate the designed controller scheme via closed-loop simulations. A key part of MPC design is how to obtain a simplified control-oriented prediction model of the physical process to predict its likely evolution. One approach is to use system identification methods to obtain a simplified model from experimental data, such as the open-loop step-response data. Getting open-loop experimental data is sometimes expensive or even forbidden for safety reasons. As an alternative, the simulation data of first-principle-based models can be used for system identification. Another approach is based on the linearization of first-principle-based model at some operating points.

3.1.1 Related works

In the PAROC framework [87] developed by Pistikopoulos et.al, simulation data are generated from first-principle-based gPROMS models and then used by MATLAB's system identification and model reduction toolbox to obtain multiple approximate linear state-space models. Based on these linear state-space models, explicit/multi-parametric MPC algorithms [10, 14] are implemented on MATLAB side. The closed-loop simulations are performed via the gPROMS ModelBuilder® tool gO:MATLAB, which couples the computation of the gPROMS and MATLAB. Although the PAROC framework takes advantage of MATLAB's power in MPC de-

sign and gPROMS' benefits in professional process modeling, the PAROC framework obviously does not take full advantage of the mathematical information of existing first-principle-based gPROMS models. The control-oriented model obtained by their method is still a black-box model and does not inherit the interpretability from the first-principle-based gPROMS model. By directly linearizing the first-principle-based gPROMS model, we can obtain interpretative simplified control-oriented models. In the previous *gNLMPC* [82], the nonlinear gPROMS model is directly used to construct a nonlinear MPC problem, which is solved by using the built-in numerical optimization capabilities of the gPROMS platform. Clearly, the *gNLMPC* would have good closed-loop performance but at a high online computation cost. In fact, linear model-based MPC is successful in the vast majority of process industry applications even though many manufacturing processes are inherently nonlinear [89].

3.1.2 Contribution

This chapter presents the development of a rapid-prototype MPC tool based on the gPROMS platform. The gPROMS-MPC tool employs the online successive linearization strategy to linearize a general nonlinear first-principle-based gPROMS model, based on the supporting automatic differentiation capability of gPROMS model. The gPROMS-MPC tool implements our previous construction-free CDAL algorithm in Chapter 2 and the widely-used open-source *qpOASES* v3.2 algorithm [29] in the gPROMS platform (gPROMS Process v2.2.2), based on sparse and condensed MPC formulations, respectively.

The online successive linearization strategy is an effective solution to deal with nonlinear MPC problems [18, 27], by achieving a good trade-off between computational cost and closed-loop control performance. In addition to linearization at each sampling time, the gPROMS-MPC tool allows users to decide when to perform the linearization at a lower sampling frequency or at some specific points to derive one or multiple invariant linear state-space models in embedded MPC design. The embedded MPC design is the final step of the MPC design workflow, that

is, evaluating whether the chosen one or multiple invariant linear state-space models meet the closed-loop performance requirements under the computation limits from embedded platforms. Since our construction-free CDAL algorithm is also matrix-free and library-free, which makes the gPROMS-MPC tool being suitable for embedded code generation. The gPROMS-MPC is competent in completing all the workflow only on the gPROMS platform.

3.2 MPC problem formulation

gPROMS models typically comprise mixed sets of non-linear differential and algebraic equations, which can be written in the form

$$f(\dot{x}, x, y, u) = 0 \tag{3.1}$$

where $x(t)$ and $y(t)$ are the sets of differential and algebraic variables, respectively (both of which are unknowns to be determined by the gPROMS simulation), while $\dot{x}(t)$ are the derivatives of $x(t)$ with respect to time t, u is the set of input variables that are given functions of time. Now consider the current time point (x_c, y_c, u_c) on the simulation trajectory. A linear model can be obtained by linearising the Eqn (3.1) at this point,

$$\frac{\partial f}{\partial x}\delta x + \frac{\partial f}{\partial \dot{x}}\dot{\delta x} + \frac{\partial f}{\partial y}\delta y + \frac{\partial f}{\partial u}\delta u = 0 \tag{3.2}$$

where $\delta x = x - x_c$, $\delta y = y - y_c$ and $\delta u = u - u_c$.

3.2.1 Linearized State-Space model with minimal subset of differentials

Most first-principle-based gPROMS models involve thousands of equations and variables. When performing linearization, by specifying: a set of input variables U, and a set of output variables Y, gPROMS can provide the automatical variables-reduction technique according to variables dependencies. gPROMS provides the following linearized model

$$\begin{aligned} \dot{\delta X} &= A\delta X + B\delta U \\ \delta Y &= C\delta X + D\delta U \end{aligned} \tag{3.3}$$

where the state variables X are determined automatically by gPROMS' built-in *Linearise* as the minimal subset of x that is necessary to express the effects of the specified inputs U on the specified outputs Y via relationships of the above eqn (3.2).

Before being applied to MPC problem, the continuous-time state-space model (3.3) requires to be discretized to discrete-time form. Our MPC tool currently utilizes the one-step Euler discretization method. Given a sampling time T_s, we have the following discrete-time state-space model,

$$\begin{aligned} \delta X_{t+1} &= A_d \delta X_t + B_d \delta U_t + e \\ \delta Y_t &= C_d \delta X_t + D_d \delta U_t \end{aligned} \tag{3.4}$$

where $A_d = I + T_s A$, $B_d = T_s B$, $C_d = C$, $D_d = D$, $e = T_s f(x_c, \dot{x}_c, y_c, u_c)$. Our MPC tool considers an MPC tracking problem, thus a Δ-formulation augmented state-space model is used,

$$\begin{aligned} \hat{X}_{t+1} &= \hat{A}\hat{X}_t + \hat{B}\hat{U}_t + \hat{e} \\ \Delta Y_t &= \hat{C}\hat{X}_t \end{aligned}$$

where $\hat{X}_t = \begin{bmatrix} \delta X_t \\ \delta U_{t-1} \end{bmatrix}$, $\hat{U}_t = \Delta \delta U_t$ denotes the increment of δU_t, $\hat{A} = \begin{bmatrix} A_d & B_d \\ 0 & I \end{bmatrix}$, $\hat{B} = \begin{bmatrix} B_d \\ I \end{bmatrix}$, $\hat{e} = \begin{bmatrix} e \\ 0 \end{bmatrix}$, $\hat{C} = \begin{bmatrix} C_d & 0 \end{bmatrix}$, and D is a zero matrix and thus eliminated. Our MPC tool considers the case that box constraints are subject to the increased input, the input, and the output. The MPC tracking problem is shown below,

$$\begin{aligned} \min \quad & \frac{1}{2} \sum_{t=0}^{T-1} \|(\delta Y_{t+1} - \delta r)\|^2_{W^y} + \left\| \Delta \hat{U}_t \right\|^2_{W^{\Delta u}} \\ \text{s.t.} \quad & \hat{X}_{t+1} = \hat{A}\hat{X}_t + \hat{B}\Delta\hat{U}_t + \hat{e}, t = 0, 1, \ldots, T-1 \\ & \delta Y_{t+1} = \hat{C}\hat{X}_{t+1}, t = 0, \ldots, T-1 \\ & \Delta U_{\min} \le \Delta \hat{U}_t \le \Delta U_{\max}, t = 0, \ldots, T-1 \\ & U_{\min} - U_c \le \delta U_t \le U_{\max} - U_c, t = 0, \ldots, T-1 \\ & Y_{\min} - Y_c \le \delta Y_t \le Y_{\max} - Y_c, t = 1, \ldots, T \\ & \hat{X}_0 = 0 \end{aligned} \tag{3.5}$$

where $\delta r = r - Y_c$ and r denotes the desired tracking set-points. $W^y \succ 0$ and $W^{\Delta u} \succ 0$ denote the weights matrices of the output and input, respectively. Note that the initial values of $\hat{X}$ are zeros. $[\Delta U_{\min}, \Delta U_{\max}]$, $[U_{\min}, U_{\max}]$ and $[Y_{\min}, Y_{\max}]$ denote the box-constraints of the specified increased input, input, and output, respectively.

3.2.2 Condensed and Sparse formulation

Solving the above MPC tracking problem (3.5) requires constructing it into a quadratic programming (QP) problem. There are two kinds of MPC-to-QP construction, one is the condensed construction that eliminates the states, and another is the sparse construction that keeps the states in the resulting QP problem. Eliminating all states from the MPC problem (3.5) to yield a smaller-scale, condensed QP problem of the form:

$$\begin{aligned} \min \quad & \frac{1}{2} z' H_c z + h_c' z \\ \text{s.t.} \quad & g_c^l \le G_c z \le g_c^u \\ & z_c^l \le z \le z_c^u \end{aligned} \tag{3.6}$$

where the vector of decision variables only comprises the increase control inputs, $z \overset{\text{def}}{=} \left[\Delta \hat{U}_0', \Delta \hat{U}_1', \cdots, \Delta \hat{U}_{T-1}'\right]'$, and its Hessian matrix H_c is dense.

The problem (3.5) can also yield a structured QP problem by keeping the discrete-time state-space model as equality constraints,

$$\begin{aligned} \min \quad & \frac{1}{2} z' H_s z + h_s' z \\ \text{s.t.} \quad & B_s z = b_s \\ & g_s^l \le G_s z \le g_s^u \\ & z_s^l \le z \le z_s^u \end{aligned} \tag{3.7}$$

where the vector of decision variables comprise both the states and the increased control inputs $z \overset{\text{def}}{=} \left[\Delta \hat{U}_0', \hat{X}_1', \Delta \hat{U}_1', \hat{X}_2', \cdots, \Delta \hat{U}_{T-1}', \hat{X}_T'\right]'$, and and its Hessian matrix H_s is sparse.

Whether the sparse formulation (3.7) or the compressed formulation (3.6) is more computationally advantageous, mainly depends on the number of the states n_x, the control input n_u and the length of the prediction horizon T [53]. The condensed formulation (3.6) is obviously preferable when the number of states is large, which would often be the case when spatial-temporal equations are involved. If the ratio $\frac{n_x}{n_u}$ is small and the prediction horizon is long, the sparse formulation (3.7) is often more efficient. Moreover, the choice may also depend on whether the condensing procedure (the state elimination) requires to be done once offline (like the LTI MPC case) or performed online, as is the case where the gPROMS-MPC tool utilizes the online successive linearization strategy for nonlinear systems. Thus, the gPROMS-MPC tool implements the two formulations to allow users to choose.

3.3 Implementation of our MPC tool

The implementation of the gPROMS-MPC tool is based on the gPROMS Foreign Process Interface, which is a C/C++ interface to allow developers' custom code to interact dynamically with the gPROMS model at runtime. Although many optimization algorithms can solve MPC problems, the online successive linearization strategy used in the gPROMS-MPC tool clearly encourages us to adopt our previous construction-free CDAL algorithm in Chapter 2 and a parametric active-set *qpOASES* v3.2 algorithm [29] from an efficiency perspective. The CDAL and *qpOASES* solve the sparse (3.7) and condensed (3.6) QP formulations from the MPC problem, respectively.

The CDAL is based on the coordinate descent (CD) and augmented Lagrangian (AL) methods. The outer loop involves the accelerated AL iteration, and the inner loop (solving the AL subproblem) uses the CD method, exploiting the structure of the MPC problem. And an efficient CD-AL coupling scheme and preconditioner are proposed in the implementation of the CDAL to speed up computation. Its most notable feature is that the CDAL directly solves the MPC formulation (3.5) without resorting to an explicit MPC-to-QP construction. Clearly, the *construction-*

free feature is suitable for the online successive linearization strategy used in the gPROMS-MPC tool to avoid the online construction cost. Especially in some online successive linearization based MPC problems, its online construction cost is comparable to the online solving itself thanks to warm-starting, gradually changing set-points, and slowly updating dynamics. In addition, our CDAL is also *matrix-free* (avoids multiplications and factorizations of matrices) and *library-free* (without any library dependency), and its 90-line C-code implementation makes it competent in the work of embedded MPC code generation. The detailed description and implementation of the CDAL algorithm are presented in Chapter 2.

Although gPROMS provides minimal state-space realization, its derived linear state-space model still involves large state dimensions for some plants, especially when it involves spatial-time equations. In these cases, solving the condensed QP formulation associated with MPC is a better choice. The condensed QP formulation (3.6) is solved by calling the *qpOASES* package, which is an open-source C++ implementation of the online parametric active set strategy [29]. The *qpOASES* not only supports warm-starting strategy and also supports solving QPs with time-varying Hessian matrices, which makes it suitable for the online successive linearization-based MPC problems. In addition to integrating the called *qpOASES* package for efficiently online solving QPs, the gPROMS-MPC tool efficiently implements the online condensed MPC-to-QP construction. Herein the main condensed equation listed in (3.8),

$$\begin{bmatrix} \delta Y_1 \\ \delta Y_2 \\ \vdots \\ \delta Y_T \end{bmatrix} = G \begin{bmatrix} \hat{X}_1 \\ \hat{X}_2 \\ \vdots \\ \hat{X}_T \end{bmatrix} = GM \begin{bmatrix} \hat{U}_0 \\ \hat{U}_1 \\ \vdots \\ \hat{U}_{T-1} \end{bmatrix} + Gm \tag{3.8}$$

where $M = \begin{bmatrix} \hat{B} & 0 & \cdots & 0 \\ \hat{A}\hat{B} & B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \hat{A}^{T-1}\hat{B} & \hat{A}^{T-2}\hat{B} & \cdots & \hat{B} \end{bmatrix}$,

$$m = \begin{bmatrix} e \\ \hat{A}e + e \\ \vdots \\ \hat{A}^{T-1}e + \hat{A}^{T-2}e + \cdots + e \end{bmatrix}, G = \begin{bmatrix} \hat{C} & 0 & \cdots & 0 \\ 0 & \hat{C} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \hat{C} \end{bmatrix}.$$

Then, the dense hessian matrix H_c and gradient vector h_c of the condensed QP formulation (3.6) are

$$H_c = (GM)' \begin{bmatrix} W^y & 0 & \cdots & 0 \\ 0 & W^y & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & W^y \end{bmatrix} (GM) + \begin{bmatrix} W^{\Delta u} & 0 & \cdots & 0 \\ 0 & W^{\Delta u} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & W^{\Delta u} \end{bmatrix}$$

$$h_c = (GM)' \begin{bmatrix} W^y & 0 & \cdots & 0 \\ 0 & W^y & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & W^y \end{bmatrix} \left(Gm - \begin{bmatrix} \delta r \\ \delta r \\ \vdots \\ \delta r \end{bmatrix} \right)$$

To exploit the above structures, the gPROMS-MPC tool implements an efficient condensing procedure, which shares the same idea with the Andersson et al. work [2] via the use of OpenBLAS v0.3.20 [117].

3.4 Application example of Flash-Separation

The developed gPROMS-MPC tool worked well on some MPC benchmark examples in the literature, such as the ill-conditioned AFTI-16 [11] and nonlinear CSTR [100], which demonstrates the effectiveness of its functionality. To further validate the correctness of the gPROMS-MPC tool when applied to the commercial built-in gPROMS Model Library (gML), this section presents the MPC controller design for a flash separation of a mixture of eight components. To investigate the performances of different control schemes, three control schemes are used for comparison, namely multi-PID scheme, successive-linearization-based MPC (SL-MPC) scheme, and linear MPC scheme. The multi-PID scheme is built as shown in Figure 3.1, which involves one source model (Feed_NG), one pipeline model (P1), one flash drum model (Drum), two value models (V1 and V2), two sink models (gas_product and liquid_product), one

stream analyzer model (S1) that only pass the mass flowrate and three continuous-time PID controller models (liquid_control and flow_control are connected as the cascade PID). The parameters of the flowsheet with PID scheme are listed in Table 3.1.

In the SL-MPC and linear MPC schemes, the MPC controller is used to replace three PID models and the stream analyzer model, and other model parameters remain the same. The MPC controller directly manipulates the stem position of two valves simultaneously to regulate the pressure and liquid level fraction of the Drum, namely a two-input-two-output plant. As for the MPC settings, the prediction horizon is 10, the sampling time is 0.02 s, the constraints come from the physical constraints of the two values V1 and V2, that is, box-constraints $[0, 1]$ The cost weights of the increased input and the output are $W^{\Delta u} = diag([1, 1])$ and $W^{y} = diag([100, 100])$, respectively. The difference between the SL-MPC and linear MPC scheme is that the former updates the linear state-space model at each sampling time via linearization along the trajectory, and the latter only utilizes one linear state-space model that is linearized at the initial point. As we mentioned in Section 3.1, even though many manufacturing processes are inherently nonlinear, the performance of linear MPC scheme often satisfies requirements in the vast majority of process industry applications. Thus, obtaining a good linear state-space model, with good close-loop performance in operating ranges, is the key part, which is also the most time-consuming and labor-intensive step in the whole MPC design workflow. That is what the gPROMS-MPC tool does, being an MPC design platform based on gPROMS' powerful modeling capability.

The simulation scenario is to track the desired setpoints of liquid level and pressure fraction from the initial steady-state condition in which pressure $= 69$ bar and liquid level fraction $= 0.5$. The simulation time is from 0 to 300 seconds, and the desired setpoints of liquid level fraction are 0.4, 0.5, and 0.4 at every 100 second, and the desired setpoint of pressure keeps at 69 bar all the time. The comparison simulation results are shown in Figure 3.2(a), 3.2(b), 3.2(c) and 3.2(d), which are liquid level fraction, pressure, liquid mass flowrate and vapour mass flowrate, re-

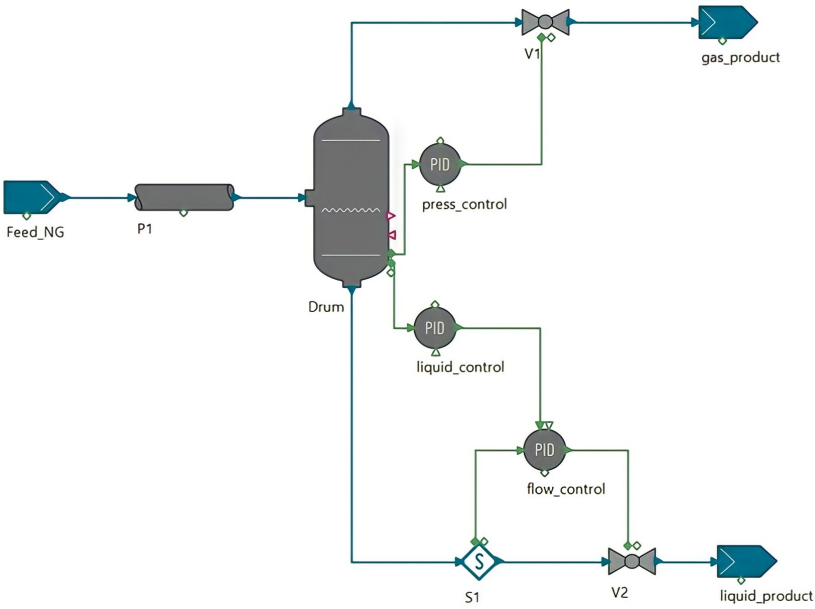
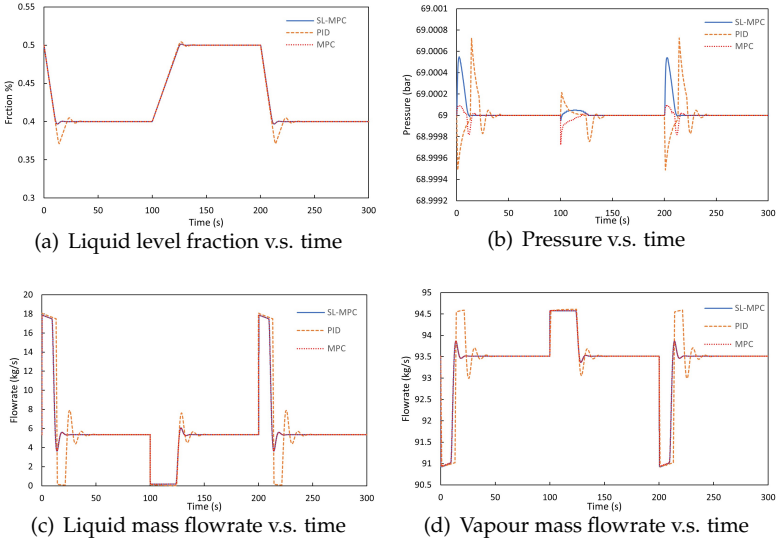


Figure 3.1: Flash separation flowsheet under the continuous-time PID control scheme

Table 3.1: The settings of flash separation with PID scheme

model name	model parameters
Feed_NG	components name and mass fraction (kg/kg): [METHANE=0.5202], [ETHANE=0.1504], [PROPANE=0.1180], [I-BUTANE=0.0167], [N-BUTANE=0.0462], [I-PENTANE=0.0098] [N-PENTANE, 0.0090],[N-HEXANE, 0.0006] pressure = 70 bar, Vapour mass fraction = 0.95 kg/kg
P1	Inner diameter = 0.152606 m, length = 10 m Friction facotr correlation: Fully turbulent
Drum	Shape: Drum, Orientation: Vertical, Diameter = 0.5 m, Volume = 3 m^3 Include effect of hydrostatic pressure
V1	Phase: Vapour, Flow relation: Fisher universal gas sizing equation, Flow coefficient = 200000 $scfh^{-1}psi^{-1}$, Recovery factor = 34.8, Leakage fraction = 0, Inherent characteristic: Linear
V2	Phase: Liquid, Flow relation: Fisher equation Flow coefficient = 100 $scfh^{-1}psi^{-1}$, Leakage fraction = 0 Inherent characteristic: Linear,
pressure_control	Controller action: Direct, gain = 1000, Integral time constant = 5, initial setpoint: 69 bar Process variable: Pressure, Manipulated variable: Stem position
liquid_control	Controller action: Direct, gain = 10, Integral time constant = 1, initial setpoint: 0.5 Process variable: Liquid level fraction, Manipulated variable: mass flowrate
flow_control	Controller action: Reverse, gain = 100, Integral time constant = 10, setpoint: from liquid_control Process variable: Mass flowrate, Manipulated variable: Stem position

spectively. The results show that the closed-loop control performance of the SL-MPC and MPC scheme is better than the PID scheme, having less integral overshoot and oscillation, and the SL-MPC and MPC scheme have almost the same tracking performance, which can be explained by the fact that the nonlinearity of the flash-separation example is not strong in operating ranges. It shows that the linearized model at the initial equilibrium point can be used in embedded MPC code generation.



(a) Liquid level fraction v.s. time

(b) Pressure v.s. time

(c) Liquid mass flowrate v.s. time

(d) Vapour mass flowrate v.s. time

Figure 3.2: Comparison results of PID, SL-MPC, and MPC scheme in the flash-separation example

3.5 Conclusion

This chapter develops the gPROMS-MPC tool, which not only interacts directly with the gPROMS first-principle-based model for closed-loop simulations but also utilizes its mathematical information to derive the linearized state-space model for MPC design. The gPROMS-MPC tool adopts the online successive linearization-based MPC to handle general

nonlinear systems and also allows users to choose when to linearized, such as at some specific points, to obtain one or multiple linear models for later embedded MPC design. The gPROMS-MPC tool implements our previous construction-free CDAL and the online parametric active-set *qpOASES* algorithm. The CDAL is also matrix-free and library-free, which provides benefits in embedded C-code generation.

The derived linear model from the gPROMS is a state-space formulation (input-state-output), which would suffer high state-dimension issues and require a state estimation algorithm in embedded MPC deployment. They could be addressed to some extent by using the model reduction technique to reduce state dimension, but we recommend transforming it into an equivalent input-output ARX model as illustrated in Chapter 5, to avoid designing a state estimation algorithm. And the resulting ARX-based MPC problem can be solved by our extended CDAL-ARX algorithm in the next Chapter 4.

Chapter 4

A construction-free CDAL algorithm for embedded ARX-based MPC

4.1 Introduction

In earlier MPC developments, some methods shared the same receding horizon control idea, under different names. The Model Predictive Heuristic Control (MPHC) [94], the Model Algorithmic Control (MAC) [91] used a finite impulse response model, the Dynamic Matrix Control (DMC) employed a truncated step-response model [23], and the Generalized Predictive Control (GPC) involved a transfer function model [22]. As the MPC field has grown, state-space (SS) models replaced input-output (I/O) models, and most MPC theory is based on SS formulations [68].

However, in industrial control applications, MPC based on input-output models, such as the autoregressive model with exogenous terms (ARX) model, may still be preferable [88], for two main reasons: (1) there is no need for a state-observer; (2) I/O models are easier to identify and to adapt online (such as using recursive least-squares or Kalman Filter algorithms), which makes them widely used in adaptive control [6]. In

particular, the latter is particularly appealing in practical cases in which the dynamics of the systems changes during operations, such as in the case of changes of mass and inertia in rockets due to fuel consumption, wear of heating equipment in chemical processes, and many others. In fact, an observable SS model can be equivalently transformed into an ARX model, and the next Chapter 5 shows the equivalence of SS-based MPC and ARX-based MPC problems. In Chapter 5, we proposed an alternative for the acquisition of ARX model based on the first-principle-based modeling paradigm, rather than the data-driven identification paradigm. This allows us to acquire ARX models using many existing first-principle-based models in different engineering fields. The resulting interpretative ARX model can be adopted in an adaptive MPC framework by adding an online updating scheme for the ARX model, see Chapter 6. A common practice in MPC is to first formulate a quadratic programming (QP) problem in terms of a control-oriented prediction model and MPC parameters, and then pass it to the optimization solver. Such a problem construction step can be performed offline when the prediction model is fixed. Otherwise, it requires to be repeated online when the prediction model or MPC parameters are varying. In such varying cases, the online computation time includes both constructing and solving the QP problem associated with MPC. Indeed, often constructing and solving the MPC problem have comparable costs, such as when warm-starting strategies are employed, and set-points change slowly. The online construction of the MPC problem becomes necessary in the adaptive ARX-based MPC framework, and the case of linear parameter varying ARX (LPV-ARX) models, in which model parameters depend on a measured time-varying signal, the so-called scheduling variable. Thus, a construction-free ARX-MPC algorithm, in that MPC-to-QP construction is explicitly eliminated, would significantly save the computational loads in those cases.

4.1.1 Related works and Contribution

Some ARX-based MPC algorithms in the literature first convert the ARX model into SS form, treat the problem as a standard SS-based MPC problem [43], and then construct and solve a condensed or sparse quadratic programming (QP) problem. In fact, the ARX-to-SS transformation is not necessary in condensed and sparse MPC-to-QP constructions, which only depend on whether to eliminate or keep the ARX output variables. Choosing the condensed or sparse construction only depends on the total online computation cost (constructing and solving) [53], in time-varying ARX-based MPC problems. The OSQP solver, based on the alternating direction method of multipliers (ADMM), can directly consume the ARX model as equality constraints, which is the sparse QP formulation by keeping the output variables of the ARX model. However, it still employs an explicit MPC-to-QP construction to formulate the equality constraint matrix, and more importantly, the OSQP solver needs to repeatedly factorize and cache the Hessian matrix of the quadratic objective at each sampling time, in time-varying ARX-based MPC problems. In [99], the dynamic equality constraint from the ARX model was relaxed by using a large penalty parameter, and it resulted in an ill-conditioning bounded variable least-squares (BVLS) problem, although the active-set based method was used to mitigate the numerical difficulties to some extent. Besides computation efficiency, easy-to-deployment of an ARX-MPC algorithm should also be considered, in which the code simplicity and library-dependency are important. In this respect, compared to the active-set or interior-point based methods, the first-order method, such as the primal or dual fast gradient method, the ADMM method, is simpler but also becomes complicated in time-varying cases, in that some offline operations have to be performed online [52].

This chapter presents a simple and efficient algorithm for solving ARX-based MPC problems. Based on the coordinate-descent augmented Lagrangian method, the resulting CDAL-ARX algorithm enjoys three main features: (i) it is *construction-free*, in that it avoids the online MPC-to-QP construction in time-varying ARX cases to save computation cost;

(*ii*) is *matrix-free*, in that it avoids multiplications and factorizations of matrices, which are required by other first-order methods in time-varying ARX cases; and (*iii*) is *library-free*, as our 150-lines of C-code implementation is without any library dependency, which matters in embedded deployment.

4.2 ARX-based MPC problem formulation

Consider the multi-input multi-output (MIMO) ARX model described by

$$y_t = \sum_{i=1}^{n_a} A(i) y_{t-i} + \sum_{i=1}^{n_b} B(i) u_{t-i} \tag{4.1}$$

where $y_t \in \mathbb{R}^{n_y}$ and $u_t \in \mathbb{R}^{n_u}$ are the output and input of the system, respectively, $A(i) \in \mathbb{R}^{n_y \times n_y}$, $i = 1, \ldots, n_a$, and $B(i) \in \mathbb{R}^{n_y \times n_u}$, $i = 1, \ldots, n_b$, and n_a, n_b define the model order of the ARX model.

This chapter considers the following MPC tracking formulation based on the ARX model (4.1)

$$\begin{aligned}
\min_{Y,U,\Delta U} \quad & \frac{1}{2} \sum_{t=1}^{T} \|(y_t - r_t)\|^2_{W^y} + \|\Delta u_{t-1}\|^2_{W^{\Delta u}} \\
\text{s.t.} \quad & y_t = \sum_{i=1}^{n_a} A(i) y_{t-i} + \sum_{i=1}^{n_b} B(i) u_{t-i},\ t = 1, \ldots, T \\
& \Delta u_t = u_t - u_{t-1},\ t = 0, \ldots, T-1 \\
& y_{\min} \le y_t \le y_{\max},\ t = 1, \ldots, T \\
& u_{\min} \le u_t \le u_{\max},\ t = 0, \ldots, T-1 \\
& \Delta u_{\min} \le \Delta u_t \le \Delta u_{\max}, t = 0, \ldots, T-1
\end{aligned} \tag{4.2}$$

where T is the prediction horizon, $W^y \succ 0$ and $W^{\Delta u} \succ 0$ are diagonal weights matrices on the outputs and the input increments, respectively, r_t, $t = 1, \ldots, T$ are the future desired set-point vectors, Δu_{t-1} are the input increments, $[y_{\min}, y_{\max}]$, $[u_{\min}, u_{\max}]$, and $[\Delta u_{\min}, \Delta u_{\max}]$ define box constraints on outputs, inputs, and input increments, respectively, and $Y = (y_1, \ldots, y_T)$, $U = (u_0, \ldots, u_{T-1})$, and $\Delta U = (\Delta u_0, \ldots, \Delta u_{T-1})$ are the optimization variables.

4.3 Coordinate Descent Augmented Lagrangian method

In Chapter 2, we proposed the CDAL algorithm for SS-based MPC problems. We want to adapt here the method to solve problem (4.2) without computing a state-space realization of the ARX model (4.1) while retaining the construction-free, matrix-free, and library-free properties of CDAL.

4.3.1 Augmented Lagrangian method

The following assumptions are needed to ensure the convergence of the Augmented Lagrangian method.

Assumption 1 *Problem (4.2) has a feasible solution.*

Note that Assumption (1) is satisfied in all practical situations in which the reference r_t is far enough from the output bounds and the prediction horizon T is long enough.

Assumption 2 *The equality constraint matrix arising from stacking all the equality constraints in* (4.1) *is full rank at the optimal solution of the problem.*

Let $\mathcal{Y}$, $\mathcal{U}$, and $\Delta\mathcal{U}$ denote the hyper-boxes on Y, U, and ΔU, respectively, defined by the box constraints in (4.2), respectively. The bound-constrained Augmented Lagrangian function $\mathcal{L}_\rho : \mathcal{Y} \times \mathcal{U} \times \Delta\mathcal{U} \times \mathbb{R}^{Tn_y} \times \mathbb{R}^{Tn_u} \to \mathbb{R}$ is given by

$$
\begin{aligned}
\mathcal{L}_\rho(Y, U, \Delta U, \Lambda, \Gamma) &= \frac{1}{2}\sum_{t=1}^{T} \|(y_t - r_t)\|^2_{Wy} + \|\Delta u_{t-1}\|^2_{W\Delta u} \\
&+ \sum_{t=1}^{T} \lambda_t' \left(\sum_{i=1}^{n_a} A(i) y_{t-i} + \sum_{i=1}^{n_b} B(i) u_{t-i} - y_t \right) \\
&+ \frac{\rho}{2} \sum_{t=1}^{T} \left\| \sum_{i=1}^{n_a} A(i) y_{t-i} + \sum_{i=1}^{n_b} B(i) u_{t-i} - y_t \right\|_2^2 \\
&+ \sum_{t=1}^{T} \gamma_t' \left(u_{t-2} + \Delta u_{t-1} - u_{t-1} \right) + \frac{\rho}{2} \sum_{t=1}^{T} \| u_{t-2} + \Delta u_{t-1} - u_{t-1} \| \, \|_2^2
\end{aligned}
\tag{4.3}
$$

where $\Lambda = \{\lambda_t\}$ and $\Gamma = \{\gamma_t\}, \forall t = 1, \ldots, T$ are the dual vectors associated with the equality constraints induced by the ARX model and the input increments, respectively, and ρ is the penalty parameter. According to [15], the scaled AL method (ALM) iterates the following updates

$$(Y^k, U^k, \Delta U^k) = \operatorname{argmin} \frac{1}{\rho} \mathcal{L}_\rho(Y, U, \Delta U, \Lambda^{k-1}, \Gamma^{k-1}) \tag{4.4a}$$

$$\lambda_t^k = \lambda_t^{k-1} + \sum_{i=1}^{n_a} A(i) y_{t-i}^k + \sum_{j=1}^{n_b} B(i) u_{t-i}^k - y_t^k \\ , \forall t = 1, \ldots, T \tag{4.4b}$$

$$\gamma_t^k = \gamma_t^{k-1} + u_{t-1}^k + \Delta u_t^k - u_t^k, \forall t = 1, \ldots, T \tag{4.4c}$$

The minimization step (4.4a) updates the primal vector, Steps (4.4b) and (4.4c) update the dual vectors. We refer the reader to [15] for a well-known convergence proof of ALM under Assumptions 1, 2. To improve the speed of convergence of ALM, [47] proposed an accelerated version of ALM whose convergence rate is $O(1/k^2)$ for linearly constrained convex programs by using Nesterov's acceleration technique [71]. The accelerated ALM algorithm for MPC problems has been summarized in our previous Chapter 2.

4.3.2 Coordinate-descent method

Sub-problem (4.4a) is a strongly convex box-constrained QP problem, which can be solved by many methods. Among others, as we showed in Chapter 2, problem (4.4a) can be solved by a simple coordinate-descent method, which minimizes the objective function along only one coordinate direction at each iteration while keeping the other coordinates fixed [110]. A convergence proof of at-least linear convergence when solving convex differentiable minimization problems was shown in [65]. Under Assumption 1 (non-emptiness of the feasible set) and since the objective function $\mathcal{L}_\rho(\cdot)$ is continuously differentiable and convex with respect to each coordinate, the CD method proceeds repeatedly for $k =$

Algorithm 6 Accelerated augmented Lagrangian method [47]

Input: Initial guess $Y^0 \in \mathcal{Y}$, $U^0 \in \mathcal{U}$, $\Delta U^0 \in \mathcal{U}$, Λ^0 and Γ^0; maximum number N_{out} of iterations; parameter $\rho > 0$, $\epsilon > 0$.

1. Set $\alpha_1 = 1$; $\hat{\Lambda}^0 = \Lambda^0$; $\hat{\Gamma}^0 = \Gamma^0$;
2. **for** $k = 1, 2, \cdots, N_{\text{out}}$ **do**
   2.1. $(Y^k, U^k, \Delta U^k) = \operatorname{argmin} \frac{1}{\rho}\mathcal{L}_\rho(Y, U, \Delta U, \hat{\Lambda}^{k-1}, \hat{\Gamma}^{k-1})$
   2.2. **for** $t = 1, 2, \cdots, T$ **do**
      2.2.1. $\lambda_t^k = \hat{\lambda}_t^{k-1} + \sum_{i=1}^{n_a} A(i) y_{t-i}^k + \sum_{j=1}^{n_b} B(i) u_{t-i}^k - y_t^k$
      2.2.2. $\gamma_t^k = \hat{\gamma}_t^{k-1} + (u_{t-2}^k + \Delta u_{t-1}^k - u_{t-1}^k)$;
   2.3. **if** $\|\Lambda^k - \hat{\Lambda}^{k-1}\|_2^2 + \|\Gamma^k - \hat{\Gamma}^{k-1}\|_2^2 \le \epsilon$, **stop**;
   2.4. $\alpha_{k+1} = \frac{1+\sqrt{1+4\alpha_k^2}}{2}$;
   2.5. $\hat{\Lambda}^k = \Lambda^k + \frac{\alpha_k - 1}{\alpha_{k+1}}(\Lambda^k - \Lambda^{k-1})$;
3. **end**.

Output: $Y^{N_{\text{out}}}$, $U^{N_{\text{out}}}$, $\Delta U^{N_{\text{out}}}$.

$1, 2, \ldots,$ as follows:

$$\text{choose } j_k \in \{1, 2, \ldots, n_z\} \tag{4.5a}$$

$$z_{j_k}^k = \operatorname*{argmin}_{z_{j_k} \in \mathcal{Z}} \frac{1}{\rho}\mathcal{L}_\rho(z_{j_k}, z_{\neq j_k}^{k-1}, \Lambda^{k-1}, \Gamma^{k-1}) \tag{4.5b}$$

where

$$z = \begin{bmatrix} y_1' & u_0' & \Delta u_0' & \ldots & y_T' & u_{T-1}' & \Delta u_{T-1}' \end{bmatrix}'$$

is the optimization vector, $z \in \mathcal{Z} \triangleq \mathcal{Y} \times \mathcal{U} \times \Delta\mathcal{U}$, $\mathcal{Z} \subseteq \mathbb{R}^{n_z}$, $n_z \triangleq T(n_y + n_u + n_u)$. $\mathcal{L}_\rho(z_{j_k}, z_{\neq j_k}^{k-1}, \Lambda^{k-1}, \Gamma^{k-1})$ denotes by the value $\mathcal{L}_\rho(z, \Lambda^{k-1}, \Gamma^{k-1})$ when $z_{\neq j_k} = z_{\neq j_k}^k$ is fixed. Here $z_{\neq j_k}$ denotes the subvector obtained from z by eliminating its j_kth component z_{j_k}. The convergence of the iterations (4.5) depends on the coordinate picking rule, namely how j_k is

chosen. Existing research works have analyzed the influence of different coordinate selection rules such as the cyclic rule [38], Gauss-Southwell rule [76], the greedy rule [75], and the random selection rule [72], on the convergence rate of the coordinate descent method. We choose the simplest variant using cyclic coordinate search to favor implementation simplicity. In fact, the cyclic implementation preserves the order of optimization variables with respect to the prediction horizon t, type (output y, input u, or input increment Δu), and component. The implementation of one pass through all n_z coordinates using cyclic CD is reported in Procedure 7. In Procedure 7, the operator $\{s, \sigma\} = \mathrm{CCD}\{M, d, \sigma\}_{\underline{s}}^{\bar{s}}$ represents one pass iteration of the reverse cyclic CD method through all its n_s coordinates $s_1, \ldots, s_{n_s}$ for the following box-constrained QP $\min_{s \in [\underline{s}, \bar{s}]} \frac{1}{2} s'Ms + s'd$, that is to execute the following n_s iterations

$$
\begin{aligned}
&\text{for } i = 1, \ldots, n_s \\
&\quad \hat{s}_i \leftarrow \max(\underline{s}_i, \min(\bar{s}_i, s_i - \tfrac{1}{M_{i,i}}(M_{i,\cdot}s + d_i)) \\
&\quad \sigma \leftarrow \sigma + (\hat{s}_i - s_i)^2 \\
&\quad s_i \leftarrow \hat{s}_i \\
&\text{end}
\end{aligned}
\tag{4.6}
$$

The quantities e_t, f_t, g_t used in Procedure 7 are defined for $t = 1, \cdots, T$ as follows:

$$
\begin{aligned}
e_t &= -W^y r - (\lambda_t + \sum_{i=1}^{n_a} A(i) y_{t-i} + \sum_{i}^{n_b} B(i) u_{t-i}) \\
&\quad + \sum_{n_i=1}^{\min(n_a, T-t)} A(n_i)' (\lambda_{t+n_i} + \sum_{i \neq n_i}^{n_a} A(i) y_{t+n_i-i} + \sum_{i=1}^{n_b} B(i) u_{t+n_i-i} - y_{t+n_i}) \\
f_t &= -(\gamma_t + u_{t-2} + \Delta u_{t-1}) + (\gamma_{t+1} + \Delta u_t + u_t) \\
&\quad + \sum_{n_i=1}^{\min(n_b, T-t+1)} B(n_i)' (\lambda_{t+n_i} + \sum_{i=1}^{n_a} A(i) y_{t+n_i-i} + \sum_{i \neq n_i}^{n_b} B(i) u_{t+n_i-i} - y_{t+n_i}) \\
g_t &= \gamma_t + u_{t-2} - u_{t-1}
\end{aligned}
$$

Procedure 7 Full pass of cyclic coordinate descent on all block variables

Input: $\Lambda = \{\lambda_1, \ldots, \lambda_T\}$, $\Gamma = \{\gamma_1, \ldots, \gamma_T\}$, $Y = \{y_1, \cdots, y_T\}$, $U = \{u_0, \cdots, u_{T-1}\}$, $\Delta U = \{\Delta u_0, \cdots, \Delta u_{T-1}\}$; MPC settings $A(1), \ldots, A(n_a)$, $B(1), \ldots, B(n_b)$, W^y, $W^{\Delta u}$, $y_{\min}$, $y_{\max}$, $u_{\min}$, $u_{\max}$, $\Delta u_{\min}$, $\Delta u_{\max}$; parameter $\sigma, \rho > 0$.

1. $\sigma \leftarrow 0$;
2. **for** $t = 1, \ldots, T-1$ **do**
   - 2.1. $j = \min(n_a, T-t)$;
   - 2.2. $\{y_t, \sigma\} \leftarrow \mathrm{CCD}\{\frac{1}{\rho}W^y + I + \sum_{i=1}^{j} A(i)'A(i), e_t, \sigma\}_{y_{\min}}^{y_{\max}}$;
   - 2.3. $j = \min(n_b, T-t+1)$;
   - 2.4. $\{u_{t-1}, \sigma\} \leftarrow \mathrm{CCD}\{2I + \sum_{i=1}^{j} B(i)'B(i), f_t, \sigma\}_{u_{\min}}^{u_{\max}}$;
   - 2.5. $\{\Delta u_t, \sigma\} \leftarrow \mathrm{CCD}\{\frac{1}{\rho}W^{\Delta u} + I, g_t, \sigma\}_{\Delta u_{\min}}^{\Delta u_{\max}}$;
3. $\{y_T, \sigma\} \leftarrow \mathrm{CCD}\{\frac{1}{\rho}W^y + I, e_T, \sigma\}_{y_{\min}}^{y_{\max}}$;
4. $\{u_{T-1}, \sigma\} \leftarrow \mathrm{CCD}\{I + B(1)'B(1), f_T, \sigma\}_{u_{\min}}^{u_{\max}}$;
5. $\{\Delta u_{T-1}, \sigma\} \leftarrow \mathrm{CCD}\{\frac{1}{\rho}W^{\Delta u} + I, g_T, \sigma\}_{\Delta u_{\min}}^{\Delta u_{\max}}$;
6. **end**.

Output: Y, U, ΔU, Λ, Γ, σ.

$$
\begin{aligned}
e_T &= -W^y r - (\lambda_T + \sum_{i=1}^{n_a} A(i) y_{T-i} + \sum_{i=1}^{n_b} B(i) u_{T-i}) \\
f_T &= -(\gamma_T + u_{T-2} \\
&\quad + \Delta u_{T-1}) + B(1)'(\lambda_T + \sum_{i=1}^{n_a} A(i) y_{T-i} + \sum_{i\neq 1}^{n_b} B(i) u_{T-i} - y_T) \\
g_T &= \gamma_T + u_{T-2} - u_{T-1}
\end{aligned}
$$

which shows that they involve several matrix-vector multiplications. It would greatly affect the computation efficiency since their computational cost is proportional to the product of the inner iterations and the outer

iterations. To eliminate their explicit calculation, we propose here below an efficient coupling scheme between CD and AL that reduces the cost per iteration, without changing the rate of convergence of the algorithm.

4.3.3 Efficient coupling scheme between CD and AL

Our proposed efficient coupling scheme exploits the fact that CD only updates one coordinate each time, and the execution (4.6) of the operator $\mathrm{CCD}(\cdot)$ involves the next update of dual Lagrangian vectors. Here we take Step 2.2 of Procedure 7 as an example, which has been modified from equation (4.6) to Procedure 8. Note that the dual Lagrangian vectors used in Procedure 8 have been updated before Procedure 8. The symbols $\{D_0^y, D_1^y, \ldots, D_{T-1}^y\}$ denote the diagonal elements of their Hessian matrices used in Step 2.2 and 3

$$\begin{aligned}
&\text{for } t = 1, \ldots, T-1 \\
&\quad j = \min(n_a, T-t); \\
&\quad D_t^y \leftarrow \mathrm{diag}\left(\tfrac{1}{\rho} W^y + I + \textstyle\sum_{i=1}^{j} A(i)'A(i)\right) \\
&\text{end} \\
&D_{T-1}^y \leftarrow \tfrac{1}{\rho} W^y + I
\end{aligned} \tag{4.7}$$

To avoid repeating division operations, the values $\{\frac{1}{D_0^y}, \frac{1}{D_1^y}, \ldots, \frac{1}{D_{T-1}^y}\}$ are cached before the iterations start. The other steps involving the operator $\mathrm{CCD}(\cdot)$ in Procedure 7 follow the same idea.

4.3.4 Algorithm

Summarizing all the ingredients described in the previous sections, we obtain the construction-free ARX-based MPC Algorithm 9, which we call CDAL-ARX. Here, construction-free means that CDAL-ARX directly uses the ARX model coefficients without the need to construct a QP problem explicitly. Note that the main update of the Lagrangian variables in Algorithm 9 is placed early in Step 2.1, which is different from the original version of Algorithm 1 in Chapter 2 because the CD method allows the use of our proposed efficient coupling scheme. The quantities N_{out} and N_{in} denote the maximum number of AL (outer-loop) and CD

Procedure 8 One pass of cyclic coordinate descent for Step 2.2 of Procedure 7 after using efficient coupling scheme

Input: $j = \min(n_a, T-t)$; y_{t+1}, λ_t, $\lambda_{t+1}, \ldots, \lambda_{t+j}$; parameter $\rho > 0$; update amount $\sigma \geq 0$.

1. **for** $i = 1, \ldots, n_y$ **do**
   1.1. $s \leftarrow -\lambda_{t,i} + \sum_{n_i=1}^{j} A(n_i)'_{:,i} \lambda_{t+n_i}$;
   1.2. $\theta \leftarrow \left[y_{t,i} - \frac{\frac{1}{\rho} W_i^y (y_{t,i} - r_i) + s}{D_{t,i}^y} \right]_{y_{\min,i}}^{y_{\max,i}}$;
   1.3. $\Delta \leftarrow \theta - y_{t,i}$;
   1.4. $\sigma \leftarrow \sigma + \Delta^2$;
   1.5. $y_{t,i} \leftarrow \theta$;
   1.6. $\lambda_{t,i} \leftarrow \lambda_{t,i} + \Delta$;
   1.7. **for** $n_i = 1, \ldots, j$ **do**
      1.7.1. $\lambda_{t+n_i} \leftarrow \lambda_{t+n_i} + \Delta \cdot A(n_i)_{:,i}$
2. **end**.

Output: y_t, λ_t, $\lambda_{t+1}, \ldots, \lambda_{t+j}$, σ.

(inner-loop) iterations, respectively. The tolerances ϵ_{out} and ϵ_{in} define the stopping criteria of the outer and inner iterations, respectively.

4.4 Numerical example

In this section, we test our proposed ARX-based MPC algorithm against other MPC solvers, which rely on condensed or sparse MPC-to-QP construction, respectively. The best choice between condensed and sparse QP forms mainly depends on the number of outputs n_y, control inputs n_u, and the length of the prediction horizon T [53]. For numerical comparisons with our ARX-based MPC algorithm, this chapter considers both condensed and sparse MPC-to-QP constructions, which are then solved by the qpOASES [29] and OSQP [104], respectively. The reported

Algorithm 9 Accelerated cyclic CDAL algorithm for ARX-based MPC

Input: primal/dual warm-start $Y = \{y_1, y_2, \cdots, y_T\}$, $U = \{u_0, u_1, \cdots, u_{T-1}\}$, $\Delta U = \{\Delta u_0, \Delta u_1, \cdots, \Delta u_{T-1}\}$, $\Lambda^{-1} = \Lambda^0 = \{\lambda_1, \lambda_2, \cdots, \lambda_T\}$, $\Gamma^{-1} = \Gamma^0 = \{\gamma_1, \gamma_2, \cdots, \gamma_T\}$; History input and output data $\{y_0, y_{-1}, \ldots, y_{1-n_a}\}$, $\{u_{-1}, u_{-2}, \ldots, u_{1-n_b}\}$; MPC settings $\{A(1), A(2), \ldots, A(n_a)$, $B(1), B(2), \ldots, B(n_b)$, W^y, $W^{\Delta u}$, $y_{\min}$, $y_{\max}$, $u_{\min}$, $u_{\max}$, $\Delta u_{\min}$, $\Delta u_{\max}\}$; Algorithm settings $\{\rho, N_{\text{out}}, N_{\text{in}}\, \epsilon_{\text{out}}, \epsilon_{\text{in}}\}$

1. $\alpha_1 \leftarrow 1$; $\hat{\Lambda}^0 \leftarrow \Lambda^0$; $\hat{\Gamma}^0 \leftarrow \Gamma^0$;
2. **for** $k = 1, 2, \cdots, N_{\text{out}}$ **do**
   - 2.1. **for** $t = 1, 2, \cdots, T$ **do**
     - 2.1.1. $\lambda_t^k = \hat{\lambda}_t^{k-1} + (\sum_{i=1}^{n_a} A(i) y_{t-i}^k + \sum_{j=1}^{n_b} B(i) u_{t-i}^k - y_t^k)$
     - 2.1.2. $\gamma_t^k = \hat{\gamma}_t^{k-1} + (u_{t-2}^k + \Delta u_{t-1}^k - u_{t-1}^k)$;
   - 2.2. **for** $k_{in} = 1, 2, \cdots, N_{\text{in}}$ **do**
     - 2.2.1. $(Y, U, \Delta U, \sigma) \leftarrow$ Procedure 7 with use of Procedure 8;
     - 2.2.2. **if** $\sigma \leq \epsilon_{\text{in}}$ **break** the loop;
   - 2.3. **if** $\|\Lambda^k - \hat{\Lambda}^{k-1}\|_2^2 + \|\Gamma^k - \hat{\Gamma}^{k-1}\|_2^2 \leq \epsilon_{\text{out}}$ **stop**;
   - 2.4. $\alpha_{k+1} \leftarrow \frac{1+\sqrt{1+4\alpha_k^2}}{2}$;
   - 2.5. $\hat{\Lambda}^k \leftarrow \Lambda^k + \frac{\alpha_k - 1}{\alpha_{k+1}}(\Lambda^k - \Lambda^{k-1})$;
3. **end**.

Output: $Y, U, \Delta U, \Lambda, \Gamma$

comparison simulation results were obtained on a MacBook Pro with a 2.7 GHz 4-core Intel Core i7 and 16GB RAM. Algorithm 9, qpOASES v3.2 and OSQP v0.6.2 are all executed in MATLAB R2020a via their C-mex implementations.

4.4.1 Problem Descriptions

1. *Time-changing ARX model example:* one notable feature of ARX model is its online adaptability, making it particularly appealing in practical cases in which the system dynamic changes operations. In such time-changing ARX model examples, our CDAL-ARX algorithm can take advantage of its construction-free feature to avoid the computation cost of the online construction step in comparison to other algorithms. We tested CDAL-ARX on randomly-generated two-input-two-output ARX models with order $n_a = 4$ and $n_b = 4$ and time-varying system matrices. For demonstration purposes, here below we report one instance of them, whose ARX model matrix parameters are listed as follows,

$$\begin{aligned} A(i)^t &= A(i) + 0.1M^t, i = 1, \dots 4 \\ B(i)^t &= B(i) + 0.1M^t, i = 1, \dots 4 \end{aligned} \tag{4.8}$$

where $A(1) = \begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix}, A(2) = \begin{bmatrix} 0.7 & 0.1 \\ 0.1 & 0.7 \end{bmatrix}, A(3) = \begin{bmatrix} 0.5 & 0.1 \\ 0.1 & 0.5 \end{bmatrix},$
$A(4) = \begin{bmatrix} 0.3 & 0.1 \\ 0.1 & 0.3 \end{bmatrix}, B(1) = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}, B(2) = \begin{bmatrix} 0.8 & 0.4 \\ 0.4 & 0.8 \end{bmatrix},$
$B(3) = \begin{bmatrix} 0.6 & 0.3 \\ 0.3 & 0.6 \end{bmatrix}, B(4) = \begin{bmatrix} 0.4 & 0.2 \\ 0.2 & 0.4 \end{bmatrix}, M^t = \begin{bmatrix} \sin(\frac{t}{10}) & \cos(\frac{t}{10}) \\ \cos(\frac{t}{10}) & \sin(\frac{t}{10}) \end{bmatrix}.$

2. DNN-based LPV-ARX model example: one particular type of LPV input-output representation is LPV-ARX model, and its scheduling variable can be piecewise affine (PWA) maps. Modern deep neural network theory tells that deep ReLU networks can represent exponentially many more PWA regions than shallow one swith a fixed amount of memory [101]. We tested on CDAL-ARX on randomly-generated two-input-two-output quasi-LPV-ARX models of larger order $n_a = 6$ and $n_b = 6$, whose coefficient matrices are piecewise

affine (PWA) maps of the scheduling vector w_{t-1}

$$\begin{bmatrix} y_t(1) \\ y_t(2) \end{bmatrix} = \begin{bmatrix} \mathcal{N}_1(w_{t-1})' \\ \mathcal{N}_2(w_{t-1})' \end{bmatrix} x_{t-1} \tag{4.9}$$

where $x_{t-1} = \left[y'_{t-1}, \ldots, y'_{t-6}, u'_{t-1}, \ldots, u'_{t-6}\right]' \in \mathbb{R}^{24}$, $w_{t-1} = [y'_{t-1}, \ldots, y'_{t-6}, u'_{t-2}, \ldots, u'_{t-6}]' \in \mathbb{R}^{22}$, and $\mathcal{N}_1, \mathcal{N}_2 \in \mathbb{R}^{22} \to \mathbb{R}^{24}$ are deep feedforward neural networks with three layers and ReLU activation function, namely $\mathcal{N}_1(w_{t-1}) = W_{1,3} \max(0, W_{1,2} \max(0, W_{1,1} w_{t-1} + b_{1,1}) + b_{1,2}) + b_{1,3}$, $\mathcal{N}_2(w_{t-1}) = W_{2,3} \max(0, W_{2,2} \max(0, W_{2,1} w_{t-1} + b_{2,1}) + b_{2,2}) + b_{2,3}$. Here we choose the number of neurons in each hidden layer as three times the number of inputs according to [101], that is, $W_{1,1}$ and $W_{2,1} \in \mathbb{R}^{66\times 22}$, $b_{1,1}$ and $b_{2,1} \in \mathbb{R}^{66}$, $W_{1,2}$ and $W_{2,2} \in \mathbb{R}^{66\times 66}$, $b_{1,2}$ and $b_{2,2} \in \mathbb{R}^{66}$, $W_{1,3}$ and $W_{2,3} \in \mathbb{R}^{24\times 66}$, $b_{1,3}$ and $b_{2,3} \in \mathbb{R}^{24}$. For demonstration purposes, we define $b_{1,3}, b_{2,3}$ by collecting the coefficients defining $A(1), \ldots, A(4), A(4), A(4), B(1), \ldots, B(4)$, $B(4), B(4)$ as in (4.8); the remaining network parameters are randomly generated uniformly between 0 and 0.1. At each time t, the linear model consumed by our CDAL-ARX algorithm is given by evaluating the deep ReLU networks as in (4.9).

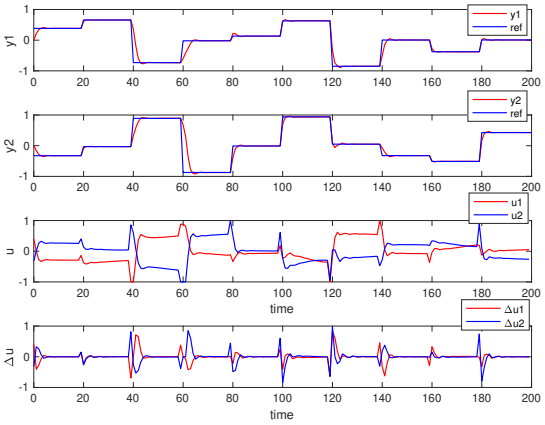
In both examples, we use the same MPC parameters $W_y = I$, $W_{\Delta u} = 0.1I$, $[y_{\min}, y_{\max}] = [-1, 1]$, $[u_{\min}, u_{\max}] = [-1, 1]$, $[\Delta u_{\min}, \Delta u_{\max}] = [-1, 1]$. Different prediction horizon lengths T are used to investigate numerical performance, namely $T = 10$, 20, and 30. Their history input-output conditions are both zeros. For example, $y_{-3} = y_{-2} = y_{-1} = y_0 = [0\ 0]'$, and $u_{-3} = u_{-2} = u_{-1} = [0\ 0]'$ for the first case. In the two examples, the closed-loop simulation is run over 200 sampling steps, and the desired references for y_1 and y_2 are randomly changed every 20 steps. Warm-start is used in all solvers (qpOASES, OSQP, CDAL-ARX). We keep default solver settings in both qpOASES and OSQP, so that they produce solutions of similar precision, that are measured in terms of Euclidean distance (since qpOASES belongs to the class of active-set methods, in principle, it always provides a high-precision solution at termination, so its solution quality cannot be tuned as easily as in the case of ADMM). For a fair comparison, in the two examples we set $\epsilon_{in} = 10^{-6}$ and $\epsilon_{out} = 10^{-6}$

under $\rho = 1$ to define the stopping criteria of our CDAL-ARX solver, so to obtain closed-loop control sequences with similar precision. In both examples, the generated closed-loop simulation results are almost indistinguishable, as shown in Figure 4.1(a) and 4.1(b), respectively, which show good tracking performance and no violation in input and output constraints.

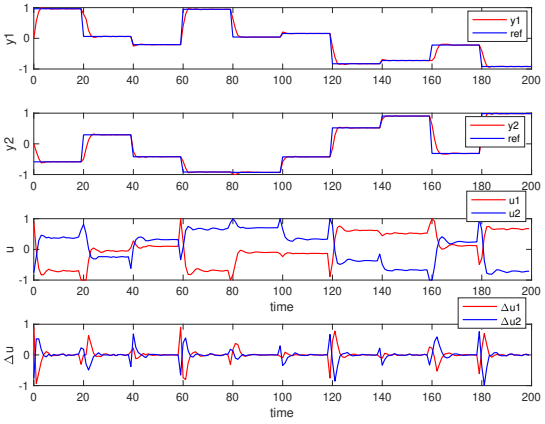
Using the qpOASES and OSQP solvers requires the online construction of the QP problem, whose computation time must be counted in the total time. Table 4.1 lists the solution time of CDAL-ARX and lists the construction and solution time when using qpOASES (condensed construction) and OSQP (sparse construction). From Table 4.1, it can be noticed that CDAL-ARX is always solving the MPC problem in a smaller CPU time when compared to the sum of construction and solution time of qpOASES and OSQP. Moreover, as the prediction horizon increases, qpOASES and OSQP may fail to solve the problem due to the ill-conditioning issue. Note also that the computation time of CDAL-ARX is often shorter than the pure solution time of qpOASES and OSQP (i.e., not counting the construction time), which seems to indicate that the reported speed-ups are due to both adopting the proposed augmented Lagrangian method and avoiding the construction step.

4.5 Conclusion

This chapter introduced a solution algorithm for solving MPC problems based on ARX models that avoid constructing the associated QP problem explicitly. Due to its matrix-free and library-free features, the proposed CDAL-ARX algorithm can be useful in adaptive embedded linear MPC applications based on ARX models, especially when combined with a fast and robust recursive linear identification method.



(a) Time-varying ARX model



(b) DNN-based LPV-ARX model

Figure 4.1: Closed-loop tracking results

Table 4.1: Computation time (ms) of CDAL-ARX and comparison with other solvers

Examples	T	CDAL-ARX avg, max	qpOASES avg, max	OSQP avg, max
Time-varying ARX	10	0.14, 1.5	0.42^{*}, 2.8^{*} $0.08^{\dagger}$, $1.4^{\dagger}$	0.41^{*}, 2.9^{*} $0.18^{\dagger}$, $0.92^{\dagger}$
	20	0.25, 2.8	1.2^{*}, 6.3^{*} $0.18^{\dagger}$, $4.4^{\dagger}$	1.0^{*}, 3.8^{*} $1.9^{\dagger}$, $17^{\dagger}$
	30	0.36, 3.6	2.6^{*}, 10.2^{*} fail	2.4^{*}, 8.1^{*} $22^{\dagger}$, $48^{\dagger}$
DNN-based LPV-ARX	10	0.51, 2.5	0.46^{*}, 3.2^{*} $0.57^{\dagger}$, $3.9^{\dagger}$	0.42^{*}, 3.6^{*} $1.1^{\dagger}$, $14^{\dagger}$
	20	1.2, 4.6	1.2^{*}, 5.5^{*} fail	0.97^{*}, 4.5^{*} $16^{\dagger}$, $32^{\dagger}$
	30	2.0, 7.3	3.1^{*}, 10.8^{*} fail	2.8^{*}, 8.9^{*} fail

*construction time, †solution time. For qpOASES and OSQP, the time to evaluate the MPC law is the sum of construction and solution time.

Chapter 5

Equivalence of SS-based MPC and ARX-based MPC

5.1 Introduction

One critical component for a successful MPC controller is how to obtain a simplified control-oriented prediction model of the physical process to predict its likely evolution. Two main kinds of control-oriented models used in MPC are the state-space (SS) and input-output models. Most earlier MPC methods employ a different types of input-output models. Input-output models were almost superseded by state-space models after three decades of MPC development [68], and the SS model provides a series of mature theories to address controllability, observability, optimal control, etc., elegantly [45, 46]. Furthermore, a SS model can be regarded as interpretative when obtained from the first-principle-based modeling paradigm. In contrast, the input-output model, such as the Auto-Regressive model with eXogenous (ARX) terms model, is a black-box model without interpretability [61].

Nonetheless, input-output models are still prevalent and preferred in industrial MPC applications. One reason is that most industrial control application scenarios cannot measure full state information, which is required by state-space-based (SS-based) MPC. In that context, an observer

(or estimator), like the well-known Luenberger observer, the Kalman Filter, and the moving horizon estimator [92], is adopted to estimate full states from streaming input-output data. The simultaneous deployment of an observer and MPC not only increases tuning difficulties and online computation costs when the dimension of states is large. Clearly, an input-output ARX model-based (ARX-based) MPC does not require cooperating with an online observer. Another reason is that the input-output ARX model is widely used in adaptive control [6], and its cheap online adaptation cost is attributed to the linear relationship between input and output. The adaptability of input-output ARX models is very appealing in practical industrial scenarios where practical plants suffer changing dynamics in their whole life operation cycle, such as in the case of fouling of heating equipment in chemical processes, changes of mass and inertia in rockets due to fuel consumption, and many others.

The input-output ARX model has long been considered black-box because its acquisition is through a data-driven paradigm such as system identification methods. Conversely, the SS model can be obtained either through the first-principle-based modeling paradigm or through the data-driven paradigm. The first-principle-based modeling paradigm typically builds nonlinear SS models with interpretability, which can derive simplified control-oriented models for MPC via linearization on operating ranges or online successive linearization. Speaking of adaptability, compared to the ARX model, the SS model is inferior in this respect. For the SS model to achieve adaptability, there are two approaches: one is when the SS model comes from the first-principle-based modeling paradigm, the joint estimation of model parameters and states can be adopted via the dual Extended Kalman Filter (EKF) [109]; the other is when the SS model comes from the data-driven paradigm, the online subspace identification algorithm, like Multivariable Output Error State Space (MOESP) [106] and Numerical algorithms for Subspace State Space System Identification (N4SID) algorithms [80], can be adopted. Compared to the online ARX identification method, both two are more computationally heavy and complicated, especially the subspace methods of the latter are based on robust numerical tools such as QR decom-

position and singular value decomposition, which are not suitable for online schemes. An online updating scheme of the MIMO ARX model can be decomposed into some multi-input-single-output models, which then can be updated parallelly by recursive-least-squares (RLS) or Kalman Filter (KF) algorithms without matrix-inverse operations in that case. It seems that interpretability and adaptability were hard to preserve simultaneously in a control-oriented model from a computational perspective.

The relationship between the SS model and the ARX model has been investigated. The ARX model can be transformed into an SS model, and vice versa. Most literature treats the ARX-based MPC problem as a standard SS-based MPC problem by converting the ARX model into the SS model, see [43]. In [4], Aoki and Havenner only revealed that the SS model could be transformed into the ARX model by using the Cayley-Hamilton theorem [31], without further discussing its application in MPC. In [86, 84], Minh et al. found that an SS model can be transformed into an equivalent ARX model in terms of an observer gain, and this enables a unifying input-output and state-space perspectives for predictive control. The control input of their proposed predictive controllers was a linear combination of past input-output data [83, 59, 85]. Their predictive control schemes did not introduce constraints capabilities like modern MPC technology, only showing that the relationship between SS and ARX model can provide an interpretation of why it is not necessary to perform an explicit state-space model identification and observer design. Few works explored the application values of the SS-to-ARX transformation.

Black-box ARX models without interpretability cannot exploit numerous existing first-principle-based models like the SS model. Simultaneously persevering the interpretability and adaptability is a desired solution for safety-critical and adaptive MPC applications. To address these issues, this chapter leverages an equivalent SS-to-ARX transformation to introduce interpretability into an adaptive ARX model. Firstly, this chapter present how an SS model can be transformed into a unique equivalent ARX model via the Cayley-Hamilton (CH) theorem [31, 4]. Note that the order of the transformed ARX model is equal to the state

dimensions of the original SS model in this CH-based SS-to-ARX transformation. However, the closed-loop performance of the transformed ARX-based MPC is sensitive to process noise and measurement noise, because the transformed ARX-based MPC controller is essentially equivalent to the SS-based MPC controller with a deadbeat observer in that case. A deadbeat observer is a minimum-time observer which is sensitive to process noise and measurement noise. To reduce the noise sensitivity existing in CH-based transformation, a generalized SS-to-ARX transformation was presented based on the Observer-Theory (OT), which can be viewed as an extension of CH-based transformation. Furthermore, this chapter derived the Kalman Filter (KF) based SS-to-ARX transformation, which shares the same ARX structure with the OT-based SS-to-ARX transformation, explaining why OT-based SS-to-ARX transformation is robust to noise.

This chapter presents an alternative to obtaining the input-output ARX model based on a first-principle-based paradigm, rather than a data-driven paradigm. By firstly linearizing existing first-principle-based (high-fidelity) models among different fields, we can obtain an (or multiple) interpretative SS model at the operating range. And then, the interpretative SS model is transformed into an (or multiple) equivalent ARX model based on the SS-to-ARX transformation theory presented in this chapter. The ARX model obtained from this paradigm can be considered to inherit the interpretability, and the ARX-based MPC problem can also be adaptive when combing an online updating scheme for the ARX model, to handle time-varying dynamics. This is the basis of the next Chapter 6, which is an interpretive and adaptive MPC framework. More importantly, this chapter reveals that choosing the order of an ARX model should depend on the process and measurement noise, to achieve a good closed-loop performance. This is totally different from the choosing rule of ARX model order in a data-driven paradigm, which is based on fitting criteria.

The structure of the chapter is as follows. Section 5.2 firstly defines the state-space based MPC tracking problem in its subsection 5.2.1, and the subsection 5.2.2, 5.2.3 and 5.2.4 present the derivations of the CH-

based, OT-based, and KF-based SS-to-ARX transformations, respectively. In Section 5.4, numerical experiments are presented. Finally, we draw conclusions in Section 5.5. $H \succ 0$ ($H \succeq 0$) denotes positive definiteness (semi-definiteness) of a square matrix H. For a vector z, $\|z\|_H^2$ denotes the operation $z'Hz$. H' (or z') denotes the transpose of matrix H (or vector z).

5.2 Problem Description and Methods

5.2.1 State-space based MPC problem

This chapter considers the following discrete-time state-space model,

$$x_{t+1} = Ax_t + Bu_t \tag{5.1a}$$
$$y_t = Cx_t \tag{5.1b}$$

where each $\{x_t\} \in \mathbb{R}^n$ are the state variables, each $\{u_t\} \in \mathbb{R}^q$ are the input variables and each $\{y_t\} \in \mathbb{R}^m$ are the output variables. Then, a state-space based tracking MPC formulation is shown as follows,

$$\begin{aligned}
\min \quad & \sum_{t=0}^{T-1} \|(y_{t+1} - r_{t+1})\|_{W_y}^2 + \|\Delta u_t\|_{W_{\Delta u}}^2 \\
\text{s.t.} \quad & x_{t+1} = Ax_t + Bu_t, t = 0, \ldots, T-1 \\
& y_t = Cx_t, t = 1, \ldots, T \\
& u_t = u_{t-1} + \Delta u_t, t = 0, \ldots, T-1 \\
& y_{\min} \le y_t \le y_{\max},\ t = 1, \ldots, T \\
& u_{\min} \le u_t \le u_{\max},\ t = 0, \ldots, T-1 \\
& \Delta u_{\min} \le \Delta u_t \le \Delta u_{\max}, t = 0, \ldots, T-1 \\
& x_0 = \hat{x}_0
\end{aligned} \tag{5.2}$$

where $\{r_t\}$ are the desired output tracking reference signals, $\{y_t\}$ are the measured outputs subject to bound constraints $[y_{\min}, y_{\max}]$, $\{u_t\}$ are the control inputs subject to bound constraints $[u_{\min}, u_{\max}]$, $\{\Delta u_t\}$ are control input increments subject to bound constraints $[\Delta u_{\min}, \Delta u_{\max}]$, $W_y \succ 0$

and $W_{\Delta u} \succ 0$ are the diagonal weights matrices. $\hat{x}_0$ is the estimated value of full state variable at current time t, which is estimated by estimation algorithm or observer. We assume that only the measured outputs, inputs, and input increments have box-constraints; the states do not.

5.2.2 Cayley-Hamilton based SS-to-ARX transformation

Based on the states evolution equation (5.1a), calculating from previous time $t-n$ time to current time t can lead to the following equation

$$\begin{aligned} x_t &= A^n x_{t-n} \\ &\quad + \begin{bmatrix} B, AB, \cdots, A^{n-1}B \end{bmatrix} \begin{bmatrix} u_{t-n} \\ u_{t-n+1} \\ \vdots \\ u_{t-1} \end{bmatrix} \end{aligned} \tag{5.3}$$

By writing down each output equation (5.1b) from previous time $t-n$ to current time t,

$$\begin{aligned} y_{t-n} &= Cx_{t-n} \\ y_{t-n+1} &= CAx_{t-n} + CBu_{t-n} \\ y_{t-n+2} &= CA^2x_{t-n} + CBu_{t-n+1} + CABu_{t-n} \\ &\vdots \\ y_t &= CA^n x_{t-n} \\ &\quad + \begin{bmatrix} CB, CAB, \cdots, CA^{n-1}B \end{bmatrix} \begin{bmatrix} u_{t-n} \\ u_{t-n+1} \\ \vdots \\ u_{t-1} \end{bmatrix} \end{aligned} \tag{5.4}$$

Here the Cayley-Hamilton theorem [31] is introduced to remove the state vector x_{t-n}. The Cayley–Hamilton theorem states that every square matrix satisfies its own characteristic equation. For a given $n \times n$ matrix A, then the characteristic polynomial of A is defined as $p_A(\lambda) = \det(\lambda I_n - A)$, the determinant is also a degree-n monic polynomial in λ, $p_A(\lambda) = \lambda^n + c_1\lambda^{n-1} + \cdots + c_{n-1}\lambda + c_n$. One can create an analogous polynomial $p_A(A)$ in the matrix A instead of the scalar variable λ, defined as

$p_A(A) = A^n + c_1A^{n-1} + \cdots + c_{n-1}A + c_nI_n$. The Cayley–Hamilton theorem states that this polynomial expression is equal to the zero matrices, which allows A^n to be expressed as a linear combination of the lower matrix powers of A.

$$A^n + c_1A^{n-1} + \cdots + c_{n-1}A + c_nI_n = \begin{pmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{pmatrix} \tag{5.5}$$

By substituting this equation (5.5) into the output equations (5.4) from previous $t-n$ time to current time t, we can obtain the following transformed ARX model

$$\begin{aligned} &y_t + c_1y_{t-1} + \cdots + c_{n-1}y_{t-n+1} + c_ny_{t-n} \\ &= \Theta_1u_{t-1} + \Theta_2u_{t-2} + \Theta_3u_{t-3} + \ldots + \Theta_nu_{t-n} \end{aligned} \tag{5.6}$$

where $\Theta_1 = CB, \Theta_2 = CAB + c_1CB, \Theta_3 = CA^2B + c_1CAB + c_2CB, \cdots$, $\Theta_n = CA^{n-1}B + c_1CA^{n-2}B + \cdots + c_{n-1}CB$.

Let us call the above SS-to-ARX transformation the CH-based transformation, which implies that the order of the ARX model is equal to the state dimension of the SS model. But the CH-based transformation suffers the noise sensitivity issues. The next subsection 5.2.3 shows that the observed-theory-based (OT-based) SS-to-ARX transformation is a generalized transformation theory, which can be robust to noise.

5.2.3 Observer-Theory based SS-to-ARX transformation

Utilizing a gain matrix L in the SS model (5.1) to begin the derivation of the SS-to-ARX transformation as follows

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t - Ly_t + Ly_t \\ &= (A - LC)x_t + Bu_t + Ly_t \end{aligned} \tag{5.7}$$

Based on the above evolution equation (5.7), calculating from previous time $t-p$ to current time t can lead to the following equation

$$\begin{aligned} x_t = (A-LC)^p x_{t-p} + \sum_{i=1}^{p} (A-LC)^{i-1} L y_{t-i} \\ + \sum_{i=1}^{p} (A-LC)^{i-1} B u_{t-i} \end{aligned} \tag{5.8}$$

If a gain matrix L exists such that $(A-LC)^p$ vanishes to zero, i.e.,

$$(A-LC)^k \equiv 0, \quad k \geq p \tag{5.9}$$

From linear system theory, the existence of such a gain matrix L is assured as long as the system is observable. Thus, the term $(A-LC)^p x_{t-p}$ in (5.8) is zero for $k \geq p$, then multiplying the matrix C on both sides of the equation (5.8) can derive the following ARX model,

$$\begin{aligned} y_t = \sum_{i=1}^{k} C(A-LC)^{i-1} L y_{t-i} \\ + \sum_{i=1}^{k} C(A-LC)^{i-1} B u_{t-i} \end{aligned} \tag{5.10}$$

Next, we provide the analysis to tell why the gain matrix L can be viewed as an observer gain. The original state-space model (5.1) has an observer gain L of the following form

$$\begin{aligned} \hat{x}_{t+1} &= A\hat{x}_t + Bu_t + L(y_t - \hat{y}_t) \\ \hat{y}_t &= C\hat{x}_t \end{aligned} \tag{5.11}$$

where $\hat{x}_t$ is the estimated state. The state estimation error can be denoted as $e_t = x_t - \hat{x}_t$, then e_t follows the dynamic equation,

$$e_{t+1} = (A-LC)e_t \tag{5.12}$$

Thus, the estimated state $\hat{x}_t$ will converge the actual value x_t as t tends to infinity if the matrix $A-LC$ is asymptotically stable, namely the condition (5.9), that is why we call (5.10) is the OT-based SS-to-ARX transformation.

In fact, the connection between CH-based and OT-based SS-to-ARX transformation can be established by a generalized Cayley-Hamilton theorem. Defining a sequence matrices $\{M_i\}$, each of them belongs to $\mathbb{R}^{n\times m}$. As long as $km \geq n$ then it is guaranteed for an observable system that $\{M_i\}$ exists such that

$$A^p + M_1 C A^{p-1} + M_2 C A^{p-2} + \ldots + M_p C = 0 \tag{5.13}$$

which is a generalized Cayley-Hamilton theorem. Following the same idea of the standard Cayley-Hamilton-based derivation in section 5.2.2, can derive a general ARX model as follows

$$y_t = \sum_{i=1}^{p} \left(\bar{A}_i y_{t-i} + \bar{B}_i u_{t-i}\right) \tag{5.14}$$

We omit this detailed derivation since the OT-based SS-to-ARX transformation is more easily computable, only required to satisfy the condition (5.9) such as using pole placement methods.

5.2.4 Kalman Filter based SS-to-ARX transformation

This section shows the derivation of the SS-to-ARX transformation in the presence of process and measurement noise. It tells that an ARX model can be equivalent to an SS model with its optimal Kalman filter.

Consider the case where the state-space model (5.1) has process and measurement noises

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + w_t \\ y_t &= Cx_t + v_t \end{aligned} \tag{5.15}$$

where the process noise w_t and measurement noise v_t are two zero mean white noise with covariances Q and R, respectively. The linear stochastic state-space (5.15) model can also be expressed in the form of a Kalman filter

$$\begin{aligned} \hat{x}_{t+1} &= A\hat{x}_t + Bu_t + K\epsilon_t \\ y_t &= C\hat{x}_t + \epsilon_t \end{aligned} \tag{5.16}$$

where ϵ_t is a white sequence of residual with covariance $\Sigma = CPC' + R$ and P is the unique positive definite symmetric solution of the algebraic Riccati equation

$$P = APA' - APC'(CPC' + R)^{-1}CPA' + Q$$

The Kalman filter gain K is given by $K = APC'(CPC'+R)^{-1} = APC'(\Sigma)^{-1}$. Then, the system (5.16) can be expressed as

$$\begin{aligned} \hat{x}_{t+1} &= A\hat{x}_t + Bu_t + K\epsilon_t - Ky_t + Ky_t \\ &= A\hat{x}_t + Bu_t + K\epsilon_t - KC\hat{x}_t - K\epsilon_t + Ky_t \\ &= (A - KC)\hat{x}_t + Bu_t + Ky_t \\ y_t &= C\hat{x}_t + \epsilon_t \end{aligned} \tag{5.17}$$

Based on the above evolution equation (5.17), calculating from previous time $t - p_1$ to current time t can lead to the following equation

$$\begin{aligned} y_t = C(A - KC)^{p_1}\hat{x}_t + \sum_{i=1}^{p_1} C(A - KC)^{i-1}Ky_{t-i} \\ + \sum_{i=1}^{p_1} C(A - KC)^{i-1}Bu_{t-i} + \epsilon_t \end{aligned} \tag{5.18}$$

Provided that p_1 is enough large such that

$$(A - KC)^k \approx 0, k \geq p_1$$

then the input-output model (5.18) can be approximated by the following ARX model

$$\begin{aligned} y_t = \sum_{i=1}^{p_1} C(A - KC)^{i-1}Ky_{t-i} \\ + \sum_{i=1}^{p_1} C(A - KC)^{i-1}Bu_{t-i} + \epsilon_t \end{aligned} \tag{5.19}$$

which has the same structure as equation (5.11) of the deterministic case, only that the value of p_1 may be larger than p. This shows that the equation (5.11) is also robust to noise, and its noise robustness increases as

the order increases. More importantly, it tells that directly online identifying an ARX model can replace the design procedure of a Kalman Filter in which the process and measurement noise covariances require to be estimated.

5.3 ARX-based MPC problem

After the above discussions, an interpretative SS model can be equivalently transformed into an adaptive ARX model. Thus, the SS-based MPC problem (5.2) has the following equivalent ARX-based MPC problem,

$$
\begin{aligned}
\min \quad & \sum_{k=0}^{T-1} \|(y_{t+1} - r_{t+1})\|^2_{W_y} + \|\Delta u_t\|^2_{W_{\Delta u}} \\
\text{s.t.} \quad & y_t = \sum_{i=1}^{p} (\overline{A}_i y_{t-i} + \overline{B}_i u_{t-i}), t = 1, \ldots, T \\
& u_t = u_{t-1} + \Delta u_t, t = 0, \ldots, T-1 \\
& y_{\min} \leq y_{t+1} \leq y_{\max}, t = 0, \ldots, T-1 \\
& u_{\min} \leq u_t \leq u_{\max}, t = 0, \ldots, T-1 \\
& \Delta u_{\min} \leq \Delta u_t \leq \Delta u_{\max}, t = 0, \ldots, T-1
\end{aligned} \tag{5.20}
$$

where the past input-output data $\{u_{-1}, \cdots, u_{1-p}\}$, $\{y_0, \cdots, y_{1-p}\}$ needs to be provided.

Traditionally, solving the ARX-based MPC problem (5.20) has to be first constructed into a condensed Quadratic Programming (QP) problem or a sparse QP problem via the construction procedure, and then apply some standard QP algorithms to obtain a solution. The previous Chapter 4 proposed a construction-free ARX-baed MPC algorithm suitable for our considered application scenarios. And it has presented the principle of the construction-free ARX-baed MPC algorithm and the numerical comparisons with other non-construction-free algorithms. Thus, this study applies our construction-free ARX-baed MPC algorithm to solve the ARX-based MPC problem (5.20) and not provides numerical comparisons with other algorithms due to limited space. Moreover, the main

purpose of this study is to present the equivalence of SS-based MPC and ARX-based MPC and investigate the robustness of different SS-to-ARX transformations to noise via closed-loop simulations.

5.4 Numerical example

This section utilizes a classical open-loop unstable AFTI-16 aircraft example to show the equivalence of SS-based MPC and ARX-based MPC and investigate the sensitivity of different SS-to-ARX transformations to the process and measurement noise. The reported comparison simulation results were obtained on a MacBook Pro with 2.7 GHz 4-core Intel Core i7 and 16GB RAM, and ARX-based MPC problems are solved by our C-mex implementation of previous Chapter 4 in MATLAB (2020a).

The two-input-two-output plant is continuous-time linearized AFTI-16 aircraft model reported in [48, 11] as follows

$$\begin{cases} \dot{x} = \begin{bmatrix} -0.0151 & -60.5651 & 0 & -32.174 \\ -0.0001 & -1.3411 & 0.9929 & 0 \\ 0.00018 & 43.2541 & -0.86939 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} x \\ \qquad + \begin{bmatrix} -2.516 & -13.136 \\ -0.1689 & -0.2514 \\ -17.251 & -1.5766 \\ 0 & 0 \end{bmatrix} u \\ y = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x \end{cases}$$

The sampling time of designing a digital controller is 0.05 s, and the corresponding discrete-time model using exact discretization is

$$\begin{cases} x_{t+1} = \begin{bmatrix} 0.0093 & -3.0083 & -0.1131 & -1.6081 \\ 4.7030 \times 10^{-6} & 0.9862 & 0.0478 & 3.8501 \times 10^{-6} \\ 3.7028 \times 10^{-6} & 2.0833 & 1.0089 & -4.3616 \times 10^{-6} \\ 1.3556 \times 10^{-7} & 0.0526 & 0.0498 & 1 \end{bmatrix} x_t \\ \qquad + \begin{bmatrix} -0.0804 & -0.6347 \\ -0.0291 & -00143 \\ -0.8679 & -0.0917 \\ -0.0216 & -0.0022 \end{bmatrix} u_t \\ y_t = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x_t \end{cases}$$

Three SS-to-ARX transformations are designed through different poles placement: the first one is based on Cayley-Hamilton theorem from the

subsection 5.2.2, named as *ARX-CH*, its ARX coefficients matrices are

$$\overline{A}_1 = \begin{bmatrix} 3.9944 & 0 \\ 0 & 3.9944 \end{bmatrix}, \overline{A}_2 = \begin{bmatrix} -5.8834 & 0 \\ 0 & -5.8834 \end{bmatrix}$$
$$\overline{A}_3 = \begin{bmatrix} 3.7837 & 0 \\ 0 & 3.7837 \end{bmatrix}, \overline{A}_4 = \begin{bmatrix} -0.8947 & 0 \\ 0 & -0.8947 \end{bmatrix}$$
$$\overline{B}_1 = \begin{bmatrix} -0.0291 & -0.0143 \\ -0.0216 & -0.0022 \end{bmatrix}, \overline{B}_2 = \begin{bmatrix} 0.0461 & 0.0386 \\ 0.0199 & 0.0012 \end{bmatrix}$$
$$\overline{B}_3 = \begin{bmatrix} -0.0049 & -0.0343 \\ 0.0213 & 0.0026 \end{bmatrix}, \overline{B}_4 = \begin{bmatrix} -0.0121 & 0.0100 \\ -0.0196 & -0.0016 \end{bmatrix}$$

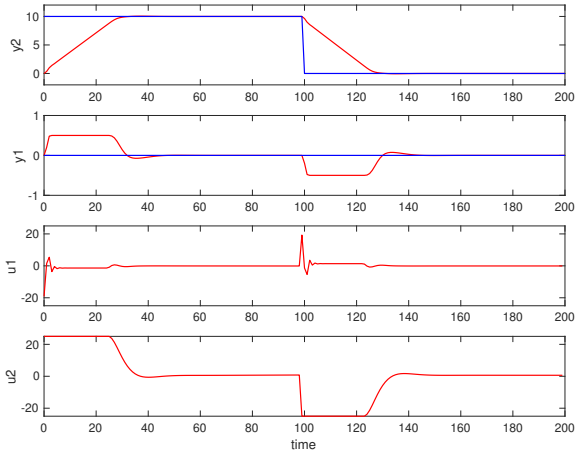
the second and third ARX transformations are based on observer-theory from the subsection 5.2.3, and their poles are placed at $[0.01, 0.02, 0.03, 0.04]$ and $[0.04, 0.08, 0.12, 0.16]$, named *ARX-OT-1* and *ARX-OT-2*, respectively. Based on their truncation errors, the order of *ARX-OT-1* and *ARX-OT-2* are 4 and 6, respectively. We simulate three scenarios with different process and measurement noises magnitude, that is, $q = r = 0, q = r = 0.01, q = r = 0.05$. The SS-based MPC is cooperated with the Kalman filter with the given known noise covariances matrices, named *SS-KF*. The input constraints are $|u_i| \leq 25^\circ, i = 1, 2$, the output constraints are $-0.5 \leq y_1 \leq 0.5$ and $-100 \leq y_2 \leq 100$. The control goal is to make the pitch angle y_2 track a reference signal r_2. In designing the MPC controller we take $W_y = \text{diag}([10,10])$, $W_u = 0$, $W_{\Delta u} = \text{diag}([0.1, 0.1])$, and the prediction horizon is $T = 10$.

The Figure 5.1(a) is the closed-loop performance results of *ARX-CH* under $q = r = 0$, namely without any noise. The *ARX-OT-1* and *ARX-OT-2* and *SS-KF* transformations all coincide with the same plot results (thus omitted), which shows that the tracking performance is good, and there are no constraint violations in input and output. It tells the equivalence of SS-based and ARX-based MPC problems under the presented SS-to-ARX transformation theory. But the *ARX-CH* transformation suffers bad closed-loop performance under noise $q = r = 0.01$ see Figure 5.1(b), and diverges under noise $q = r = 0.05$ (thus no figure). It shows that the Cayley-Hamilton transformation is very sensitive to noise. The Figure 5.2, 5.3 and 5.4 are the closed-loop control simulation results of *ARX-OT-1*, *ARX-OT-2* and *SS-KF*, under noise $q = r = 0.01$ and $q = r = 0.05$, respectively. Clearly, they all are more robust than

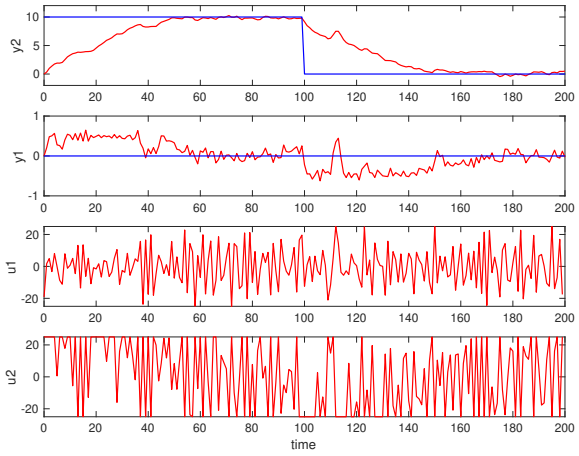
ARX-CH, and the order of better control performance is *ARX-OT-1* and *ARX-OT-2* and *SS-KF*, which is represented by the average MPC tracking cost. It also shows that the larger values of placed poles can be more robust to noise, and the closed-loop performance can approximate the optimal Kalman Filter, but more orders of the ARX model requires.

5.5 Conclusion

This chapter proposes the first-principle-based modeling paradigm for the acquisition of ARX model. This paradigm firstly obtains control-oriented SS models through linearizing the first-principle-based (high-fidelity) models, which are then transformed into equivalent ARX models via the Cayley-Hamilton, Observer-Theory, and Kalman Filter based SS-to-ARX transformations presented in this chapter. Numerical results of the AFTI-16 MPC numerical example show the equivalence of SS-based and ARX-based MPC problems and the noise robustness of different SS-to-ARX transformations, which point out that choosing the ARX model order should depend on the process and measurement noise, to achieve a good closed-loop performance, rather than depending on fitting criteria in data-driven ARX identification paradigm. This interpretative ARX model can naturally be adopted in an adaptive MPC framework by adding an online updating scheme for the ARX model; see the next Chapter 6.

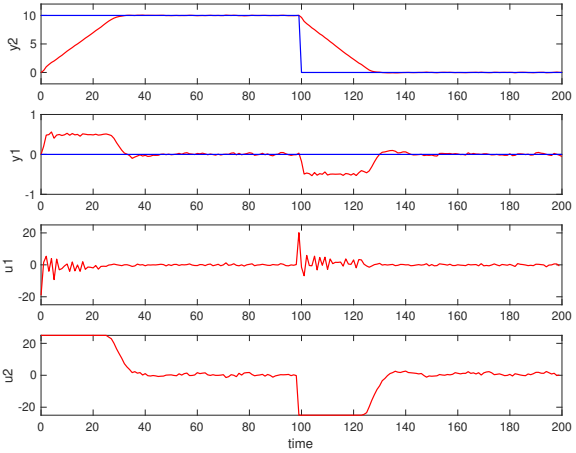


(a) $q = r = 0.0$, average MPC tracking cost $= 8511.2422$

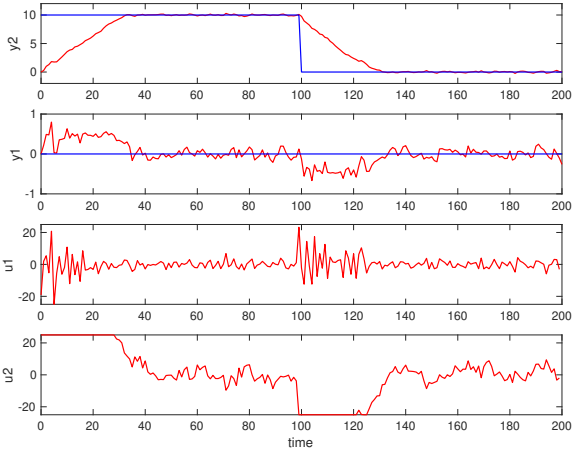


(b) $q = r = 0.01$, average MPC tracking cost $= 26002.4804$

Figure 5.1: The closed-loop control performance of *ARX-CH*

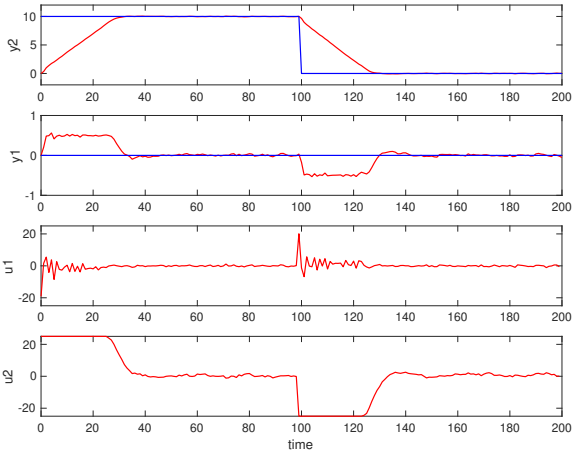


(a) $q = r = 0.01$, average MPC tracking cost $= 8628.8547$

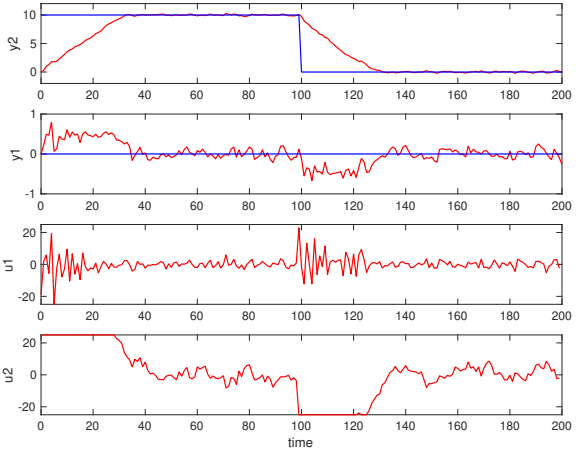


(b) $q = r = 0.05$, average MPC tracking cost $= 9772.1471$

Figure 5.2: The closed-loop control performance of *ARX-Ob1*

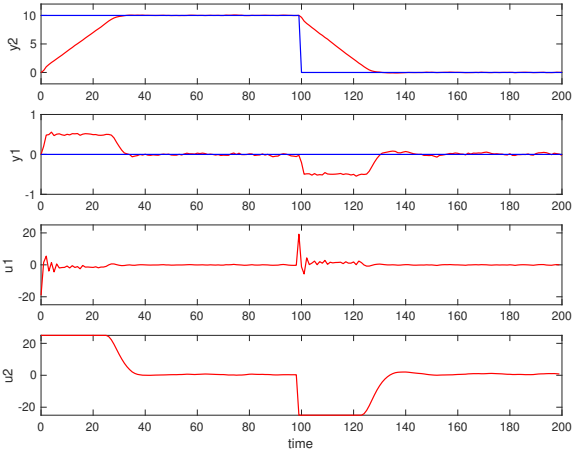


(a) $q = r = 0.01$, average MPC tracking cost $= 8604.6714$

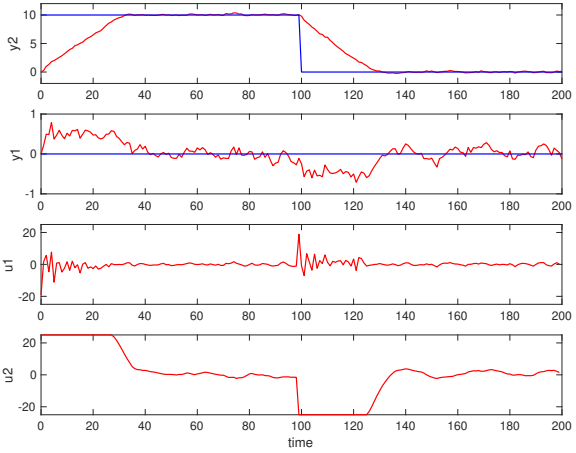


(b) $q = r = 0.05$, average MPC tracking cost $= 9583.1624$

Figure 5.3: The closed-loop control performance of *ARX-Ob2*



(a) $q = r = 0.01$, average MPC tracking cost $= 8596.0559$



(b) $q = r = 0.05$, average MPC tracking cost $= 9043.3026$

Figure 5.4: The closed-loop control performance of *SS-KF*

Chapter 6

An interpretative and adaptive MPC for nonlinear systems

6.1 Introduction

After major developments in the field of MPC over the past three decades, MPC for linear plants described by the linear state-space model has made significant progress in theoretical stability analysis and real-time numerical algorithm implementation [56]. However, most industrial plants are nonlinear, and their constrained nonlinear MPC (NMPC) formulation is a non-convex optimization problem that encounters practical difficulties in terms of computational complexity and algorithm implementation, such as solving it within sampling time on embedded platforms. It is known that a trade-off exists between the described accuracy of the nonlinear model and the online NMPC computation cost: the more accurate/complicated the model, the greater the online computational cost.

Most industrial approaches are based on linear models obtained from the linearization technique in solving MPC problems for nonlinear systems. A nonlinear plant generally admits a locally-linearized model when considering regulating a particular operating point. In most cases, a lin-

ear MPC yield adequate performance, especially for chemical process industries. This approach with one linear model has an advantage in designing and deploying the offline explicit MPC solutions based on multi-parametric quadratic programming [14] and the online MPC solutions based on fast convex optimization algorithms for embedded platforms to satisfy real-time requirements. For tracking problems that operate over a wide range of operating conditions, multiple linear model based MPC addresses the non-adequate accuracy issue in one linear model based MPC. Multiple linear models are also called linear parameter varying (LPV), often used in the context of gain-scheduling, namely an MPC controller scheduled by linear models [21]. One approach for obtaining an LPV model is the online successive linearization at the current states based on a first-principles model [54].

One improved approach is the linear time-varying (LTV) model, obtained from linearizing the nonlinear model at each prediction horizon [26]. However, the LTV-MPC approach comes at a higher computational cost than the LPV-MPC approach. Since the states and inputs during the prediction horizon are unknown, the typical way utilizes the shifted optimal inputs sequence of the previous MPC solution as the inputs, and the states are calculated by integrating the nonlinear model. Another successful and broadly used approach is the real-time iteration (RTI) scheme. The RTI approach assumes that the MPC solution at the current time is very similar to the solution obtained at the previous time. Under this assumption, a full Newton step can be taken, providing an excellent approximation of the fully converged NMPC solution. Not only that, but the RTI algorithm also divides the whole computation into preparation phase and feedback phase phases, to achieve a shorter feedback delay [35]. The RTI-NMPC approach has been implemented into the open-source software ACADO Toolkit, which allows exporting the optimized C-code for deployment [90, 41]. Another approximate scheme in the literature is the Continuation/GMRES method [78]. In [78, 79, 50], the authors also provide its full implementation within the tool AutoGenU to support the C-code generation.

In addition to the above approximating approaches, incorporating

nonlinearity directly into the MPC problem can provide a systematic way of dealing with systems with nonlinear dynamics, constraints, and objectives but at the cost of heavy online computation. Solving the resulting non-convex optimization problem relies on an efficient and reliable nonlinear programming algorithm, which has been alleviated to some extent with the advent of tailored and professional software tools, such as the CasADi [3] and FORCES NLP [118].

To further reduce the online computation cost of NMPC, the tremendous advances in machine learning and deep learning inspired many works, such as learning a globally-linear model or learning an efficient representation of approximated MPC laws via deep neural networks. Herein we only name a few works related to MPC for nonlinear systems. In [51], the author utilized a lifting operator, called the Koopman operator, to lift the nonlinear dynamics into a higher dimensional space where its evolution is approximately linear. Then an MPC controller based on the lifted linear model can replace the NMPC. The lifted linear model can be learned by a user-defined dictionary library or deep neural networks from data [66, 44]. Another approach that directly learns the approximated MPC law to reduce the online computation cost has been explored in the literature. In [49], the authors show that a neural network can represent exactly the MPC law described by the piecewise affine function in the case of linear MPC. The authors further extended to the case of nonlinear systems [62], in which the deep neural network is utilized to learn the robust nonlinear MPC law.

In the practice of the MPC technique, in addition to numerical optimization algorithms needed to solve the MPC problem, another critical ingredient is estimation algorithms. Generally, the nonlinear dynamic model can be an interpretative first-principle state-space model or a black-box model built from experimental data via system identification (or modern machine learning) in an NMPC setting. A nonlinear first-principle state-space model can provide interpretability that is preferred in practice, but it often suffers the issues like unavailable full-state information, unmeasured disturbance, or unmodeled time-varying terms. Therefore, these cases require a state estimation algorithm or a joint state

parameter estimation algorithm in which unmeasured disturbances and unmodeled time-varying terms are considered as parameters to be estimated. Many state estimation methods for nonlinear systems exist. The well-known extended Kalman Filter (EKF) is undoubtedly the predominant state estimation technique [74], and the EKF has also been applied in many joint state-parameter estimation applications [107, 30].

The identified nonlinear black-box model used in NMPC can be divided into two categories: state-space or input-output formulation, such as state-space based recurrent neural networks (RNNs) or input-output based neural-network autoregressive model with exogenous inputs. One obvious advantage of input-output based models over state-space based models is that input-output based models do not require an state estimation algorithm to estimate the generally higher-dimensional states. Despite this, input-output based models still require an online estimation algorithm to update the model parameters when discrepancies happen between the model output predictions and the streaming output measurements. The linear input-output ARX models are well known for their adaptability allowing efficient online recursive estimation algorithms, like recursive least-squares (RLS) or Kalman Filter [37]. In fact, nonlinear ARX models also have good adaptability, updating their model parameters online via the EKF algorithm [9].

As discussed above, the EKF algorithm is an essential component in NMPC practice, both for the first-principle based and black-box models. The EKF is based on the first-order linearization of nonlinear dynamics around the previously estimated vectors. Although the first-order linear approximation used by the EKF is sometimes not accurate enough, other advanced techniques can provide better estimation accuracy, but this comes at the cost of complicated implementation and expensive computational costs. The ease of implementation and computation burden are of great importance, especially in the cooperation of NMPC and EKF. Despite the widely practical usefulness of the EKF, its convergence and stability guarantee requires strict conditions, such as satisfying the nonlinear observability rank condition and having sufficiently small initial estimation error as well as disturbing noise term [93, 60]. In this chap-

ter we assume that the EKF algorithm will not suffer from divergence problems in its applications.

It is worth noting that the widely used SL-MPC approach and the EKF algorithm are both based on linearization techniques. Their combination, namely the SL-MPC with the EKF method presented in [57], has become a common choice to handle the nonlinear first-principle based model in the industrial practice of NMPC technique.In its EKF part, the states and unknown disturbances are recursively updated from the streaming input-output data based on the linearization of the discrete-time nonlinear model. In its SL-MPC part, the control input is calculated based on the estimated states and disturbances at the current time and the linearized state-space model, which is also updated online based on the linearization of a first-principle model at the estimated states and disturbances. Thus, the SL-MPC with the EKF method, in a sense, is utilizing the EKF algorithm and the streaming input-output data to continuously update online a linear state-space model, which is under the representation of states defined by the first-principle-based model. And a linear input-output plant could have infinite equivalent linear state-space realizations via coordinate transformation. The previous Chapter 5 also illustrated that a linear state-space model can be equivalently transformed into an input-output ARX model, which is well known for its adaptibility. In this chapter, the SL-MPC with EKF method is the starting point to derive our interpretative and adaptive MPC (IA-MPC) method, which is illustrated in detail in Section 6.2.

6.1.1 Contribution

In our IA-MPC method, a linear state-space model is first obtained by performing the linearization of a first-principle based model at the initial point, then transformed into an equivalent ARX model; all of these are performed offline. This offline ARX model acquisition not only makes the black-box ARX model inherit the interpretability of the first-principle based model but also exploits its adaptive nature; this is why we call our method Interpretive and Adaptive MPC (IA-MPC). In the online closed-

loop control, the EKF algorithm is used to update the ARX model parameters, and the previous construction-free CDAL-ARX algorithm in Chapter 4 is used to compute the MPC control input.

The advantage of our IA-MPC method are sumarized as follows:

1. In addition to the gained interpretability and adaptability, the acquisition of the offline ARX model also makes it possible to have a sufficiently small initial estimation error in ARX model parameters. All the EKF algorithm needs to do is the system tracking (or feedback correction), not the system identification.

2. Compared to the EKF algorithm in the SL-MPC method, the EKF algorithm in our IA-MPC method can be decoupled when updating the ARX model parameters, thanks to the decoupling feature of ARX model. The decoupled EKF computation avoids matrix inversion and only involves scalar division, even allowing parallel calculation according to the number of outputs.

3. The corresponding ARX-based MPC problem is solved by our previous construction-free, matrix-free, and library-free CDAL-ARX algorithm. Thus, our IA-MPC method would significantly reduce the difficulty in deploying nonlinear MPC on embedded platforms.

6.2 Successive Linearization NMPC with EKF

This section considers a MPC tracking problem for nonlinear systems subject to the input and output constraints, considering a first-principle nonlinear model as follows

$$\dot{x} = f(x, u, d) \tag{6.1a}$$

$$y = g(x, d) \tag{6.1b}$$

where $x \in \mathbb{R}^{n_x}$, $u \in \mathbb{R}^{n_u}$, and $y \in \mathbb{R}^{n_y}$ are the states, inputs, and outputs, respectively. And $d \in \mathbb{R}^{n_d}$ denotes the unmeasured disturbances. In industrial NMPC practice, one efficient approach for handling the

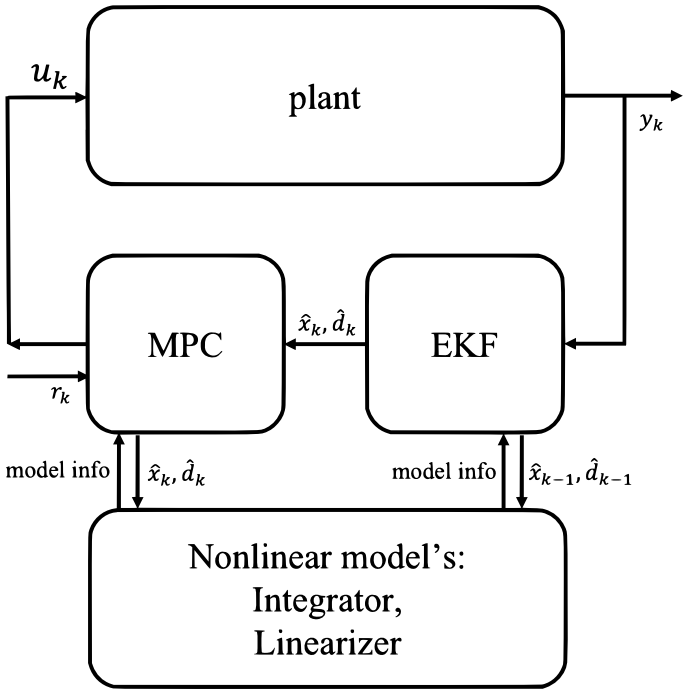


Figure 6.1: Schematic diagram of successive linearization NMPC with EKF

nonlinearity of a first-principle model is based on successive linearization, which is applied both in state estimation and MPC computation [57]. Then, an output-feedback nonlinear MPC combines a state estimator (EKF) and a state-feedback successive linearization NMPC, and its schematic diagram is shown in Figure 6.1.

6.2.1 Extended Kalman Filter

The extended Kalman Filter exploits the idea that performs the linearization at each sampling time to approximate the nonlinear system as a time-varying system and apply the linear filtering theory. In digital controller design, u and d are assumed to a constant value between the sampling time. Thus, a discrete-time model of (6.1) can be formulated as

follows:

$$x_k = F_{t_s}(x_{k-1}, u_{k-1}, d_{k-1}) \tag{6.2a}$$

$$y_k = g(x_k, d_k) \tag{6.2b}$$

where $F_{t_s}(x_{k-1}, u_{k-1}, d_{k-1})$ represents the the terminal state vector obtained by integrating the ordinary differential equation (6.1a) for one sample interval t_s with the initial condition of x_{k-1} and constant inputs of u_{k-1} and d_{k-1}. For a better state estimation, the unmeasured disturbance should also be estimated simultaneously. In general, d can be modeled as the following stochastic difference equation

$$d_k = d_{k-1} + w_{k-1} \tag{6.3}$$

where w_{k-1} is white noise sequence. Combining the Eqn (6.2a) with (6.3), the following augmented model is obtained

$$\begin{bmatrix} x_k \\ d_k \end{bmatrix} = \begin{bmatrix} F_{t_s}(x_{k-1}, u_{k-1}, d_{k-1}) \\ d_{k-1} + w_{k-1} \end{bmatrix} \tag{6.4a}$$

$$y_k = g(x_k, d_k) + v_k \tag{6.4b}$$

where the measurement noise sequence v_k is often to appear in the measured output y_k. By linearizing (6.4) at $\{\hat{x}_{k-1}, \hat{d}_{k-1}\}$, the EKF computes the new estimates $[\hat{x}'_k, \hat{d}'_k]'$ from the feedback measurement y_k and the model prediction (6.4a)

$$\begin{aligned} \begin{bmatrix} x_k \\ d_k \end{bmatrix} &\approx \begin{bmatrix} F_{t_s}\left(\hat{x}_{k-1}, u_{k-1}, \hat{d}_{k-1}\right) \\ \hat{d}_{k-1} \end{bmatrix} \\ &+ \Phi_{k-1} \begin{bmatrix} x_{k-1} - \hat{x}_{k-1} \\ d_{k-1} - \hat{d}_{k-1} \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} w_{k-1} \end{aligned} \tag{6.5a}$$

$$\begin{aligned} y_k &\approx g\left(\hat{x}_{k-1}, \hat{d}_{k-1}\right) \\ &+ \Theta_{k-1} \begin{bmatrix} x_{k-1} - \hat{x}_{k-1} \\ d_{k-1} - \hat{d}_{k-1} \end{bmatrix} + v_k \end{aligned} \tag{6.5b}$$

where

$$\begin{aligned} \Phi_{k-1} &= \begin{bmatrix} \Lambda_{k-1} & \Lambda^d_{k-1} \\ 0 & I \end{bmatrix} \\ \Theta_{k-1} &= \begin{bmatrix} H_{k-1} & H^d_{k-1} \end{bmatrix} \end{aligned} \tag{6.6}$$

Λ_{k-1},Λ^d_{k-1}, H_{k-1} and H^d_{k-1} are calculated using the following formula

$$\Lambda_{k-1} = \frac{\partial F_{t_s}(x,u,d)}{\partial x} \mid_{x=\hat{x}_{k-1},u=u_{k-1},d=\hat{d}_{k-1}} \tag{6.7a}$$

$$\Lambda^d_{k-1} = \frac{\partial F_{t_s}(x,u,d)}{\partial d} \mid_{x=\hat{x}_{k-1},u=u_{k-1},d=\hat{d}_{k-1}} \tag{6.7b}$$

$$H_{k-1} = \frac{\partial g(x,d)}{\partial d} \mid_{x=\hat{x}_{k-1},d=\hat{d}_{k-1}} \tag{6.7c}$$

$$H^d_{k-1} = \frac{\partial g(x,d)}{\partial d} \mid_{x=\hat{x}_{k-1},d=\hat{d}_{k-1}} \tag{6.7d}$$

where $\partial F_{t_s}/\partial x$ and $\partial F_{t_s}/\partial d$ represent Jacobian matrices of F_{t_s} with respect to x and d, respectively. And $\partial g/\partial x$ and $\partial g/\partial d$ represent Jacobian matrices of g with respect to x and d, respectively. Thus, we can conclude the EKF computation procedure which has been well presented and analysed, see Table 6.1.

Table 6.1: EKF for state estimation

Step	Formula
Initialization	$P_0 = \epsilon I$, ϵ is a large number, Initial guess $\hat{x}_0, \hat{d}_0$, Q is the process noise covariance of $\{x, d\}$, R is the measurement noise covariance.
1-Prediction	$\begin{bmatrix} \hat{x}_k \\ \hat{d}_k \end{bmatrix} = \begin{bmatrix} F_{t_s}(\hat{x}_{k-1}, u_{k-1}, \hat{d}_{k-1}) \\ \hat{d}_{k-1} \end{bmatrix}$
2-Correction	$\begin{bmatrix} \hat{x}_k \\ \hat{d}_k \end{bmatrix} = \begin{bmatrix} \hat{x}^-_k \\ \hat{d}^-_k \end{bmatrix} + K_k \left(y_k - g(\hat{x}^-_k, \hat{d}^-_k) \right)$
$P^-_k = \Phi_{k-1} P_{k-1} \Phi'_{k-1} + Q$ $K_k = P^-_k \Theta'_{k-1} \left(\Theta_{k-1} P^-_k \Theta'_{k-1} + R \right)^{-1}$ $P_k = \left(I - K_k \Theta_{k-1} \right) P^-_k$	

6.2.2 Successive Linearization NMPC

By recursively handling the streaming input-output data (u_{k-1}, y_{k-1}), the EKF algorithm could calculate the new estimated states and disturbances $(\hat{x}_k, \hat{d}_k)$. The $(\hat{x}_k, \hat{d}_k)$ is not only used as the nominal values to

linearize the nonlinear first-principle model but also used as the initial conditions of the linearized state-space model in the SL-MPC method. Note that the continuous-time model (6.1) has to be transformed into the discrete-time model for digital MPC design. Two approaches, *first discretise then linearise* and *first linearise then discretise*, have been reported. The EKF algorithm presented in Section 6.2.1 is based on the *first discretise then linearise* approach. In the context of SL-MPC, it is more common to use the *first linearise then discretise*. Here the continuous-time first-principle-based model (6.1) is linearized at current points $\{\hat{x}_k, u_{k-1}, \hat{d}_k\}$,

$$\dot{x} \approx A_c(x - \hat{x}_k) + B_c(u - u_{k-1}) + e_c \tag{6.8a}$$

$$y \approx C(x - \hat{x}_k) + h_c \tag{6.8b}$$

where $A_c = \frac{\partial f}{\partial x}|_{\hat{x}_k, u_{k-1}, \hat{d}_k}$, $B_c = \frac{\partial f}{\partial u}|_{\hat{x}_k, u_{k-1}, \hat{d}_k}$, $e_c = f(\hat{x}_k, u_{k-1}, \hat{d}_k)$, $C = \frac{\partial g}{\partial x}|_{\hat{x}_k, \hat{d}_k}$ and $h_c = g(\hat{x}_k, \hat{d}_k)$. Here the differential eqn (6.8a) has to be discretized for obtaining a discrete-time model. The discretization method includes the exact and the approximated discretization. One common discretization method is the the one-step Euler's approximate method, then a discrete-time model is obtained as follows

$$x_{k+1} = Ax_k + Bu_k + e \tag{6.9a}$$

$$y_k = Cx_k + h \tag{6.9b}$$

where $e = t_s\,(e_c - A_c\hat{x}_k - B_c u_{k-1})$, $A = I + t_s A_c$, $B = t_s B_c$, $h = h_c - C\hat{x}_k$.

Then, a MPC tracking problem formulation is listed as follows

$$\begin{aligned}
\min \quad & \frac{1}{2}\sum_{k=0}^{T-1} \|(y_{k+1} - r_{k+1})\|^2_{W^y} + \|\Delta u_k\|^2_{W^{\Delta u}} \\
\text{s.t.} \quad & \text{Eqn } (6.9a), k = 0, \ldots, T-1 \\
& \text{Eqn } (6.9b), k = 1, \ldots, T \\
& u_k = u_{k-1} + \Delta u_k, k = 0, \ldots, T-1 \\
& y_{\min} \le y_k \le y_{\max}, k = 1, \ldots, T \\
& u_{\min} \le u_k \le u_{\max}, \Delta u_{\min} \le \Delta u_k \le \Delta u_{\max}, k = 0, \ldots, T-1 \\
& x_0 = \hat{x}_k
\end{aligned} \tag{6.10}$$

The MPC tracking problem (6.10) can be solved by many quadratic programming (QP) algorithms, such as the active-set, the interior-point, and the ADMM-based solver. The model parameters $\{A, B, e, C, h\}$ in (6.9) are related to the recursively updated estimates $\{\hat{x}_k, \hat{d}_k\}$ and the last control input u_{k-1}, it means that the MPC tracking problem (6.10) has to be reformulated at each sampling time. Most QP solvers require explicit condensing or sparse MPC-to-QP construction. In such an SL-MPC scenario, the MPC-to-QP construction has to be performed online at each sampling time, as does solve the QP problem. Indeed, often the online MPC-to-QP construction has a comparable computational cost to solving the QP problem itself online, especially when warm-starting strategies are employed. Chapter 2 presented a construction-free *CDAL* solver, that can avoid explicit MPC-to-QP construction in state-space MPC problems.

6.3 Interpretative and Adaptive MPC

Compared to the exact NMPC method, the approximate method, SL-MPC with EKF, could significantly reduce the online computational burden for nonlinear output-feedback MPC problems. Nonetheless, the SL-MPC with EKF involves linearizations twice, and its embedded implementation, including integrator, linearizer, EKF, and time-changing MPC problem solver, remains a difficult task for control engineers.

In the SL-MPC with the EKF method, the model parameters $\{A, B, e, C, h\}$ of the model (6.10) are dependant to the last control input u_{k-1} and the $\{\hat{x}_k, \hat{d}_k\}$, which are recursively estimated by the EKF algorithm. In a sense, the model is recursively updated (or feedback corrected) by the EKF algorithm. And our previous work in Chapter 5 illustrates that an observable linear state-space (SS) model can be equivalently transformed into an ARX model, which will be also introduced in the following subsection 6.3.1. Adding an SS-to-ARX transformation step in SL-MPC with the EKF method would not affect the closed-loop control performance. However, an additional SS-to-ARX transformation step relies on choosing the observer design and additional matrix-matrix mul-

tiplication cost, which makes it no advantage in the online SL-MPC with the EKF method. This inspired us to update the ARX model directly by the EKF algorithm from the streaming input-output data, to avoid an explicit SS-to-ARX transformation. By viewing the ARX model parameters as states of a system, the EKF algorithm could recursively update the ARX model parameters. In fact, a system divides the states and parameters according to their changing rate, states change fast, and parameters change slowly. And this ARX model identification and tracking by the Kalman Filter based algorithm has been widely used in adaptive control [36].

As the EKF algorithm requires a sufficiently small error in initial estimates, and the data-driven identification (initialization) of ARX model is black-box without interpretability like a first-principle-based model, we propose the introduction of the off-line SS-to-ARX transformation in our interpretative and adaptive MPC *IA-MPC* framework. The main steps of our *IA-MPC* framework are listed in Table 6.2. Its *Step* 1 to 3 are performed offline, and the *Step* 1 is to obtain a linear state-space model by linearizing a first-principle-based model at the initial point; the optional *Step* 2 is to obtain a minimal state-space realization by using model reduction when the state dimension is large such as in distributed parameter models or computational fluid dynamic models; After possible state elimination, the *Step* 3 is to obtain an equivalent ARX model by designing an SS-to-ARX transformation. In addition to the gained interpretability and consistent initial estimates, this ARX model acquisition also avoids the difficulty in choosing the ARX model orders in data-driven ARX model identification. The detailed equivalent SS-to-ARX transformation will be illustrated in the following subsection 6.3.1. The *Step* 4 is the online closed-loop MPC control, in which the ARX model parameters are updated by the EKF algorithm from streaming input-output data (see the subsection 6.3.2) and a corresponding ARX-based MPC problem are solved by our previous construction-free *CDAL-ARX* algorithm (see the subsection 6.3.3). For a comparison with the traditional SL-MPC with EKF framework shown in Figure 6.1, the schematic diagram of our *IA-MPC* framework is shown in Figure 6.2.

Table 6.2: Interpretative and Adaptive MPC framework

Step	Detailed description
1	obtain a linear state-space model based on the linearization of a first-principle-based model at the initial point
2(optional)	obtain a minimal realization by using model reduction when states dimension is large such as in distributed parameter models or computational fluid dynamic models
3	design an SS-to-ARX transformation to obtain an equivalent ARX model (robust to process and measurement noises)
4	combine the online EKF update of ARX models and ARX-based MPC algorithms to be used in closed-loop simulations

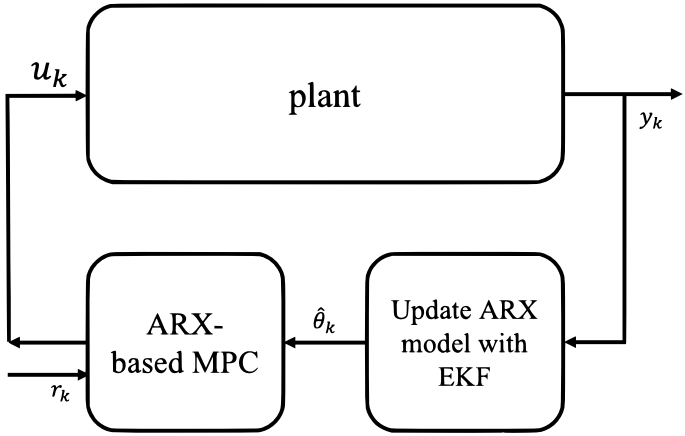


Figure 6.2: Schematic diagram of interpretative and adaptive MPC

6.3.1 Equivalent SS-to-ARX transformation

In Section 5.2.2, we concluded three SS-to-ARX transformations, namely the Cayley-Hamilton-based, Observer-Theory-based, and Kalman-Filter-based transformations. The Cayley-Hamilton-based transformation can derive the unique and equivalent ARX model but with sensitivity to noises. The Observer-Theory-based and Kalman-Filter-based transformations have the same structure in the transformed ARX model, and the latter one requires a larger order than the former one.

6.3.2 ARX model update with decoupled EKF algorithm

By viewing the time-varying ARX model parameters as system states, an ARX model can be reformulated as the following state-space form

$$\theta_k = \theta_{k-1} + w_k \tag{6.11a}$$

$$y_k = \varphi_k' \theta_k + v_k \tag{6.11b}$$

where $\{w_k\}$ is a sequence of independent random vectors and $\{v_k\}$ is a output noise sequence. Although the variances of $\{w_k\}$ and $\{v_k\}$ are unknown in real applications, and the actual parameters $\{\theta_k\}$ may change differently from the above random walk model (6.11a), the EKF algorithm can still work very well [19].

Writing the Eqn (6.11b) as the following separated formulation

$$y_k(j) = \varphi_k' \theta_k(j) + v_k(j), j = 1, \ldots, n_y \tag{6.12}$$

where n_y denotes the dimensions of y, $y_k(j)$ denotes the j-th element of y_k. θ_k and φ_k have the following relationship

$$\begin{aligned} \theta_k &= \left[\begin{array}{cccc} \theta_k(1)' & \theta_k(2)' & \ldots & \theta_k(n_y)' \end{array} \right]' \\ \theta_k(j)' &= [\Psi_1(j,:), \ldots, \Psi_p(j,:), \Omega_1(j,:), \ldots, \Omega_p(j,:), \zeta(j)] \\ \varphi_k' &= \left[y_{k-1}', \ldots, y_{k-p}', u_{k-1}', \ldots, u_{k-p}', 1 \right] \end{aligned} \tag{6.13}$$

The advantage of the above separated ARX formulation is that it allows the EKF algorithm to run in a parallel way and avoids the matrix-inverse

operation, only involving scalar division, compared to the EKF in the SL-MPC (see Tab. 6.1). We list the EKF computation scheme of our *IA-MPC* framework in Tab. 6.3.

Table 6.3: EKF for ARX model updating

Step	Formula
Initialization	$P_0 = \epsilon I$, ϵ is a large number, initial value $\theta(1), \ldots, \theta(n_y)$ from Step 3 in Tab. 6.2, Q is the process noise covariance, r is the measurement noise covariance.
1-Prediction	$\hat{\theta}_k^-(j) = \hat{\theta}_{k-1}(j), j = 1, \ldots, n_y$
2-Correction	$\hat{\theta}_k(j) = \hat{\theta}_k^-(j) + K_k \left(y_k(j) - \varphi_k' \hat{\theta}_k^-(j) \right),$ $j = 1, \ldots, n_y$
	$P_k^- = P_{k-1} + Q$
	$K_k = \frac{1}{\varphi_k' P_k^- \varphi_k + r} P_k^- \varphi_k$
	$P_k = (I - K_k \varphi_k') P_k^-$

6.3.3 Construction-free ARX-based MPC algorithm

In our *IA-MPC* framework, the ARX model is updated by the EKF algorithm at each sampling time. Thus it leads to the corresponding time-

changing ARX-based MPC tracking problem as follows.,

$$
\begin{aligned}
\min \quad & \frac{1}{2}\sum_{k=0}^{T-1} \|(y_{k+1}-r_{k+1})\|^2_{W^y} + \|\Delta u_k\|^2_{W^{\Delta u}} \\
\text{s.t.} \quad & y_k = \sum_{i=1}^{p} \hat{\Psi}_i y_{k-i} + \sum_{i=1}^{p} \hat{\Omega}_i u_{k-i} + \hat{\zeta}, k = 1,\dots,T \\
& u_k = u_{k-1} + \Delta u_k, k = 0,\dots,T-1 \\
& y_{\min} \le y_k \le y_{\max}, k = 1,\dots,T \\
& u_{\min} \le u_k \le u_{\max}, k = 0,\dots,T-1 \\
& \Delta u_{\min} \le \Delta u_k \le \Delta u_{\max}, k = 0,\dots,T-1
\end{aligned} \tag{6.14}
$$

In such a time-changing setting, the computation time spent in constructing its optimization problem and solving itself should be considered together. Indeed, often the online MPC-to-QP construction has a comparable computational cost to solving the MPC problem itself online, especially when warm-starting strategies are employed. Our previous construction-free *CDAL-ARX* algorithm can avoid the explicit MPC-to-QP construction, thus allowing it to be suitable for this scenario. In addition to the notable construction-free feature, our *CDAL-ARX* is also matrix-free and library-free, which makes it practically useful in embedded deployment.

6.4 Numerical Examples

In this section, we test the performance of our proposed IA-MPC method against other two methods, namely the nonlinear MPC method with EKF and the SL-MPC with EKF method. Their MPC solutions are based on CasADi v3.5.5[3] and our previous CDAL algorithm for state-space model based MPC in Chapter 2, respectively, except for the same EKF computational procedure. The reported simulations are executed in MATLAB R2020a on a MacBook Pro with 2.7 GHz 4-core Intel Core i7 and 16GB RAM. Four typical nonlinear MPC numerical examples are used to investigate whether our IA-MPC method works well and provides comparisons with traditional methods.

6.4.1 Problem descriptions

1. *Two tank* problem: we consider the cascaded two tanks system, which is a fluid level control system. The input signal controls the water pump that pumps the water from a reservoir into the upper water tank. The water of the upper water tank also flows through a small opening into the lower water tank. The water of the lower water tank flows through a small opening into the reservoir. Herein without considering the overflow effect, we adopt Bernoulli's principle and conservation of mass to derive the following first-principle model:

$$\begin{aligned} \dot{x}_1 &= -k_1\sqrt{x_1} + k_2 u \\ \dot{x}_2 &= k_1\sqrt{x_1} - k_3\sqrt{x_2} \\ y &= x_2 \end{aligned} \tag{6.15}$$

   where x_1 and x_2 are the water level of the upper and lower water tank, respectively. The full states cannot be measured only the x_2 as the measured output y. u is the input signal, and k_1, k_2 and k_3 are constants depending on the system properties; herein we adopt value 0.5 for all of them. The sampling time of discrete digital control is 0.2 s, and the control goal is to make the output y track the given reference signal subject to the input constraints $0 \leq u \leq 2$ and the input increment constraints $-0.5 \leq \Delta u \leq 0.5$.

2. *Bilinear motor* problem: one common nonlinear control benchmark is a bilinear DC motor plant, whose equation is described as follows,

$$\begin{aligned} \dot{x}_1 &= -\left(R_a/L_a\right) x_1 - \left(k_m/L_a\right) x_2 u + u_a/L_a \\ \dot{x}_2 &= -(B/J)x_2 + \left(k_m/J\right) x_1 u - \tau_l/J \\ y &= x_2 \end{aligned} \tag{6.16}$$

   where x_1 and x_2 are the rotor current and angular velocity, respectively. The full states cannot be measured only the x_2 as the measured output y. The control input u is the stator current. The system parameters are $L_a = 0.314$, $R_a = 12.345$, $k_m = 0.253$,

$J = 0.00441$, $B = 0.00732$, $\tau_l = 1.47$, $u_a = 60$. The bilinearity of (6.16) appears between the state and the control input. The sampling time of discrete digital control is 0.01 s, and the control goal is to make the output y track the given reference signal subject to the input constraints $0 \leq u \leq 2$ and the input increment constraints $-1 \leq \Delta u \leq 1$.

3. *CSTR* problem: one typical nonlinear process control benchmark is a continuous stirred tank reactor (CSTR) problem, which is described by the following continuous-time nonlinear model,

$$\begin{aligned} \dot{C_A} &= C_{A,i} - C_A - k_0 e^{\frac{-EaR}{T}} C_A \\ \dot{T} &= T_i + 0.3T_c - 1.3T + 11.92k_0 e^{\frac{-EaR}{T}} C_A \\ y &= T \end{aligned} \tag{6.17}$$

where C_A is the concentration of reagent A, T is the temperature of the reactor, $C_{A,i}$ is the inlet feed stream concentration, which is assumed to have the constant value 10.0 kgmol/m^3. The unmeasured disturbance comes from the inlet feed stream temperature T_i, which has slow fluctuations represented by $T_i = 298.15 + 5\sin(0.05t)\ K$. The manipulated variable is the coolant temperature T_c. The constants $k_0 = 34930800$ and $EaR = -5963.6$ (in MKS units). The sampling time of discrete digital control is 0.5 s, and the control goal is to manipulate the coolant temperature T_c to track a higher temperature of the reactor (equals a higher conversion rate) as well as reject the unmeasured disturbance T_i. The physical constraints come from the input increment constraints $-1 \leq \Delta T_c \leq 1$. Note that the unmeasured disturbance T_i is estimated by the EKF algorithm in SL-MPC and NMPC methods.

4. *Van der Pol oscillator* problem: we consider the nonlinear Van der Pol oscillator with a time-varying parameter, and its dynamics are given by,

$$\begin{aligned} \dot{x} &= v \\ \dot{v} &= \mu(t)(1 - x^2)v - x + u \end{aligned} \tag{6.18}$$

where x is the position, v is the velocity, u is the control input and $\mu(t)$ is a piecewise function defined as

$$\mu(t) = \begin{cases} 1 & \text{if } t \leq 50 \\ 3 & \text{if } t > 50 \end{cases}$$

The sampling time of discrete digital control is 0.2 s, and the control goal is to make the output y track the given reference signal subject to the input constraints $-10 \leq u \leq 10$ and the input increment constraints $-10 \leq \Delta u \leq 10$. Note that the unmeasured time-varying parameter $\mu(t)$ is estimated by the EKF algorithm in SL-MPC and NMPC methods.

6.4.2 Problems settings

The above four problems share the same MPC and EKF settings: the MPC prediction horizon $N_p = 10$, the MPC output weight $W_y = 10$ and the MPC input increment weight $W_{\Delta u} = 0.1$, the initial error covariance matrix of EKF $P_0 = 10I$, the process noise covariance matrix of EKF $Q = 0.01I$ and the measurement noise covariance of EKF $R = 0.01$. We consider two simulation scenarios, that is, without and with process noises, respectively.

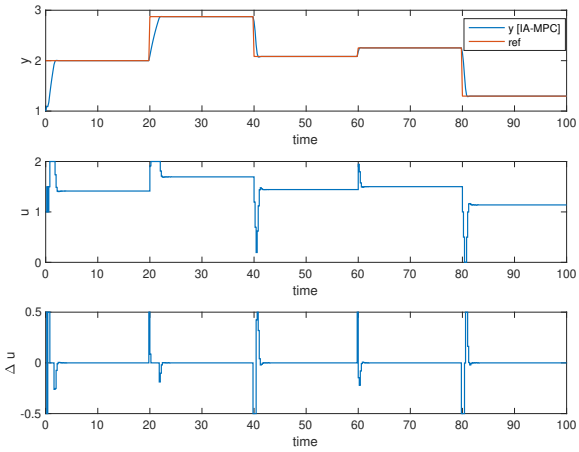
1. *Two tank* problem: its process noise scenario is given a random noise $0.05\times$rand(2,1), and the initial conditions are $x_1 = 1$, $x_2 = 1$ and $u = 1$ for obtaining the linearized state-space model, which is then transformed into the ARX model with order $p = 3$ by using poles $[0.01, 0.02]$. Its tracking signal is randomly selected every 20 s in the range $[1, 3]$.

2. *Bilinear motor* problem: its process noise scenario is given a random noise $1\times$rand(2,1), and the initial conditions are $x_1 = 5.2542$, $x_2 = -19.2205$ and $u = 1$ for obtaining the linearized state-space model, which is then transformed into the ARX model with order $p = 5$ by using poles $[0.05, 0.1]$. Its tracking signal is randomly selected every 0.4 s in the range $[-10, 10]$.

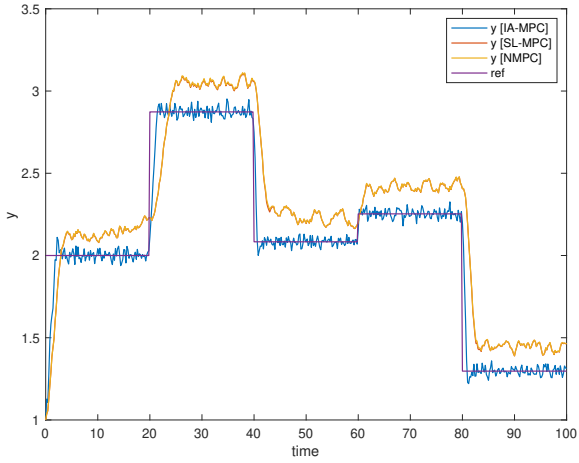
3. *CSTR* problem: its process noise scenario is given a random noise $0.1\times$rand(2,1), and the initial state of the reactor is at a low conversion rate, with $C_A = 8.57$ kgmol/m^3, $T = 311$ K, and then the linearized state-space model around the initial conditions is transformed into the ARX model with order $p = 3$ by using poles $[0.01, 0.02]$. Its tracking signal gradually changes from 311.2639 K to 370 K during 50 100 s and then holds constant.

4. *Van der Pol oscillator* problem: its process noise scenario is given a random noise $1\times$rand(2,1), and the initial conditions are $x_1 = 0$, $x_2 = 0$ and $u = 0$ for obtaining the linearized state-space model, which is then transformed into the ARX model with order $p = 3$ by using poles $[0.005, 0.01]$. Its tracking signal switches between 0 and 1 every 10 s.

Their simulation results are plotted in Figure 6.3, 6.4, 6.5 and 6.6, respectively. The *Two tank* and *Bilinear motor* problem have no unknown disturbances, and the NMPC, SL-MPC, and our IA-MPC method generate the same offset-free tracking performances subject to the input constraints in their noise-free scenario, which are plotted in Figure 6.3(a) and 6.4(a). The *CSTR* and *Van der Pol oscillator* problems have the unknown time-varying disturbance. It is found that our IA-MPC method generates better offset-free tracking performances than the NMPC and SL-MPC method under the given EKF setting in their noise-free scenario, which are plotted in Figure 6.5(a) and 6.6(a). Under the process noises, our IA-MPC method generates better noise robustness than the NMPC and SL-MPC method in the four problems, which are plotted in Figure 6.3(b), 6.4(b), 6.5(b) and 6.6(b). As expected, the online computation time of the exact NMPC method is much longer than the approximate SL-MPC and IA-MPC method, and the online computation time of the SL-MPC and IA-MPC method are almost the same since they both utilized our developed construction-free CDAL method, having no online construction time.

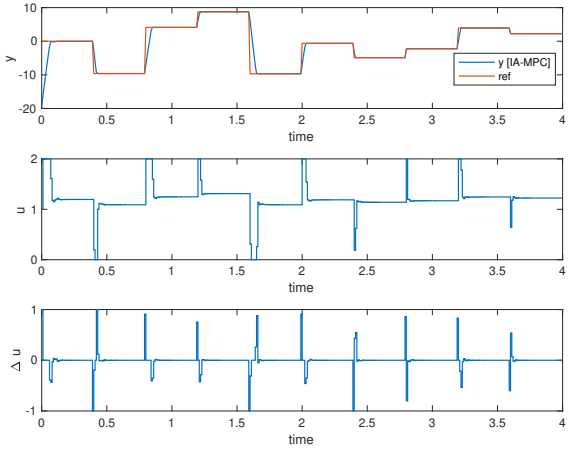


(a) Tracking performance of IA-MPC method without noise

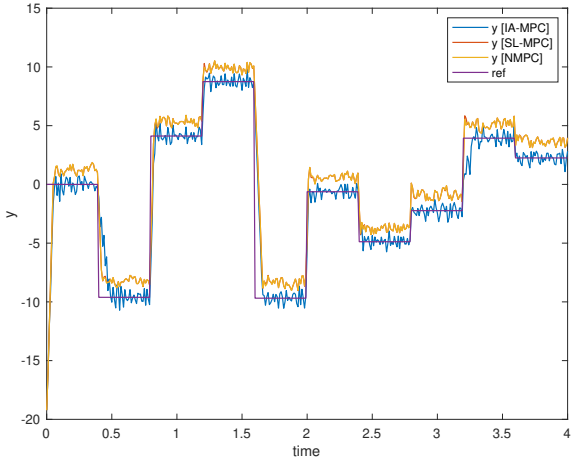


(b) Comparison of IA-MPC, SL-MPC, and NMPC methods with noise

Figure 6.3: The closed-loop control performances in the *Two tank* problem

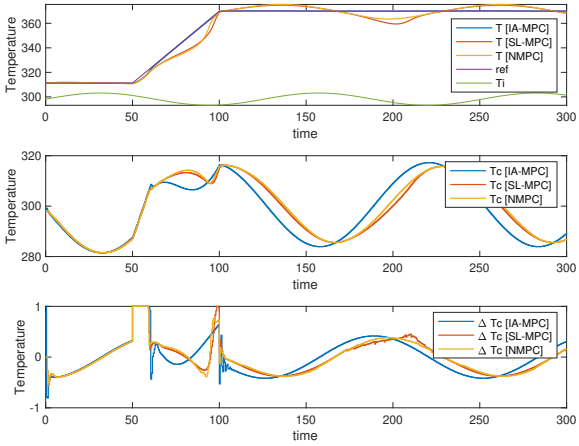


(a) Tracking performance of IA-MPC method without noise

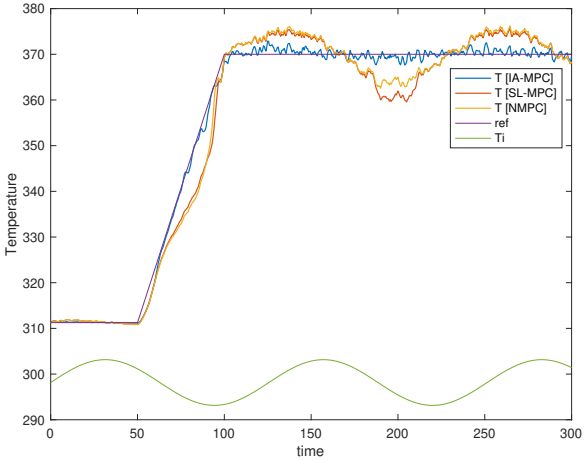


(b) Comparison of IA-MPC, SL-MPC, and NMPC methods with noise

Figure 6.4: The closed-loop control performances in the *Bilinear motor* problem

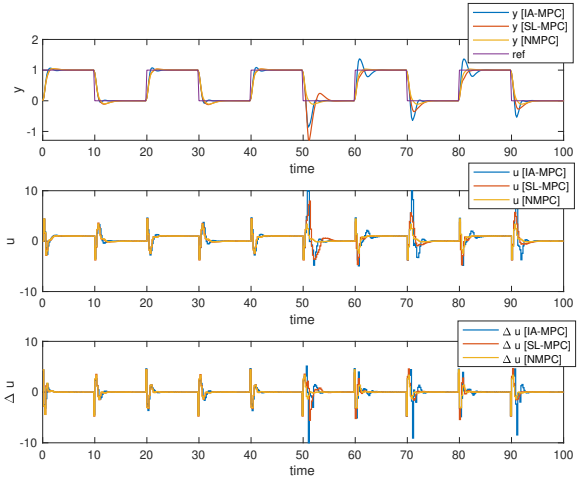


(a) Comparison of IA-MPC, SL-MPC, and NMPC methods without noise

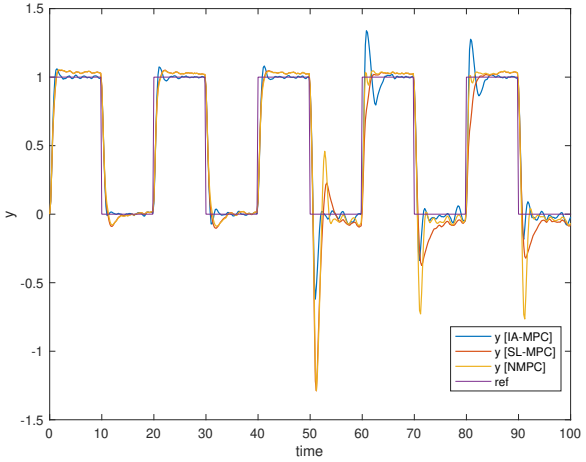


(b) Comparison of IA-MPC, SL-MPC, and NMPC methods with noise

Figure 6.5: The closed-loop control performances in the *CSTR* problem



(a) Comparison of IA-MPC, SL-MPC, and NMPC methods without noise



(b) Comparison of IA-MPC, SL-MPC, and NMPC methods with noise

Figure 6.6: The closed-loop control performance in the *Van der Pol* problem

6.5 Conclusion

This chapter proposed a novel interpretative and adaptive MPC (IA-MPC) method for nonlinear systems, which was inspired by the SL-MPC with EKF method. In our IA-MPC method, a linear state-space model is firstly obtained by performing the linearization of a first-principle-based model at the initial point, and then an equivalent ARX model is obtained via the SS-to-ARX transformation. This novel initialization of ARX model allows us to keep the interpretability of a first-principle-based model and the adaptivity of the ARX model. The closed-loop control involves the EKF algorithm to update the ARX model parameters recursively and our previously developed construction-free CDAL-ARX algorithm to calculate the MPC control input. Note that our proposed implementation of our IA-MPC method is well suited for embedded platforms, thanks to its library-free C-code simple enough. The effectiveness of our IA-MPC method was illustrated by four nonlinear typical benchmarks.

Chapter 7

Conclusion

The main objective of this thesis was to address the high computation burden of MPC in nonlinear systems and the difficulty in deploying NMPC onto embedded platforms.

7.1 Summary of contributions

- Chapter 2 presented a construction-free, matrix-free, and library-free MPC solver, based on a cyclic coordinate-descent method in the augmented Lagrangian framework. We showed that the method is efficient and competes with other existing methods, thanks to the use of a reverse cyclic rule, Nesterov's acceleration, a simple heuristic preconditioner, and an efficient coupling scheme. Compared to many QP solution methods proposed in the literature, CDAL avoids constructing the QP problem, which makes it particularly appealing for some scenarios in which its online construction is required and has a comparable computation time to solving itself.

- Chapter 3 showed the development of a rapid-prototype MPC tool based on gPROMS platform. Our gPROMS-MPC tool not only interacts directly with gPROMS first-principle-based models for closed-loop simulations but also utilizes the mathematical information of

gPROMS first-principle-based models to derive the linearized state-space model for MPC design. Our gPROMS-MPC tool allows users to choose when to linearize, such as performing linearization at each sampling time (called successive linearization) or at some specific points to obtain one or multiple good linear models for linear MPC design. Compared to previous approaches, using simulation data of gPROMS first-principle-based models to obtain linear models by performing system identification like the PAROC framework, the approach of our gPROMS-MPC tool is not only time-saving but also interpretative. Considering possible successive linearization or high state-dimension cases, our previous construction-free *CDAL* and the online parametric active-set *qpOASES* algorithm are implemented in our gPROMS-MPC tool, based on sparse and condensed QP formulations from MPC, respectively. Note that our construction-free *CDAL* is also matrix-free and library-free, thus having the support for embedded C-code generation.

- Chapter 4 introduced a solution algorithm for solving MPC problems based on ARX models that avoid constructing the associated QP problem explicitly. Due to its matrix-free and library-free features, the proposed CDAL-ARX algorithm can be useful in adaptive embedded linear MPC applications based on ARX models, especially when combined with a fast and robust recursive linear identification method.

- Chapter 5 theoretically introduced the Cayley-Hamilton, Observer-Theory, and Kalman Filter based SS-to-ARX transformations. Numerical experiment results of the AFTI-16 are presented to show the equivalence between SS-based and ARX-based MPC problems and analyze the robustness of different SS-to-ARX transformations to noises. This equivalence is suggested to be used in an interpretative and adaptive MPC framework. That is, an interpretative SS model is firstly obtained from linearizing a first-principle-based model and transformed into an equivalent ARX model, and then this equivalent ARX-based MPC controller can be updated online.

- Chapter 6 proposed a novel interpretative and adaptive MPC (IA-MPC) method for nonlinear systems, which was inspired by the SL-MPC-EKF method. In our IA-MPC method, a linear state-space model is first obtained by performing the linearization of a first-principle-based model at the initial point, and then an equivalent ARX model is obtained via the SS-to-ARX transformation. This novel initialization of ARX model allows us to keep the interpretability of a first-principle-based model and the adaptivity of the ARX model. The closed-loop control involves the EKF algorithm to update the ARX model parameters recursively and our previously developed construction-free CDAL-ARX algorithm to calculate the MPC control input. Note that the proposed implementation of our IA-MPC method is well suited for embedded platforms, thanks to its library-free C-code simple enough. The effectiveness of our IA-MPC method was illustrated by four nonlinear typical benchmarks.

7.2 Open problems for future research

Based on the contributions of our methods and algorithms discussed in this thesis, in this section, we discuss possible future works and relevant problems that remain to be addressed, and further research is strongly encouraged.

- **Extension to nonlinear optimal control problem**: Our proposed CDAL and CDAL-ARX algorithm are developed for on linear state-space and ARX-based MPC problems. Although the online successive linearization strategy for handling the model nonlinearity is a good choice and suitable for our construction-free CDAL and CDAL-ARX algorithm, a broader class of nonlinear optimal control problems, involving nonlinear objective index or constraints, widely exists in practical applications, like motion planning and trajectory optimization in robotics. It was shown that the Augmented Lagrangian framework could be extended for general nonlinear programming problems. In our future work, based on the

CDAL algorithm framework, some modifications, like online adaptive changing of ρ parameter value, will be used to develop an efficient and robust nonlinear optimal control algorithm.

- **More practical industrial applications**: The gPROMS modeling platform has been widely used in the process industry, such as fuel cell energy, natural gas processing, oilfield, formulated product manufacture, etc. In their control design workflow, our rapid-prototype gPROMS-MPC tool could play an important role in easing the difficulties of MPC technology implementation. Furthermore, our future work will mainly focus on extending our gPROMS-MPC tool to solve the economic MPC problem, which often appears in process industries.

- **Deep ReLUs based LPV-ARX identification for MPC** The linear parameter varying (LPV) approach is widely used in the control field, including MPC, because it can take advantage of well-established linear controller design methods. This allows, for example, the application of linear MPC schemes with lower computational costs in MPC for nonlinear plants. And the LPV-MPC approach is one of the successful approaches for handling nonlinear plants, even for strongly nonlinear plants, due to the intrinsic robustness of MPC. The two LPV modeling approaches are state-space and input-output, and the input-output LPV-ARX model is preferred in practical use. Most existing data-driven LPV-ARX modeling technique resort to mixed integer programming techniques to learn different regions. The modern deep neural network (DNN) theory tells us that Deep ReLU networks can represent exponentially many more linear regions than shallow ones for a fixed amount of memory. Our future work will utilize the efficient representation and powerful learning ability of Deep ReLUs, to develop a Deep ReLUs-based LPV-ARX identification framework for MPC. And our CDAL-ARX algorithm can then be used to solve the corresponding LPV-ARX-based MPC problems.

- **Applications to large-scale systems**: In the first-principle-based

modeling field, the distributed parameter models and computational fluid dynamic models are often used to describe the spatial-time equations, resulting to a complicated integral, partial differential, and algebraic equations (IPDAEs). For these large-scale systems, some auto-differential software, like gPROMS, could be used to derive their large-scale linear state-space models over different operating ranges. And then, our illustrated SS-to-ARX equivalence theorem can be used to obtain extremely compact input-output LPV-ARX models. This analytical derivation of LPV-ARX models could exploit hundreds and thousands of existing first-principle-based models and are of interpretability, unlike the data-driven paradigms. And our CDAL-ARX algorithm can then be used to solve the corresponding LPV-ARX-based MPC problems.

- **Guaranteed feasibility and stability analysis for IA-MPC**: The focus of this thesis was on an efficient algorithm for MPC in nonlinear systems. Like the SL-MPC with EKF method lacks a rigorous theoretical stability guarantee but is still widely adopted in many practical applications for its effectiveness. Its connected IA-MPC method, presented in this thesis, is also practically useful. Our future work will also focus on analyzing the stability guarantee of the IA-MPC framework, borrowing from work such as adaptive control of linear time-varying systems. Except for the stability guarantee, the feasibility of online MPC problems possibly occurs, which may be due to hard output constraints. Our future extensions of the approach to handle soft output constraints, are to introduce slack variables in the output constraints in our CDAL algorithm that simultaneously handles the feasibility and the full-rank requirement 1. Moreover, we will also exploit the covariance matrix that directly quantifies the uncertainty associated with ARX model parameters to set up robust MPC schemes.

Bibliography

[1] P. Amodio and F. Mazzia. "A parallel Gauss–Seidel method for block tridiagonal linear systems". In: *SIAM Journal on Scientific Computing* 16.6 (1995), pp. 1451–1461.

[2] J.A.E Andersson et al. "A condensing algorithm for nonlinear MPC with a quadratic runtime in horizon length". In: *Automatica* (2013), pp. 97–100.

[3] J.A.E. Andersson et al. "CasADi – A software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation* 11.1 (2019), pp. 1–36.

[4] M. Aoki and A. Havenner. "State space modeling of multiple time series". In: *Econometric Reviews* 10.1 (1991), pp. 1–59.

[5] *Aspen Tech, Aspen plus*. `http://www.aspentech.com/products/aspen-plus.aspx`. 2001–2022.

[6] K.J. Åström and B. Wittenmark. *Adaptive control*. Courier Corporation, 2013.

[7] A. Bemporad. "A Numerically Stable Solver for Positive Semi-Definite Quadratic Programs Based on Nonnegative Least Squares". In: *IEEE Transactions on Automatic Control* 63.2 (2018), pp. 525–531.

[8] A. Bemporad. "A Quadratic Programming Algorithm Based on Nonnegative Least Squares with Applications to Embedded Model Predictive Control". In: *IEEE Transactions on Automatic Control* 61.4 (2016), pp. 1111–1116.

[9] A. Bemporad. "Recurrent Neural Network Training with Convex Loss and Regularization Functions by Extended Kalman Filtering". In: *arXiv preprint arXiv:2111.02673* (2021).

[10] A. Bemporad, F. Borrelli, and M. Morari. "Model predictive control based on linear programming˜ the explicit solution". In: *IEEE transactions on automatic control* 47.12 (2002), pp. 1974–1985.

[11] A. Bemporad, A. Casavola, and E. Mosca. "Nonlinear control of constrained linear systems via predictive reference management". In: *IEEE transactions on Automatic Control* 42.3 (1997), pp. 340–349.

[12] A. Bemporad and M. Morari. "Robust model predictive control: A survey". In: *Robustness in identification and control*. Springer, 1999, pp. 207–226.

[13] A. Bemporad, M. Morari, and N.L Ricker. "Model predictive control toolbox". In: *User's Guide, Version* 2 (2004).

[14] A. Bemporad, M. Morari, and E.N. Pistikopoulos V. Dua. "The explicit linear quadratic regulator for constrained systems". In: *Automatica* 38.1 (2002), pp. 3–20.

[15] D.P. Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[16] D.P. Bertsekas. "Nonlinear programming". In: *Journal of the Operational Research Society* 48.3 (1997), pp. 334–334.

[17] S. Boyd et al. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[18] E.F Camacho and C. Bordons. "Nonlinear model predictive control". In: *Model Predictive control*. Springer, 2007, pp. 249–288.

[19] L. Cao and H.M Schwartz. "Analysis of the Kalman filter based estimation algorithm: an orthogonal decomposition approach". In: *Automatica* 40.1 (2004), pp. 5–19.

[20] K.W. Chang, C.J. Hsieh, and C.J. Lin. "Coordinate Descent Method for Large-scale L2-loss Linear Support Vector Machines." In: *Journal of Machine Learning Research* 9.7 (2008).

[21] L. Chisci, P. Falugi, and G. Zappa. "Gain-scheduling MPC of nonlinear systems". In: *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal* 13.3-4 (2003), pp. 295–308.

[22] D.W. Clarke, C. Mohtadi, and P.S. Tuffs. "Generalized Predictive Control—Part I. The basic algorithm". In: *Automatica* 23.2 (1987), pp. 137–148.

[23] C.R. Cutler and B.L. Ramaker. “Dynamic matrix control-A computer control algorithm”. In: *joint automatic control conference*. 17. 1980, p. 72.

[24] Simulink Documentation. *Simulation and Model-Based Design*. 2020. URL: https://www.mathworks.com/products/simulink.html.

[25] U. Eren et al. “Model predictive control in aerospace systems: Current state and opportunities”. In: *Journal of Guidance, Control, and Dynamics* 40.7 (2017), pp. 1541–1566.

[26] P. Falcone et al. “Linear time-varying model predictive control and its application to active steering systems: Stability analysis and experimental validation”. In: *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal* 18.8 (2008), pp. 862–875.

[27] P. Falcone et al. “Predictive active steering control for autonomous vehicle systems”. In: *IEEE Transactions on control systems technology* 15.3 (2007), pp. 566–580.

[28] H.J. Ferreau, H.G. Bock, and M. Diehl. “An online active set strategy to overcome the limitations of explicit MPC”. In: *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal* 18.8 (2008), pp. 816–830.

[29] H.J. Ferreau et al. “qpOASES: A parametric active-set algorithm for quadratic programming”. In: *Mathematical Programming Computation* 6.4 (2014), pp. 327–363.

[30] S.F. Fux et al. “EKF based self-adaptive thermal model for a passive house”. In: *Energy and Buildings* 68 (2014), pp. 811–817.

[31] F.R Gantmakher. *The theory of matrices*. Vol. 131. American Mathematical Soc., 1959.

[32] C.E. Garcia, D.M. Prett, and M. Morari. “Model predictive control: Theory and practice—A survey”. In: *Automatica* 25.3 (1989), pp. 335–348.

[33] T. Geyer. *Model predictive control of high power converters and industrial drives*. John Wiley & Sons, 2016.

[34] P. Giselsson and S. Boyd. “Diagonal scaling in Douglas-Rachford splitting and ADMM”. In: *53rd IEEE Conference on Decision and Control*. IEEE. 2014, pp. 5033–5039.

[35] S. Gros et al. “From linear to nonlinear MPC: bridging the gap via the real-time iteration”. In: *International Journal of Control* 93.1 (2020), pp. 62–80.

[36] L. Guo. “Estimating time-varying parameters by the Kalman filter based algorithm: stability and convergence”. In: *IEEE Transactions on Automatic Control* 35.2 (1990), pp. 141–147.

[37] L. Guo and L. Ljung. “Performance analysis of general tracking algorithms”. In: *IEEE Transactions on Automatic Control* 40.8 (1995), pp. 1388–1402.

[38] M. Gurbuzbalaban et al. “When cyclic coordinate descent outperforms randomized coordinate descent”. In: *Advances in Neural Information Processing Systems* 30 (2017).

[39] B. He and X. Yuan. “On the acceleration of augmented Lagrangian method for linearly constrained optimization”. In: *Optimization online* 3 (2010).

[40] B. Hermans, A. Themelis, and P. Patrinos. “QPALM: a Newton-type proximal augmented Lagrangian method for quadratic programs”. In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. 2019, pp. 4325–4330.

[41] B. Houska, H.J. Ferreau, and M. Diehl. “An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range”. In: *Automatica* 47.10 (2011), pp. 2279–2285.

[42] C.J. Hsieh et al. “A dual coordinate descent method for large-scale linear SVM”. In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 408–415.

[43] J.K. Huusom et al. “ARX-model based model predictive control with offset-free tracking”. In: *Computer Aided Chemical Engineering*. Vol. 28. Elsevier, 2010, pp. 601–606.

[44] E. Kaiser, J.N. Kutz, and S.L. Brunton. “Data-driven discovery of Koopman eigenfunctions for control”. In: *Machine Learning: Science and Technology* 2.3 (2021), p. 035023.

[45] R.E. Kalman. “Contributions to the theory of optimal control”. In: *Bol. soc. mat. mexicana* 5.2 (1960), pp. 102–119.

[46] R.E. Kalman. *Lectures on controllability and observability*. Tech. rep. STANFORD UNIV CA DEPT OF OPERATIONS RESEARCH, 1970.

[47] M. Kang, M. Kang, and M. Jung. "Inexact accelerated augmented Lagrangian methods". In: *Computational Optimization and Applications* 62.2 (2015), pp. 373–404.

[48] P. Kapasouris, M. Athans, and G. Stein. "Design of feedback control systems for stable plants with saturating actuators". In: *Proceedings of the 27th IEEE Conference on Decision and Control*. 1988, 469–479 vol.1. DOI: 10.1109/CDC.1988.194356.

[49] B. Karg and S. Lucia. "Efficient representation and approximation of model predictive control laws via deep learning". In: *IEEE Transactions on Cybernetics* 50.9 (2020), pp. 3866–3878.

[50] S. Katayama and T. Ohtsuka. "Automatic Code Generation Tool for Nonlinear Model Predictive Control with Jupyter". In: *IFAC-PapersOnLine* 53.2 (2020), pp. 7033–7040.

[51] M. Korda and I. Mezić. "Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control". In: *Automatica* 93 (2018), pp. 149–160.

[52] D. Kouzoupis et al. "First-order methods in embedded nonlinear model predictive control". In: *2015 European Control Conference (ECC)*. IEEE. 2015, pp. 2617–2622.

[53] D. Kouzoupis et al. "Towards proper assessment of QP algorithms for embedded model predictive control". In: *2015 European Control Conference (ECC)*. IEEE. 2015, pp. 2609–2616.

[54] F. Kuhne, W.F. Lages, and J.G. da Silva Jr. "Model predictive control of a mobile robot using linearization". In: *Proceedings of mechatronics and robotics* 4.4 (2004), pp. 525–530.

[55] G. Lan, Guanghui, and R.D.C. Monteiro. "Iteration-complexity of first-order augmented Lagrangian methods for convex programming". In: *Mathematical Programming* 155.1 (2016), pp. 511–547.

[56] J.H. Lee. "Model predictive control: Review of the three decades of development". In: *International Journal of Control, Automation and Systems* 9.3 (2011), pp. 415–424.

[57] J.H. Lee and N.L. Ricker. "Extended Kalman filter based nonlinear model predictive control". In: *Industrial & Engineering Chemistry Research* 33.6 (1994), pp. 1530–1541.

[58] W. Li and J. Swetits. "A new algorithm for solving strictly convex quadratic programs". In: *SIAM Journal on Optimization* 7.3 (1997), pp. 595–619.

[59] R.K. Lim and M.Q. Phan. "Identification of a multistep-ahead observer and its application to predictive control". In: *Journal of guidance, control, and dynamics* 20.6 (1997), pp. 1200–1206.

[60] L. Ljung. "Asymptotic behavior of the extended Kalman filter as a parameter estimator for linear systems". In: *IEEE Transactions on Automatic Control* 24.1 (1979), pp. 36–50.

[61] L. Ljung. "Black-box models from input-output measurements". In: *IMTC 2001. Proceedings of the 18th IEEE instrumentation and measurement technology conference. Rediscovering measurement in the age of informatics (Cat. No. 01CH 37188)*. Vol. 1. IEEE. 2001, pp. 138–146.

[62] S. Lucia and B. Karg. "A deep learning-based approach to robust nonlinear model predictive control". In: *IFAC-PapersOnLine* 51.20 (2018), pp. 511–516.

[63] D.G Luenberger. "Linear and nonlinear programming Addison-Wesley". In: *Reading, MA* (1984).

[64] Z.Q. Luo and P. Tseng. "On the convergence of a matrix splitting algorithm for the symmetric monotone linear complementarity problem". In: *SIAM Journal on Control and Optimization* 29.5 (1991), pp. 1037–1060.

[65] Z.Q. Luo and P. Tseng. "On the convergence of the coordinate descent method for convex differentiable minimization". In: *Journal of Optimization Theory and Applications* 72.1 (1992), pp. 7–35.

[66] B. Lusch, J.N. Kutz, and S.L. Brunton. "Deep learning for universal linear embeddings of nonlinear dynamics". In: *Nature communications* 9.1 (2018), pp. 1–10.

[67] R. Findeisen M. Kögel. "Fast predictive control of linear systems combining Nesterov's gradient method and the method of multipliers". In: *2011 50th IEEE Conference on Decision and Control and European Control Conference*. IEEE. 2011, pp. 501–506.

[68] D.Q. Mayne. "Model predictive control: Recent developments and future promise". In: *Automatica* 50.12 (2014), pp. 2967–2986.

[69] M. Morari and J.H. Lee. "Model predictive control: past, present and future". In: *Computers & Chemical Engineering* 23.4-5 (1999), pp. 667–682.

[70] V. Nedelcu, I. Necoara, and Q. Tran-Dinh. "Computational complexity of inexact gradient augmented Lagrangian methods: application to constrained MPC". In: *SIAM Journal on Control and Optimization* 52.5 (2014), pp. 3109–3134.

[71] Y. Nesterov. "A method of solving a convex programming problem with convergence rate $O(1/k^2)$". In: *Soviet Mathematics Doklady* 27.2 (1983), pp. 372–376.

[72] Y. Nesterov. "Efficiency of coordinate descent methods on huge-scale optimization problems". In: *SIAM Journal on Optimization* 22.2 (2012), pp. 341–362.

[73] Y. Nesterov. "Smooth minimization of non-smooth functions". In: *Mathematical programming* 103.1 (2005), pp. 127–152.

[74] M. Nørgaard, N.K. Poulsen, and O. Ravn. "New developments in state estimation for nonlinear systems". In: *Automatica* 36.11 (2000), pp. 1627–1638.

[75] J. Nutini, I. Laradji, and M. Schmidt. "Let's Make Block Coordinate Descent Go Fast: Faster Greedy Rules, Message-Passing, Active-Set Complexity, and Superlinear Convergence". In: *arXiv preprint arXiv:1712.08859* (2017).

[76] J. Nutini et al. "Coordinate descent converges faster with the Gauss-Southwell rule than random selection". In: *International Conference on Machine Learning*. PMLR. 2015, pp. 1632–1641.

[77] M. Oh and C.C. Pantelides. "A modelling and simulation language for combined lumped and distributed parameter systems". In: *Computers & Chemical Engineering* 20.6-7 (1996), pp. 611–633.

[78] T. Ohtsuka. "A continuation/GMRES method for fast computation of nonlinear receding horizon control". In: *Automatica* 40.4 (2004), pp. 563–574.

[79] T. Ohtsuka. "A tutorial on C/GMRES and automatic code generation for nonlinear model predictive control". In: *2015 European Control Conference (ECC)*. IEEE. 2015, pp. 73–86.

[80] P. Van Overschee and B. De Moor. "N4SID: Subspace algorithms for the identification of combined deterministic-stochastic systems". In: *Automatica* 30.1 (1994), pp. 75–93.

[81] P. Patrinos and A. Bemporad. "An accelerated dual gradient-projection algorithm for embedded linear model predictive control". In: *IEEE Transactions on Automatic Control* 59.1 (2013), pp. 18–33.

[82] B.M. Pfeiffer et al. "Nonlinear model predictive control based on existing mechanistic models of polymerisation reactors". In: *IFAC-PapersOnLine* 53.2 (2020), pp. 6076–6081.

[83] M. Phan and J.N. Juang. "Predictive feedback controllers for stabilization of linear multivariable systems". In: *Guidance, Navigation, and Control Conference*. 1996, p. 3769.

[84] M. Phan et al. "Linear system identification via an asymptotically stable observer". In: *Journal of Optimization Theory and Applications* 79.1 (1993), pp. 59–86.

[85] M.Q. Phan, R.K. Lim, and R.W. Longman. "Unifying input-output and state-space perspectives of predictive control". In: *Department of mechanical and aerospace engineering technical report* 3044 (1998).

[86] M.Q. Phan and R.W. Longman. "Relationship Between State-space And Input-output Models Via Observer Markov Parameters". In: *WIT Transactions on The Built Environment* 22 (1970).

[87] E.N. Pistikopoulos et al. "PAROC—An integrated framework and software platform for the optimisation and advanced model-based control of process systems". In: *Chemical Engineering Science* 136 (2015), pp. 115–138.

[88] S.J. Qin and T.A. Badgwell. "A survey of industrial model predictive control technology". In: *Control engineering practice* 11.7 (2003), pp. 733–764.

[89] S.J. Qin and T.A. Badgwell. "An overview of nonlinear model predictive control applications". In: *Nonlinear model predictive control* (2000), pp. 369–392.

[90] R. Quirynen et al. "Autogenerating microsecond solvers for nonlinear MPC: a tutorial using ACADO integrators". In: *Optimal Control Applications and Methods* 36.5 (2015), pp. 685–704.

[91] R. Ramine and K.M. Raman. "Model algorithmic control (MAC); basic theoretical properties". In: *Automatica* 18.4 (1982), pp. 401–414.

[92] J.B. Rawlings and B.R Bakshi. "Particle filtering and moving horizon estimation". In: *Computers & chemical engineering* 30.10-12 (2006), pp. 1529–1541.

[93] K. Reif et al. "Stochastic stability of the discrete-time extended Kalman filter". In: *IEEE Transactions on Automatic control* 44.4 (1999), pp. 714–728.

[94] J. Richalet et al. “Model predictive heuristic control: Applications to industrial processes”. In: *Automatica* 14.5 (1978), pp. 413–428.

[95] P. Richtárik, Peter, and M. Takáč. “Distributed coordinate descent method for learning with big data”. In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 2657–2681.

[96] S. Richter, C.N. Jones, and M. Morari. “Computational complexity certification for real-time MPC with input constraints based on the fast gradient method”. In: *IEEE Transactions on Automatic Control* 57.6 (2011), pp. 1391–1403.

[97] N. Saraf and A. Bemporad. “A bounded-variable least-squares solver based on stable QR updates”. In: *IEEE Transactions on Automatic Control* 65.3 (2020), pp. 1242–1247.

[98] N. Saraf and A. Bemporad. “An efficient bounded-variable nonlinear least-squares algorithm for embedded MPC”. In: *Automatica* 141 (2022), p. 110293.

[99] N. Saraf and A. Bemporad. “Fast model predictive control based on linear input/output models and bounded-variable least squares”. In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE. 2017, pp. 1919–1924.

[100] D.E Seborg et al. *Process dynamics and control*. John Wiley & Sons, 2016.

[101] T. Serra, C. Tjandraatmadja, and S. Ramalingam. “Bounding and counting linear regions of deep neural networks”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 4558–4566.

[102] *Siemens Process Systems Enterprise, gPROMS*. `https://www.psenterprise.com/products/gproms`. Siemens 1997–2022.

[103] B. Stellato et al. “OSQP: An Operator Splitting Solver for Quadratic Programs”. In: *Mathematical Programming Computation* 12 (2020). `http://arxiv.org/abs/1711.08013`, Code avaliable at `https://github.com/oxfordcontrol/osqp`. Awarded best paper of the journal for year 2020., pp. 637–672.

[104] B. Stellato et al. “OSQP: An operator splitting solver for quadratic programs”. In: *Mathematical Programming Computation* 12.4 (2020), pp. 637–672.

[105] R. Takapoui and H. Javadi. “Preconditioning via diagonal scaling”. In: *arXiv preprint arXiv:1610.03871* (2016).

[106] M. Verhaegen. “Identification of the deterministic part of MIMO state space models given in innovations form from input-output data”. In: *Automatica* 30.1 (1994), pp. 61–74.

[107] E.A. Wan and A.T. Nelson. “Dual extended Kalman filter methods”. In: *Kalman filtering and neural networks* 123 (2001).

[108] Y. Wang and S. Boyd. “Fast model predictive control using online optimization”. In: *IEEE Transactions on control systems technology* 18.2 (2009), pp. 267–278.

[109] T.A. Wenzel et al. “Dual extended Kalman filter for vehicle state and parameter estimation”. In: *Vehicle system dynamics* 44.2 (2006), pp. 153–171.

[110] S.J. Wright. “Coordinate descent algorithms”. In: *Mathematical Programming* 151.1 (2015), pp. 3–34.

[111] S.J. Wright. “Efficient convex optimization for linear MPC”. In: *Handbook of Model Predictive Control*. Springer, 2019, pp. 287–303.

[112] L. Wu. “An interpretative and adaptive MPC for nonlinear systems”. In: *arXiv preprint arXiv:2209.01513* (2022).

[113] L. Wu. “Equivalence of SS-based MPC and ARX-based MPC”. In: *arXiv preprint arXiv:2209.00107* (2022).

[114] L. Wu and A. Bemporad. “A construction-free coordinate-descent augmented-Lagrangian method for embedded linear MPC based on ARX models”. In: *arXiv preprint arXiv:2207.06098* (2022).

[115] L. Wu and A. Bemporad. “A Simple and Fast Coordinate-Descent Augmented-Lagrangian Solver for Model Predictive Control”. In: *arXiv preprint arXiv:2109.10205* (2021).

[116] L. Wu and M. Nauta. “A rapid-prototype MPC tool based on gPROMS platform”. In: *arXiv preprint arXiv:2209.00092* (2022).

[117] Z. Xianyi, W. Qian, and W. Saar. “OpenBLAS: An optimized BLAS library”. In: *Accedido: Agosto* (2016).

[118] A. Zanelli et al. “FORCES NLP: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs”. In: *International Journal of Control* 93.1 (2020), pp. 13–29.

[119] P. Zometa, M. Kögel, and R. Findeisen. “μAO-MPC: a free code generation tool for embedded real-time linear model predictive control”. In: *2013 American Control Conference*. IEEE. 2013, pp. 5320–5325.