Adaptive Model Predictive Control of Constrained Multiple-Input Multiple-Output Systems and Its Application to the Quad Tank System

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Automatic Control Laboratory

Adaptive Model Predictive Control of Constrained Multiple-Input Multiple-Output Systems and Its Application to the Quad Tank System

Master's Thesis

Işık İlber Sırmatel

Supervisor: Marko Tanaskovic

Professor: Prof. Dr. Manfred Morari

June 2, 2014

Abstract

Adaptive control involves adjusting the controller using data gathered in real-time, with the goal of controlling uncertain and/or time varying dynamic systems, which in practice invariably have constraints on the control inputs and usually also on the controlled variables. Although there is a well established theory of adaptive control, there are few results on constrained MIMO systems. To address this issue, an adaptive MPC algorithm for constrained linear MIMO systems has recently been developed, which integrates real-time set-membership identification (SMI) with constrained control. The identification step recursively identifies the set of all plant models consistent with initial information on the plant and input-output data collected in real-time. The controller then uses this set to guarantee satisfaction of input and output constraints for all the plant models inside it, and thus also for the true plant. The method is able to guarantee robust output constraint satisfaction, recursive feasibility, and offset-free reference tracking. It is also computationally tractable, requiring only solving standard convex optimization problems. Building on this structure, in this thesis we focus on methods to improve the set-membership identification step and extend it to deal with time-varying systems. For polytopic SMI, a method to evaluate the informativeness of measurements is developed to make careful use of the limited resource that is the number of faces of the polytope. Furthermore, a basic bounded complexity method is augmented with learning capability, which can tune the method to plant dynamics. Extensions that can handle time-varying systems are constructed, using ideas of model set inflation for slowly varying and center tracking for rapidly varying systems. Zonotopic SMI methods are investigated as a computationally advantageous alternative to polytopes and zonotopic counterparts of the polytopic methods to deal with time-varying systems are proposed. The developed algorithms are verified through simulations on a nonlinear quadruple-tank process model.

Contents

Nomenclature

Adaptive MPC of Constrained MIMO Systems

$\Delta u(t)$	Change in control input
ϵ_d	Magnitude bound on output disturbance
ϵ_v	Magnitude bound on measurement noise
$\hat{d}(t)$	Prediction error
$\hat{e}(t)$	Tracking error
Λ	Auxiliary decision variables
$\|.\|_X^2$	Quadratic form of a vector weighted with X
$\mathbb{R}$	Set of real numbers
$\mathbb{R}^n$	Set of real n-vectors
$\mathbb{R}^{m\times n}$	Set of real $m \times n$ matrices
$\mathbb{S}_+^n$	Set of symmetric positive semidefinite $n \times n$ matrices
$\mathbb{S}_{++}^n$	Set of symmetric positive definite $n \times n$ matrices
$\mathcal{F}(t)$	Model set containing model set polytopes $\mathcal{F}_j(t),\ j = 1, \ldots, n_y$
$\mathcal{F}_j(t)$	Model set polytope for plant output j
$\mathcal{M}_j(t)$	Measurement strip for plant output j
μ_{ji}	Parameter defining bounds on FIR coefficient for the input-output pair i-j
ρ_{ji}	Parameter defining bounds on FIR coefficient for the input-output pair i-j
$\tilde{y}$	Plant output measurement
$\varphi(t)$	Regressor vector
$A_j(t)$	Matrix defining the MSP $\mathcal{F}_j(t)$
$b_j(t)$	Vector defining the MSP $\mathcal{F}_j(t)$
C	Matrix defining constraints on control input
$d(t)$	Output disturbance

E Matrix defining constraints on plant output

f Vector defining constraints on change in control input

g Vector defining constraints on control input

H FIR model

h FIR coefficient

$H_c(t)$ Nominal model of model set $\mathcal{F}(t)$

$H_j(t)$ FIR coefficients vector j (element of $\mathcal{F}_j(t)$)

$H_{c,j}(t)$ Nominal model of model set polytope $\mathcal{F}_j(t)$

$J(t)$ Cost function

L Matrix defining constraints on change in control input

L_{ji} Parameter defining bounds on FIR coefficient for the input-output pair i-j

m Length of FIR model

n_i Number of constraints on control input

n_o Number of constraints on plant output

n_u Number of control inputs

n_y Number of plant outputs

$n_{\Delta u}$ Number of constraints on change in control input

p Vector defining constraints on plant output

Q Weighting matrix on tracking error

R Weighting matrix on change in control input

$r_{A_j}(t)$ Number of faces in $A_j(t)H_j \leq b_j(t)$ part of the MSP $\mathcal{F}_j(t)$

S Weighting matrix on control input

$U(t)$ Future control inputs vector

$u(t)$ Control input

$v(t)$ Measurement noise

W Matrix defining future regressor vectors

w Matrix defining future regressor vectors

$y(t)$ Plant output

$y_{\text{des}}(t)$ Desired output reference

Z Matrix defining future regressor vectors

z Vector defining future regressor vectors

Improvements on Real-Time Polytopic Set-Membership Identification

Γ_{A_j} Face filtering parameter for $A_j(t)H_j \leq b_j(t)$ part of $\mathcal{F}_j(t)$

$\Gamma^{\mathrm{a}}_{A_j}$ Parameter defining amplitude for $A_j(t)H_j \leq b_j(t)$ part of $\mathcal{F}_j(t)$ in self-tuning FFPU

$\Gamma^{\mathrm{m}}_{A_j}$ Parameter defining medium value for $A_j(t)H_j \leq b_j(t)$ part of $\mathcal{F}_j(t)$ in self-tuning FFPU

Γ_{D_j} Face filtering parameter for $D_j(t)H_j \leq b^D_j(t)$ part of $\mathcal{F}_j(t)$

$\hat{D}_j$ Predefined set of direction vectors for plant output j

κ_{l_j} Cut ratio of lower face of the measurement strip for plant output j

κ_{u_j} Cut ratio of upper face of the measurement strip for plant output j

$\mathcal{M}_{\mathrm{l}_j}(t)$ Lower face of measurement strip for plant output j

$\mathcal{M}_{\mathrm{u}_j}(t)$ Upper face of measurement strip for plant output j

$\mathcal{S}^{\mathrm{P}}_j(t)$ Polytope support strip for plant output j

$b^D_j(t)$ Vector defining $D_j(t)H_j \leq b^D_j(t)$ part of $\mathcal{F}_j(t)$ in learning BCPU

$D_j(t)$ Matrix defining $D_j(t)H_j \leq b^D_j(t)$ part of $\mathcal{F}_j(t)$ in learning BCPU

d_{l_j} Shortest distance from the lower face of measurement strip to the upper vertex for plant output j

d_{PSS_j} Width of the polytope support strip for plant output j

d_{u_j} Shortest distance from the upper face of measurement strip to the lower vertex for plant output j

$M_{\hat{D}_j}$ Number of predefined direction vectors for plant output j

M_{A_j} Face number limit on $A_j(t)H_j \leq b_j(t)$ part of $\mathcal{F}_j(t)$

M_{D_j} Face number limit on $D_j(t)H_j \leq b^D_j(t)$ part of $\mathcal{F}_j(t)$

$p^{\mathrm{P}}_{\mathrm{l}_j}$ Lower vertex at the intersection of polytope support strip and model set polytope for plant output j

$p^{\mathrm{P}}_{\mathrm{u}_j}$ Upper vertex at the intersection of polytope support strip and model set polytope for plant output j

$q^{\mathrm{P}}_{\mathrm{l}_j}$ Lower bound defining polytope support strip for plant output j

$q^{\mathrm{P}}_{\mathrm{u}_j}$ Upper bound defining polytope support strip for plant output j

$r_{D_j}(t)$ Number of faces in $D_j(t)H_j \leq b^D_j(t)$ part of $\mathcal{F}_j(t)$

Extensions for Time-varying Systems

$\bar{\mathcal{F}}_j(t)$ Translated version of model set polytope $\mathcal{F}_j(t)$

$\bar{H}_{c,j}(t)$ Translated version of nominal model $H_{c,j}(t)$

$\hat{\mathcal{F}}_j(t)$ Inflated version of model set polytope $\mathcal{F}_j(t)$

$\mathcal{M}_j^{\mathrm{m}}(t)$ Median hyperplane of $\mathcal{M}_j(t)$

$\mathcal{T}_j^{\mathrm{P}}(t)$ Polytope tight strip of model set polytope $\mathcal{F}_j(t-1)$

Ω_{DPI} Default polytopic inflation factor

Ω_{PE} Polytopic explosion factor

Ω_{PSI} Polytopic set inflation factor

Ω_{TPE} Trailing polytopic explosion factor

Ξ Slack variables vector for soft output constraints

$A_j^{\mathcal{M}}(t)$ Matrix defining faces coming from measurement strip in $\mathcal{F}_j(t)$

$b_j^{\epsilon}(t)$ Vector containing magnitude bounds on disturbance and noise defining faces coming from measurement strips in $\mathcal{F}_j(t)$

$b_j^{\tilde{y}}(t)$ Vector containing measurements defining faces coming from measurement strips in $\mathcal{F}_j(t)$

N_{PCT} Polytopic center tracking horizon

P Weighting matrix on slack variables for soft output constraints

$V_{\mathrm{T}}^{\mathrm{P}}(t)$ Translation vector in polytopic center tracking

Zonotopic Methods

$\bar{\mathcal{G}}_j(t)$ Translated version of model set zonotope $\mathcal{G}_j(t)$

$\bar{T}(k)$ Matrix defining the zonotope shape for family of overbounding zonotopes in improved zonotopic update

$\bar{v}(k)$ Vector defining the zonotope center for family of overbounding zonotopes in improved zonotopic update

η Vector defining zonotope tight strip

$\hat{\mathcal{G}}_j(t)$ Inflated version of model set zonotope $\mathcal{G}_j(t)$

$\mathcal{G}_j(t)$ Model set zonotope for plant output j

$\mathcal{S}^{\mathrm{Z}}(t)$ Zonotope support strip

$\mathcal{T}^{\mathrm{Z}}(t)$ Zonotope tight strip

Ω_{DZI} Default zonotopic inflation factor

Ω_{TZE} Trailing zonotopic explosion factor

Ω_{ZE} Zonotopic explosion factor

Ω_{ZSI} Zonotopic set inflation factor

σ Bound defining zonotope tight strip

$\mathbf{B}^r$ r-dimensional hypercube

L Matrix defining the shape of the minimal volume orthotope in improved zonotopic update

N_{ZCT} Zonotopic center tracking horizon

q Vector defining the center of the minimal volume orthotope in improved zonotopic update

$q_{\mathrm{l}}^{\mathrm{Z}}$ Lower bound defining zonotope support strip

$q_{\mathrm{u}}^{\mathrm{Z}}$ Upper bound defining zonotope support strip

r Zonotope order

$T(k)$ Matrix defining the zonotope shape for family of overbounding zonotopes in basic zonotopic update

$v(k)$ Vector defining the zonotope center for family of overbounding zonotopes in basic zonotopic update

$V_{\mathrm{T}}^{\mathrm{Z}}(t)$ Translation vector in zonotopic center tracking

w Vector defining the zonotope center

Z Matrix defining the zonotope shape

Simulation Results

$\bar{h}_i$ Liquid level in tank i at steady state

$\bar{v}_i$ Voltage applied to pump i at steady state

γ_i Valve coefficient of valve i

g Acceleration of gravity

$h_i(t)$ Liquid level in tank i

k_c Sensor constant

k_i Pump flow constant of pump i

$q_i(t)$ Liquid flow to tank i

S_i Cross-section of tank i

s_i Cross-section of the outlet hole of tank i

T_i Time constant of tank i

$u_i(t)$ Deviation of voltage applied to pump i from steady state

$v_i(t)$ Voltage applied to pump i

$x_i(t)$ Deviation of liquid level in tank i from steady state

$y_i(t)$ Deviation of sensor output reading from liquid level in tank i from steady state

List of Figures

List of Algorithms

List of Tables

List of Abbreviations

MPC Model Predictive Control

SMI Set-Membership Identification

QTP Quadruple-Tank Process

LTI Linear Time Invariant

FHOCP Finite Horizon Optimal Control Problem

MIMO Multiple-Input Multiple-Output

FIR Finite Impulse Response

LP Linear Program

QP Quadratic Program

MS Measurement Strip

PU Polytopic Update

ZU Zonotopic Update

FFPU Face Filtering Polytopic Update

BCPU Bounded Complexity Polytopic Update

GL Gradual Learning

BL Batch Learning

MSP Model Set Polytope

MSZ Model Set Zonotope

PSS Polytope Support Strip

ZSS Zonotope Support Strip

PTS Polytope Tight Strip

ZTS Zonotope Tight Strip

PSI Polytopic Set Inflation

ZSI Zonotopic Set Inflation

PCT Polytopic Center Tracking

ZCT Zonotopic Center Tracking

ISE Integral Square Error

IOUI Integral Output Uncertainty Interval

TCT Total Computation Time

TNF Total Number of Faces

TZO Total Zonotope Order

Chapter 1

Introduction

1.1 Background

Adaptive control involves adjusting the controller during closed-loop operation using data on system inputs and outputs gathered in real-time to control uncertain and/or time varying dynamic systems. Plants in practice inevitably have constraints on the control inputs, since every actuator has its physical limits, whereas usually the controlled variables are also constrained. These point to the direction that adaptive control methods that can deal with systems subject to constraints are potentially useful. However, even though a well established theory of adaptive control is present for quite some time [1], there are few results on the adaptive control of constrained Multiple-Input Multiple-Output (MIMO) systems [2]. To address this issue, an adaptive Model Predictive Control (MPC) algorithm handling constrained linear MIMO systems has recently been developed [3–6], which augments a robust MPC controller with real-time Set-Membership Identification (SMI) capability. During the SMI step, the set of all plant models, consistent with initial information on the plant and input-output data collected in real-time, is identified in a recursive manner, and a nominal model from within this model set is selected. The MPC controller then solves a Quadratic Program (QP) in receding horizon fashion, which minimizes the tracking error for the nominal model while enforcing input and output constraints for all models inside the model set. Since it is assumed that the true plant belongs to the initial model set defined prior to closed-loop operation, and this set is updated recursively using measurements and past inputs with assumed disturbance and noise bounds, the model set always contains the model of the true plant. Thus, the controller can guarantee input and output constraint satisfaction for the true plant with the adaptive MPC method. It can also guarantee offset-free reference tracking against constant output disturbances, and recursive feasibility for non-expanding model sets. Furthermore, the algorithm is computationally tractable, requiring solution of only standard convex optimization problems. Thus the method has high potential of practical use for the adaptive control of constrained systems.

1.2 Motivation

Although a powerful method for the adaptive control of constrained MIMO systems, the adaptive MPC algorithm has some drawbacks:

1. Large computation times limit its use to plants with large sampling times. This problem is especially pronounced in cases where large numbers are chosen as the

predefined polytope face number limits of the polytopic SMI engine, since the size of the optimization problems being solved by the controller in receding horizon is directly related to the number of faces of the model set polytopes.

2. There exists no capability of handling time-varying systems.

Motivated by these two main points, the thesis focuses on the improvement of the SMI engine and extending its use to time-varying systems. For the first point, existing polytopic SMI methods are improved via approaches that can use the limited memory for polytope faces wisely, and thus increase the quality of the identified model sets. Furthermore, zonotopic SMI methods are investigated as a computationally advantageous alternative to polytopic methods. For the second point, extensions are developed for both polytopic and zonotopic SMI methods, which can handle plants with slowly and rapidly varying dynamics.

1.3 Methods

Polytopic SMI is improved via a method for evaluating the information content in model set updates, which can improve identification by strategically spending the limited memory available for storing faces with high information content. The second improvement is the learning approach to bounded complexity methods, which can tune these in real-time to plant dynamics, resulting in higher performance in bounded complexity style polytopic SMI. To handle time-varying systems, polytopic method extensions are constructed. These employ ideas of set inflation for countering slow variations in plant dynamics, whereas the center tracking approach is for dealing with abrupt changes. In addition, zonotopic methods that can keep model set complexity constant are investigated with the intention to yield savings in computational effort, which have the potential to extend the application of the adaptive MPC algorithm to systems with smaller sampling times. Finally, zonotopic counterparts of the methods for handling time-varying system are developed, which provide the zonotopic approach with the capability to deal with both slowly and rapidly varying dynamics. Performance of the developed algorithms are verified through simulation experiments and sensitivity analyses.

1.4 Structure

The adaptive MPC algorithm of [3] is studied in detail in Chapter 2, where the problem formulation and the structuring of the optimization problems to be solved in receding horizon are given. Chapter 3 is devoted to developing the improvements of polytopic SMI methods, namely the face filtering method and learning bounded complexity methods. Polytopic extensions for dealing with time-varying systems are presented in Chapter 4, which contains the set inflation and center tracking methods. Zonotopic methods are studied in Chapter 5, which explores two recent methods in literature. This chapter also provides zonotopic counterparts of the polytopic methods for time-varying systems. Chapter 6 first presents mathematical modeling of the Quadruple-Tank Process (QTP), and then shows simulation studies using a nonlinear QTP model that verify the developed real-time SMI methods. Chapter 7 concludes with a summary of the contributions and some thoughts on possible future work.

Chapter 2

Adaptive MPC of Constrained MIMO Systems

The adaptive MPC method for constrained linear MIMO systems is presented in this chapter. Section 2.1 states the considered control problem, whereas Section 2.2 describes the adaptive MPC algorithm proposed as its solution. Sections 2.3 and 2.4 present the blocks of the adaptive MPC algorithm, namely the real-time SMI and the robust MPC blocks. The chapter is based on the work of Tanaskovic and coworkers in [3].

2.1 Problem Statement

A Linear Time Invariant (LTI) MIMO, discrete time, strictly proper system with n_u inputs and n_y outputs is considered. It is known that the system is stable, but its dynamics are unknown. The control inputs, plant outputs, output disturbances, and measurement noise vectors at time $t \in \mathbb{Z}$ are

$$\begin{aligned} u(t) &= [u_1(t),\ \ldots,\ u_{n_u}(t)]^T \\ y(t) &= [y_1(t),\ \ldots,\ y_{n_y}(t)]^T \\ d(t) &= [d_1(t),\ \ldots,\ d_{n_y}(t)]^T \\ v(t) &= [v_1(t),\ \ldots,\ v_{n_y}(t)]^T \end{aligned} \tag{2.1}$$

where $u_i(t) \in \mathbb{R}$ is the control input i, with $i = 1, \ldots, n_u$, $y_j(t) \in \mathbb{R}$ is the plant output j, with $j = 1, \ldots, n_y$, whereas $d_j(t) \in \mathbb{R}$ and $v_j(t) \in \mathbb{R}$ are the disturbance and noise j, with $j = 1, \ldots, n_y$, affecting the plant output $y_j(t)$, respectively.
An infinite impulse response can in general describe the dynamics from each input to each output, but a Finite Impulse Response (FIR) model is used here to obtain a tractable process model. The length of the model is denoted by m, leading to m FIR coefficients $h_{ji}(k)$, with $k = 1, \ldots, m$, to describe the relation of input i and output j. With this setting the plant output $y_j(t)$ can be given as

$$\begin{aligned} y_j(t) &= \sum_{i=1}^{n_u} \sum_{k=1}^{m} u_i(t-k) h_{ji}(k) + d_j(t) \\ &= H_j^T\ \varphi(t) + d_j(t), \quad j = 1, \ldots, n_y, \end{aligned} \tag{2.2}$$

where $\varphi(t) \in \mathbb{R}^{n_u m}$ is the regressor vector defined as

$$\varphi(t) \triangleq [u_1(t-1), \ldots, u_1(t-m), \ldots, u_{n_u}(t-1), \ldots, u_{n_u}(t-m)]^T \tag{2.3}$$

and the vector $H_j \in \mathbb{R}^{n_u m}$, defined as

$$H_j \triangleq [h_{j1}(1), \ldots, h_{j1}(m), \ldots, h_{jn_u}(1), \ldots, h_{jn_u}(m)]^T, \quad j = 1, \ldots, n_y, \tag{2.4}$$

is the FIR coefficients vector describing the influence of the control inputs $u_i(t), i = 1, \ldots, n_u$ on the j-th plant output $y_j(t)$. The vectors H_j constitute the matrix $H \in \mathbb{R}^{n_y \times n_u m}$, i.e., the FIR model:

$$H \triangleq [H_1, \ \ldots, \ H_{n_y}]^T, \tag{2.5}$$

which describes the relation of $\varphi(t)$ with plant output, corrupted by output disturbance $d(t)$:

$$y(t) = H\varphi(t) + d(t). \tag{2.6}$$

The vector of measurement on plant output, corrupted by measurement noise $v(t)$, can be written as:

$$\tilde{y}(t) = y(t) + v(t). \tag{2.7}$$

Assumption 1. *(Prior assumption on disturbance and noise) Disturbance $d_j(t)$ and noise $v_j(t)$ are bounded as follows:*

$$|d_j(t)| \leq \epsilon_{d_j}, \ |v_j(t)| \leq \epsilon_{v_j}, \ \forall t \in \mathbb{Z}, \ \forall j = 1, \ldots, n_y, \tag{2.8}$$

where the bounds ϵ_{d_j} and ϵ_{v_j} are positive scalars.

In vector notation, the disturbance and noise bounds can be written as $\epsilon_d = [\epsilon_{d_1}, \ldots, \epsilon_{d_{n_y}}]^T$ and $\epsilon_v = [\epsilon_{v_1}, \ldots, \epsilon_{v_{n_y}}]^T$.

Assumption 2. *(Prior assumption on the system) The actual plant is a member of the initial model set $\mathcal{F}(0)$, with*

$$\mathcal{F}(0) \triangleq \{H \in \mathbb{R}^{n_y \times n_u m} \ : \ A_j(0)H_j \leq b_j(0), \ j = 1, \ldots, n_y\}, \tag{2.9}$$

where the inequalities are element-wise and each expression $A_j(0)H_j \leq b_j(0)$ defines a polytope in the space of FIR coefficient vectors H_j, with $A_j(0)$ denoting a matrix in $\mathbb{R}^{r_{A_j}(0) \times n_u m}$ and $b_j(0)$ denoting a vector in $\mathbb{R}^{r_{A_j}(0)}$, where $r_{A_j}(0)$ denotes the number of faces forming the polytope $A_j(0)H_j \leq b_j(0)$.

Due to Assumption 2, the initial FIR coefficient vectors $H_j, \ j = 1, \ldots, n_y$ belong to polytopic sets. $\mathcal{F}(0)$ denotes the model set at time $t = 0$, which corresponds to the a-priori available information on the system; the Model Set Polytopes (MSPs) $\mathcal{F}_j(t), \ j = 1, \ldots, n_y$ at time $t = 0$ have to be initialized according to this information. The assumption that the system is stable can be used to construct $\mathcal{F}(0)$ such that the FIR coefficients show an exponential decay and have constraints on magnitude, which can be done by constraining the FIR coefficients from the i-th input to the j-th output as follows

$$\begin{array}{lcll} |h_{ji}(k)| & \leq & L_{ji} & \text{if } k \in [1, \mu_{ji}] \\ |h_{ji}(k)| & \leq & L_{ji}\rho_{ji}^{k-\mu_{ji}} & \text{if } k \in [\mu_{ji}+1, m] \end{array} \tag{2.10}$$

where L_{ji}, ρ_{ji}, and μ_{ji} are design variables, with

$$\begin{array}{l} L_{ji} \in \mathbb{R}, \ L_{ji} \geq 0 \\ \rho_{ji} \in \mathbb{R}, \ \rho_{ji} \in (0, 1) \\ \mu_{ji} \in \mathbb{N}, \ \mu_{ji} \leq m. \end{array} \tag{2.11}$$

The control objectives are reference tracking and disturbance rejection, whilst also ensuring input and output constraint satisfaction, from time $t = 0$ to some finite time step T, with a possibly large time horizon ($T \gg m$). The control objective can be expressed through the following optimization problem:

$$\begin{aligned} \underset{\{u(t)\}_{t=0}^{T-1}}{\text{minimize}} \quad & \sum_{t=0}^{T} \Big(\|y(t) - y_{\text{des}}(t)\|_Q^2 + \|u(t)\|_S^2 + \|\Delta u(t)\|_R^2 \Big) \\ \text{subject to} \quad & Cu(t) \leq g \\ & L\Delta u(t) \leq f \\ & Ey(t) \leq p, \forall t \in [0, T], \end{aligned} \tag{2.12}$$

where $y_{\text{des}}(t) \in \mathbb{R}^{n_y}$ is the desired output reference, $\Delta u(t) = u(t) - u(t-1)$ is the control input rate of change, and $Q \in \mathbb{S}_+^{n_y}$, $S \in \mathbb{S}_{++}^{n_u}$, and $R \in \mathbb{S}_+^{n_u}$ are weighting matrices for tracking error, control input, and change in control input, respectively. The matrices $C \in \mathbb{R}^{n_i \times n_u}$, $L \in \mathbb{R}^{n_{\Delta u} \times n_u}$, $E \in \mathbb{R}^{n_o \times n_y}$, and the vectors $g \in \mathbb{R}^{n_i}$, $f \in \mathbb{R}^{n_{\Delta u}}$, and $p \in \mathbb{R}^{n_o}$, which appear in the element-wise inequalities, form convex sets defining the constraints, where n_i, $n_{\Delta u}$, and n_o are the number of constraints on control input, change in control input, and plant output, respectively. It is assumed that the set of constraints on change in control input contains the origin and that the set of constraints on control input is compact, which are usually valid in practice.

2.2 Adaptive MPC Algorithm

It is not possible to solve the optimal control problem (2.12) exactly, since the exact plant dynamics and the disturbances are unknown. A suboptimal solution has to be pursued, and the approach taken here is to deploy an adaptive MPC algorithm, which is essentially a robust MPC controller augmented with real-time SMI capability. The SMI engine uses measurements on the plant output to update the model set, possibly decreasing its size and thereby increasing the controller's knowledge of the plant dynamics, leading to improved control performance. The SMI engine also selects a nominal model from within the model set. The controller then solves a Finite Horizon Optimal Control Problem (FHOCP) in receding horizon fashion, which minimizes a weighted quadratic cost function including penalty on the nominal model's tracking error, while enforcing input and output constraints for all models inside the model set. Since the model set contains all plant models that are consistent with past measurements and Assumptions 1 and 2, the validity of which guarantees that the model set contains the dynamics of the true plant. The controller is thus able to guarantee output constraint satisfaction for the true plant. The procedure for the adaptive MPC algorithm is given in Algorithm 1.

Algorithm 1 Adaptive MPC algorithm

1) At time step t, use the regressor vector $\varphi(t)$ and the measurement of plant output $\tilde{y}(t)$ to update the model set;
2) Select a nominal model of the plant from within the model set;
3) Compute the future control inputs vector as the solution of a FHOCP, minimizing a weighted quadratic cost function (containing the tracking error of the nominal model) whilst guaranteeing robust satisfaction of input and output constraints for the model set;
4) Apply the first element of the future control inputs vector, set $t = t + 1$, go to 1).

Following sections describe the blocks constituting the adaptive MPC algorithm in detail.

2.3 Real-Time Polytopic Set-Membership Identification

Consider the sequence of plant input-output data collected from initial time $t = 0$ up to time t

$$\{\varphi(l),\ \tilde{y}(l)\}_{l=0}^{t}, \tag{2.13}$$

with $\varphi(l) \in \mathbb{R}^{n_u m}$ denoting the regressor vector containing control inputs applied from time $l-m$ up to time $l-1$, and $\tilde{y}(l) \in \mathbb{R}^{n_y}$ denoting the corresponding measured plant outputs. The model set $\mathcal{F}(t)$ at a given time step t is then defined as the set containing every matrix H that is consistent with Assumptions 1 and 2 and the input-output data (2.13) collected up to that same time step t:

$$\mathcal{F}(t) \triangleq \{H \in \mathcal{F}(0) : |\tilde{y}(l) - H\varphi(l)| \leq \epsilon_d + \epsilon_v,\ \forall l \in [0, t]\}. \tag{2.14}$$

The element-wise inequlities constituting $\mathcal{F}(t)$ as given in (2.14) result from the argument that it is not possible for the discrepancy between the measured plant output $\tilde{y}(t)$ and the predicted plant output $H\varphi(t)$ to exceed the sum of the magnitude bounds on disturbance and noise given in (2.8). The initial model set $\mathcal{F}(0)$ is defined by polytopic constraints on H_j (i.e., the rows of H) as given in (2.9), furthermore, the constraints in (2.14) specify linear constraints. As a result, the constraints on $H_j,\ j = 1, \ldots, n_y$ defining the model set $\mathcal{F}(t)$ are polytopic constraints. A set of non-redundant inequalities can be used to uniquely specify each model set polytope, thus the model set $\mathcal{F}(t)$ at a generic time step t can be written as

$$\mathcal{F}(t) \triangleq \{H \in \mathbb{R}^{n_y \times n_u m}\ :\ A_j(t)H_j \leq b_j(t),\ j = 1, \ldots, n_y\}, \tag{2.15}$$

where $A_j(t) \in \mathbb{R}^{r_{A_j}(t) \times n_u m}$ and $b_j(t) \in \mathbb{R}^{r_{A_j}(t)}$, with $r_{A_j}(t)$ denoting the number of non-redundant inequalities that constrain H_j (i.e., the number of faces in $A_j(t)H_j \leq b_j(t)$). At the core of real-time polytopic SMI is the operation of updating the matrices $A_j(t)$ and the vectors $b_j(t)$ $(j = 1, \ldots, n_y)$ at each step of the receding horizon, reflecting thus the newly obtained information with the measurement $\tilde{y}(t)$ and the regressor vector $\varphi(t)$ on the model set $\mathcal{F}(t)$. Consider the following MSPs

$$\mathcal{F}_j(t) \triangleq \{H_j \in \mathbb{R}^{n_u m}\ :\ A_j(t)H_j \leq b_j(t)\},\ j = 1, \ldots, n_y, \tag{2.16}$$

where each MSP $\mathcal{F}_j(t)$ is identified recursively in time by taking the intersection of the old MSP $\mathcal{F}_j(t-1)$ and the current Measurement Strip (MS) $\mathcal{M}_j(t)$:

$$\mathcal{F}_j(t) = \mathcal{F}_j(t-1) \cap \mathcal{M}_j(t), \tag{2.17}$$

with the MS $\mathcal{M}_j(t)$ is defined as

$$\mathcal{M}_j(t) = \{H_j \in \mathbb{R}^{n_u m}\ :\ |\varphi(t)^T H_j - \tilde{y}_j(t)| \leq \epsilon_{d_j} + \epsilon_{v_j}\}. \tag{2.18}$$

The above recursive polytopic model set update has the problem that $r_{A_j}(t)$ (i.e., the number of faces of the MSP $\mathcal{F}_j(t)$) can grow in general linearly with time, thus it can become arbitrarily large if no precautions are taken, leading to the memory needed to store $A_j(t)$ and $b_j(t)$ becoming impractical. Furthermore, size of the optimization problems the controller is solving in receding horizon depends chiefly on $r_{A_j}(t)$ (see (2.37)), which specifies a risk that these may become too large to solve within one sampling time. Thus, for a practicable adaptive MPC algorithm, we need to ensure that $r_{A_j}(t)$ is limited by a finite number. Following subsections present two existing Polytopic Update (PU) methods that are capable of limiting $r_{A_j}(t)$.

2.3.1 Polytopic Update

PU algorithm is the simplest possible update method of polytopic real-time SMI. It first checks the number of faces of the old MSP $\mathcal{F}_j(t-1)$ (i.e., $r_{A_j}(t-1)$); if $r_{A_j}(t-1)$ is less than the predefined face number limit M_{A_j} minus 1 (i.e., if there is space for both faces of $\mathcal{M}_j(t)$), both faces are added to the MSP, i.e., the following PU takes place:

$$\mathcal{F}_j(t) = \mathcal{F}_j(t-1) \cap \mathcal{M}_j(t). \tag{2.19}$$

If there is no space in the MSP, no update is carried out and the polytope remains the same, i.e., $\mathcal{F}_j(t) = \mathcal{F}_j(t-1)$. The PU procedure is given in Algorithm 2.

Algorithm 2 Polytopic update

0) Repeat for each output j, $j = 1, \ldots, n_y$.
1) At time step t, use the regressor vector $\varphi(t)$ and the measurement $\tilde{y}_j(t)$ to build the MS $\mathcal{M}_j(t) = \{H_j \in \mathbb{R}^{n_u m} \ : \ |\varphi(t)^T H_j - \tilde{y}_j(t)| \leq \epsilon_{d_j} + \epsilon_{v_j}\}$;
2) If $r_{A_j}(t-1) < M_{A_j}$ update $\mathcal{F}_j(t-1)$ to $\mathcal{F}_j(t)$ with $\mathcal{M}_j(t)$ by adding both faces using (2.19);
3) Remove redundant faces from $\mathcal{F}_j(t)$ to obtain $A_j(t)$ and $b_j(t)$.

Removing redundant faces can be done using the procedure given in [7], which involves solving a Linear Program (LP) for each face to be checked for redundancy. The details of the procedure are described in Appendix A.
Algorithm 2 guarantees that the MSPs are non-expanding, i.e., $\mathcal{F}_j(t+1) \subseteq \mathcal{F}_j(t)$, $j = 1, \ldots, n_y$, which is necessary for obtaining recursive feasibility and output constraint satisfaction, as explained in §4 of [3] in the proof for Theorem 1. A critical weakness of the PU method is that, if the face number limit M_{A_j} is not enforced as in step 2) of Algorithm 2, the number of faces of $\mathcal{F}_j(t)$ grows in general linearly with time and thus can become arbitrarily large [3]. This will render the memory requirements for storing $A_j(t)$ and $b_j(t)$ impractical and also the optimization problems being solved online can become intractable, leading the adaptive MPC algorithm to becoming unusable. Enforcing the limit M_{A_j} is thus absolutely necessary but still disadvantageous since identification has to stop completely when the face number limit M_{A_j} is reached. Another serious limitation of the algorithm is that it blindly fills up precious space in the polytope without evaluating the information content of the MSs. This leads to uninformative faces being added to the MSP and filling up space in the earlier periods of closed-loop operation. This could, later on during operation, result in substantial degradations in control performance as the SMI can no longer react to faces with high information content coming from the dynamics by updating the MSPs, since there is no space left to add more faces.

2.3.2 Bounded Complexity Polytopic Update

Bounded Complexity Polytopic Update (BCPU) method, presented in [3], and inspired by the facet-direction-limited polyhedron updating method proposed by Veres and coworkers in [8], specifies a way to overcome the limitation of the PU about not being able to continue identification for the MSP $\mathcal{F}_j(t)$ after face number limit M_{A_j} is reached. In this method, if the number of faces of the MSP is less than M_{A_j}, the MSP is updated as given in (2.19). When the number of faces reaches M_{A_j}, a different style of polytopic update is conducted, using directions that belong to a certain predefined set $\hat{D}_j$ instead of the direction specified by the regressor vector $\varphi(t)$. If the number of directions in $\hat{D}_j$ is a finite

number $M_{\hat{D}_j}$, then the number of faces of the MSP will be limited and cannot increase beyond $M_{A_j} + M_{\hat{D}_j}$.
For the method, a set of $n_u m$-dimensional vectors with the same length, denoted as the set $\hat{D}_j$, has to be defined a priori and will strongly effect the shape of the polytope. A reasonable way to choose the vectors in $\hat{D}_j$ is to take a set of vectors regularly distributed on the unit circle [9].
The BCPU involves the following intersection for updating the MSP $\mathcal{F}_j(t)$:

$$\begin{aligned}\mathcal{F}_j(t) =& \mathcal{F}_j(t-1) \\ &\cap \{H_j \in \mathbb{R}^{n_u m} : \varphi^+(t)^T H_j \leq \tilde{y}_j(t) + \delta_j^+(t)\} \\ &\cap \{H_j \in \mathbb{R}^{n_u m} : \varphi^-(t)^T H_j \leq -\tilde{y}_j(t) + \delta_j^-(t)\},\end{aligned} \tag{2.20}$$

where the vectors $\varphi^+(t)$ and $\varphi^-(t)$ are those members of $\hat{D}_j$ that are the most closely aligned with the vectors $\varphi(t)$ and $-\varphi(t)$, in the inner product sense:

$$\begin{aligned}\varphi^+(t) &= \arg\max_{v \in \hat{D}_j} \varphi(t)^T v \\ \varphi^-(t) &= \arg\max_{v \in \hat{D}_j} -\varphi(t)^T v,\end{aligned} \tag{2.21}$$

and the scalars $\delta_j^+(t)$ and $\delta_j^-(t)$ are calculated so as to ensure that the polytope obtained with the update (2.20) overbounds the one that would be obtained if the update was done using (2.19) instead. To calculate these scalars the following LP has to be solved:

$$\begin{aligned}\delta_j^+(t)/\delta_j^-(t) = \underset{\theta}{\text{maximize}} \quad & \varphi^+(t)^T\theta - \tilde{y}_j(t) / \varphi^-(t)^T\theta + \tilde{y}_j(t) \\ \text{subject to} \quad & A_j(t-1)\theta \leq b_j(t-1) \\ & \varphi^T\theta \leq \tilde{y}_j(t) + \epsilon_{d_j} + \epsilon_{v_j} \\ & -\varphi^T\theta \leq -\tilde{y}_j(t) + \epsilon_{d_j} + \epsilon_{v_j}.\end{aligned} \tag{2.22}$$

The BCPU method described here, taken from [3], uses a set of non-redundant inequalities for polytopic update, and thus differs from the similar method of [8] which uses a vertex list updating. Using inequalities is important for yielding lower memory requirements in case of high dimensional polytopes, and is thus a better choice for the adaptive MPC algorithm [3]. The procedure for the BCPU method is given in Algorithm 3.

Algorithm 3 Bounded complexity polytopic update

0) Repeat for each output j, $j = 1, \ldots, n_y$.
1) At time step t, use the regressor vector $\varphi(t)$ and the measurement $\tilde{y}_j(t)$ to build the MS $\mathcal{M}_j(t) = \{H_j \in \mathbb{R}^{n_u m} \ : \ |\varphi(t)^T H_j - \tilde{y}_j(t)| \leq \epsilon_{d_j} + \epsilon_{v_j}\}$;
2) If $r_{A_j}(t-1) < M_{A_j} - 1$ update $\mathcal{F}_j(t-1)$ to $\mathcal{F}_j(t)$ with $\mathcal{M}_j(t)$ using (2.19) by adding both faces and go to 4), otherwise go to 3);
3) Using the predefined set of direction vectors $\hat{D}_j$, obtain $\varphi^+(t)$ and $\varphi^-(t)$ as in (2.21), calculate $\delta_j^+(t)$ and $\delta_j^-(t)$ via (2.22), update $\mathcal{F}_j(t-1)$ to $\mathcal{F}_j(t)$ using (2.20);
4) Remove redundant faces from $\mathcal{F}_j(t)$ to obtain $A_j(t)$ and $b_j(t)$.

Similar to Algorithm 2, Algorithm 3 can also guarantee that the MSPs are non-expanding, i.e., $\mathcal{F}_j(t+1) \subseteq \mathcal{F}_j(t)$, $j = 1, \ldots, n_y$, thus recursive feasibility and output constraint satisfaction are guaranteed. The MSP obtained by BCPU is an approximation which

overbounds the one that would be obtained using the PU, introducing some conservatism. This conservatism can be decreased via increasing the face number limits M_{A_j} and $M_{\hat{D}_j}$ for the price of increased complexity and thus computational effort [3]. Compared to the PU algorithm 2, where identification stops when the number of faces reaches M_{A_j}, the BCPU method specifies an improvement since identification can be conducted forever using (2.20). A disadvantage of the BCPU method is that the set $\hat{D}_j$ is defined a priori without consideration of the plant dynamics: These predefined vectors (i.e., elements of $\hat{D}_j$) may not be closely resembling the regressor vectors $\varphi(t)$ that occur during closed-loop operation (which are thus related to the plant dynamics), thus SMI conducted using the BCPU method with $\hat{D}_j$ may not result in the best bounded complexity style identification opportunities, leading to MSP updates with little to no decrease in polytope size, hence also no improvement on control performance.

2.4 Robust Model Predictive Controller

Consider the possible future controls $u(k|t)$:

$$u(k|t),\ k \in [t, t+N-1],\ N \geq m \tag{2.23}$$

where N is the prediction horizon and $k|t$ implies the prediction at time step $k \geq t$ given the information at the current time step t. Future control inputs are collected in the vector U, which is defined as

$$U \triangleq \begin{bmatrix} u(t|t) \\ \vdots \\ u(t+N-1|t) \end{bmatrix} \tag{2.24}$$

Future changes in control inputs $\Delta u(k|t)$ are defined as

$$\Delta u(k|t) = \begin{cases} u(t|t) - u(t-1) & \text{if } k = t \\ u(k|t) - u(k-1|t) & \text{if } t+1 \leq k \leq t+N-1, \end{cases} \tag{2.25}$$

whereas future regressor vectors $\varphi(k|t) \in \mathbb{R}^{n_u m}, k \in [t+1, t+N]$ are defined as

$$\varphi(k|t) = \begin{cases} W\varphi(t) + Zu(t|t) & \text{if } k = t+1 \\ W\varphi(k-1|t) + Zu(k-1|t) & \text{if } t+2 \leq k \leq t+N, \end{cases} \tag{2.26}$$

where $W \in \mathbb{R}^{n_u m \times n_u m}$ and $Z \in \mathbb{R}^{n_u m \times n_u}$ are matrices defined as

$$W \triangleq \begin{bmatrix} w & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & w & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & w \end{bmatrix} \quad Z \triangleq \begin{bmatrix} z & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & z & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & z \end{bmatrix}, \tag{2.27}$$

with $w \in \mathbb{R}^{m \times m}$ and $z \in \mathbb{R}^m$ defined as follows

$$w \triangleq \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix} \quad z \triangleq \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{2.28}$$

Furthermore, the current prediction error $\hat{d}(t) \in \mathbb{R}^{n_y}$ is defined as the difference between the measured plant output $\tilde{y}(t)$ and the plant output predicted by the nominal model $H_c(t)$:

$$\hat{d}(t) \triangleq \tilde{y}(t) - H_c(t)\varphi(t). \tag{2.29}$$

Consider finally the cost function J:

$$J(U, \tilde{y}(t), \varphi(t)) \triangleq \sum_{k=t}^{t+N-1} \left(\|\hat{y}(k+1|t) - y_{\text{des}}(k+1|t)\|_Q^2 + \|u(k|t)\|_S^2 + \|\Delta u(k|t)\|_R^2 \right), \tag{2.30}$$

with $\hat{y}(k+1|t)$ defined as

$$\hat{y}(k+1|t) \triangleq H_c(t)\varphi(k+1|t) + \hat{d}(t). \tag{2.31}$$

Since $\tilde{y}(t)$ is the measured plant output, the regressor vector $\varphi(t)$ contains past controls, and $y_{\text{des}}(k|t), k \in [t+1, t+N]$ are the predicted desired outputs, these values in (2.30) are known.
Prediction error $\hat{d}(t)$ is included in the cost function with the purpose of obtaining offset free reference tracking under certain conditions (for details see §4 in [3]). Furthermore, for the case of the nominal model $H_c(t)$ being the same as the real plant, measurement noise $v(t)$ begin zero, output disturbance $d(t)$ being constant, and $N = T$, the cost function (2.30) would be equivalent to the control objective cost function given in (2.12).
With the following set of inequalities constraints on controls and changes in controls are enforced:

$$\begin{array}{rcl} Cu(k|t) & \leq & d \\ L\Delta u(k|t) & \leq & f \end{array} \quad \forall k \in [t, t+N-1]. \tag{2.32}$$

Robust output constraint satisfaction involves enforcing output constraints for all plant models belonging to the model set $\mathcal{F}(t)$ under all disturbance realizations:

$$EH\varphi(k|t) + Ed \leq p, \qquad \begin{array}{r} \forall H \in \mathcal{F}(t),\ \forall d : -\epsilon_d \leq d \leq \epsilon_d, \\ k \in [t+1, t+N], \end{array} \tag{2.33}$$

which are satisfied for any disturbance realization in case the following inequalities are satisfied:

$$EH\varphi(k|t) + \bar{d} \leq p,\ \forall H \in \mathcal{F}(t),\ \forall k \in [t+1, t+N], \tag{2.34}$$

where

$$\bar{d} \triangleq \begin{bmatrix} \bar{d}_1 \\ \vdots \\ \bar{d}_{n_o} \end{bmatrix} \tag{2.35}$$

with $\bar{d}_l \in \mathbb{R}$ given as

$$\bar{d}_l \triangleq \sum_{j=1}^{n_y} |e_{lj}| \epsilon_{d_j},\ l = 1, \ldots, n_o, \tag{2.36}$$

where e_{lj} denotes the element of the matrix E that is on the l^{th} row and j^{th}. An optimization problem having the constraints in (2.34) would be an infinite dimensional one, finding the solution of which is very hard in general. The lemma that follows

reformulates (2.34) through linear constraints. Consider for this purpose first the auxiliary decision variables $\Lambda \in \mathbb{R}^{n_o N r_A(t)}$:

$$\Lambda \triangleq \begin{bmatrix} \Lambda_1 \\ \vdots \\ \Lambda_{n_o} \end{bmatrix} \tag{2.37}$$

where

$$\Lambda_l \triangleq \begin{bmatrix} \lambda_l(t+1|t) \\ \vdots \\ \lambda_l(t+N|t) \end{bmatrix}, \; l = 1, \ldots, n_o, \tag{2.38}$$

and for each $k = t+1, \ldots, t+N$, $\lambda_l(k|t) \in \mathbb{R}^{r_A(t)}$, with $r_A(t) \triangleq \sum_{j=1}^{n_y} r_{A_j}(t)$.

Lemma 1. The constraints in (2.34) are satisfied if and only if there exists U and Λ such that the following inequalities are feasible

$$\left.\begin{aligned} A(t)^T \lambda_l(k|t) &= \begin{bmatrix} e_{l1}\varphi(k|t) \\ \vdots \\ e_{ln_y}\varphi(k|t) \end{bmatrix} \\ b(t)^T \lambda_l(k|t) &\le p_l - \bar{d}_l \\ \lambda_l(k|t) &\ge \mathbf{0} \end{aligned}\right\} \begin{array}{c} \forall l = 1, \ldots, n_o \\ \forall k \in [t+1, t+N] \end{array} \tag{2.39}$$

where $A(t)$ and $b(t)$ are defined as

$$A(t) \triangleq \begin{bmatrix} A_1(t) & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & A_2(t) & \ldots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \ldots & A_{n_y}(t) \end{bmatrix}, \quad b(t) \triangleq \begin{bmatrix} b_1(t) \\ \vdots \\ b_{n_y}(t) \end{bmatrix}, \tag{2.40}$$

with p_l denoting the l^{th} element of the vector p [3].

Furthermore, to be able to obtain recursive feasibility of input and output constraints, consider following additional constraint on the terminal stage

$$\varphi(t+N|t) = W\varphi(t+N|t) + Zu(t+N-1|t), \tag{2.41}$$

which implies that the terminal regressor vector is forced to correspond to a steady state (i.e., the controls are kept constant for the last m steps of the prediction).
The FHOCP to be solved in receding horizon at time step t as the third step of Algorithm 1 can finally be defined for fixed values of N, Q, S, and R:

$$\begin{aligned} \underset{U,\Lambda}{\text{minimize}} \quad & \sum_{k=t}^{t+N-1} \left(\|\hat{e}(k+1|t)\|_Q^2 + \|u(k|t)\|_S^2 + \|\Delta u(k|t)\|_R^2 \right) \\ \text{subject to} \quad & \left.\begin{aligned} Cu(k|t) &\le d \\ L\Delta u(k|t) &\le f \end{aligned}\right\} \quad \forall k \in [t, t+N-1] \\ & \left.\begin{aligned} A(t)^T \lambda_l(k|t) &= \begin{bmatrix} e_{l1}\varphi(k|t) \\ \vdots \\ e_{ln_y}\varphi(k|t) \end{bmatrix} \\ b(t)^T \lambda_l(k|t) &\le p_l - \bar{d}_l \\ \lambda_l(k|t) &\ge \mathbf{0} \end{aligned}\right\} \begin{array}{c} \forall l = 1, \ldots, n_o \\ \forall k \in [t+1, t+N] \end{array} \\ & \varphi(t+N|t) = W\varphi(t+N|t) + Zu(t+N-1|t), \end{aligned} \tag{2.42}$$

where the tracking error $\hat{e}(t)$ is defined as follows:

$$\hat{e}(t) = \hat{y}(t) - y_{\text{des}}(t). \tag{2.43}$$

The FHOCP (2.42) is a QP and can thus be solved efficiently. We note that the size of this QP depends, among other things, on the total number of faces of the MSPs, i.e., $r_A(t)$. This arises from the fact that the constraints (2.39) that guarantee robust input and output constraint satisfaction are constraining the auxiliary decision variables vector Λ (2.37), which is a vector in $\mathbb{R}^{n_o N r_A(t)}$.
Algorithm 1 can guarantee recursive feasibility of (2.42) along with robust input and output constraint satisfaction, as the following theorem shows.

Theorem 1. Let the Assumptions 1 and 2 hold, and further assume that the FHOCP (2.42) Algorithm 1 is solving is feasible at time $t = 0$. Then, the FHOCP (2.42) is recursively feasible and the closed-loop system constructed by the operation of Algorithm 1 guarantees robust satisfaction of input and output constraints for all $t \geq 0$ [3].

Key property needed for the proof of Theorem 1 is that the MSPs are non-expanding, i.e., $\mathcal{F}_j(t+1) \subseteq \mathcal{F}_j(t), j = 1, \ldots, n_y$. Thus an adaptive MPC algorithm, using those real-time SMI engines that are running PU algorithms which enforce that the polytopes are non-expanding, can guarantee recursive feasibility and robust input and output constraint satisfaction.
Feasibility of (2.42) at $t = 0$ implies that under the initial assumptions there exists a nonzero (possibly very small initially) input sequence that shows no violation of input and output constraints for all plant models belonging to the initial model set $\mathcal{F}(0)$, which specifies a practically reasonable case.

Chapter 3

Improvements on Real-Time Polytopic Set-Membership Identification

Proposals of the thesis for improving the real-time polytopic SMI engine of the adaptive MPC algorithm are presented in this chapter. The PU algorithm described in 2.3.1 suffers from two weaknesses:

1. Information content of measurements are not taken into account during updates, resulting in uninformative faces being added to the MSP and unnecessarily taking up valuable, limited space for the polytope faces, resulting in an overall disadvantageous SMI.

2. Identification has to stop completely when the number of faces in the MSP reaches its predefined limit M_{A_j}.

Although the BCPU algorithm described in 2.3.2 specifies a remedy for the second point via bounded complexity style updates, it has the weakness that the set of directions $\hat{D}_j$ is defined a priori without consideration of plant dynamics, leading to disadvantageous bounded complexity style SMI.
This chapter proposes improvements that address both points mentioned above and the weakness of the BCPU. Section 3.1 contains methods to evaluate information content of measurements, which can improve the quality of the identified MSPs. 3.1.1 contains a face filter based algorithm that can be used with both PU and BCPU, which bases the decision about whether adding a face to the MSP or not on the estimated information content of a measurement. 3.1.2 develops this algorithm further by adding self-tuning capability to the filter. Section 3.2 improves on the BCPU method given in 2.3.2 by adding learning capabilities that can tune the method to plant dynamics, resulting in sets of directions vectors that resemble the regressor vectors more than a predefined set, thus better bounded complexity style identification opportunities.

3.1 Improvements on Polytopic Update

3.1.1 Face Filtering Polytopic Update

The Face Filtering Polytopic Update (FFPU) improves on the PU by introducing a face filtering mechanism that can evaluate information content of the MSs. The decision about

whether to add a face of the MS to the MSP is then based on this evaluation, leading to the MSP being updated only by faces having information content above a certain threshold. The number of faces of a MSP is a limited resource and thus has to be used wisely; the FFPU method provides a way of cleverly utilizing this resource. This is made possible by selectively updating via adding faces with high information content while filtering out uninformative ones.

The information content of a face of $\mathcal{M}_j(t)$ can be estimated by how deep it is going to cut into the polytope. The depth of this cut can be quantified using the concept of cut ratio, which is the ratio of two specific scalar values defined on the MS $\mathcal{M}_j(t)$ and the MSP $\mathcal{F}_j(t-1)$. The first one is the width of the Polytope Support Strip (PSS). The PSS is defined as

$$\mathcal{S}_j^{\mathrm{P}}(t) = \{H_j \in \mathbb{R}^{n_u m} \ : \ q_{\mathrm{l}_j}^{\mathrm{P}} \leq \varphi(t)^T H_j \leq q_{\mathrm{u}_j}^{\mathrm{P}}\}, \tag{3.1}$$

where $q_{\mathrm{l}_j}^{\mathrm{P}}$ and $q_{\mathrm{u}_j}^{\mathrm{P}}$ are calculated by solving the following two LPs:

$$\begin{aligned} q_{\mathrm{l}_j}^{\mathrm{P}} &= \min\{\varphi(t)^T\theta \mid \theta \in \mathcal{F}_j(t-1)\} \\ q_{\mathrm{u}_j}^{\mathrm{P}} &= \max\{\varphi(t)^T\theta \mid \theta \in \mathcal{F}_j(t-1)\}, \end{aligned} \tag{3.2}$$

which also yields the vertices $p_{\mathrm{l}_j}^{\mathrm{P}}$ and $p_{\mathrm{u}_j}^{\mathrm{P}}$ of the MSP belonging to the boundary hyperplanes of the PSS, which can be written as

$$\begin{aligned} p_{\mathrm{l}_j}^{\mathrm{P}} &= \{\theta \mid \theta \in \mathcal{F}_j(t-1),\ \varphi(t)^T\theta = q_{\mathrm{l}_j}^{\mathrm{P}}\} \\ p_{\mathrm{u}_j}^{\mathrm{P}} &= \{\theta \mid \theta \in \mathcal{F}_j(t-1),\ \varphi(t)^T\theta = q_{\mathrm{u}_j}^{\mathrm{P}}\}. \end{aligned} \tag{3.3}$$

The width of the PSS $\mathcal{S}_j^{\mathrm{P}}(t)$ is then defined as the distance between these two vertices projected on the direction of the regressor vector $\varphi(t)$

$$d_{\mathrm{PSS}_j} = \left\| \frac{\varphi(t)^T(p_{\mathrm{u}_j}^{\mathrm{P}} - p_{\mathrm{l}_j}^{\mathrm{P}})}{\varphi(t)^T\varphi(t)}\varphi(t) \right\|_2. \tag{3.4}$$

Figure 3.1 illustrates the PSS $\mathcal{S}_j^{\mathrm{P}}(t)$ of the MSP $\mathcal{F}_j(t-1)$, and its width d_{PSS_j}.

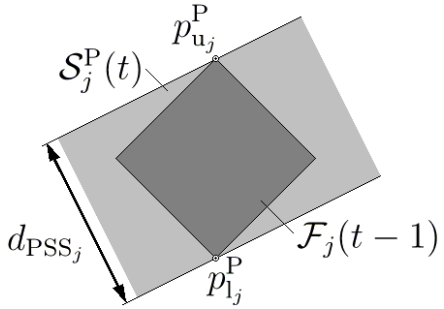

Figure 3.1: Polytope support strip and its width.

The second scalar we define for the cut ratio is the distance between a face and the opposing vertex belonging to the boundary hyperplane of the PSS. For the upper face $\mathcal{M}_{\mathrm{u}_j}(t)$, defined as

$$\mathcal{M}_{\mathrm{u}_j}(t) \triangleq \{H_j \in \mathbb{R}^{n_u m} \ : \ \varphi(t)^T H_j \leq \tilde{y}_j(t) + \epsilon_{d_j} + \epsilon_{v_j}\}, \tag{3.5}$$

this distance is defined as

$$d_{\mathrm{u}_j} \triangleq \left\| \frac{\tilde{y}_j(t) + \epsilon_{d_j} + \epsilon_{v_j} - \varphi(t)^T p^{\mathrm{P}}_{\mathrm{l}_j}}{\varphi(t)^T \varphi(t)} \right\|_2, \tag{3.6}$$

whereas for the lower face $\mathcal{M}_{\mathrm{l}_j}(t)$, defined as

$$\mathcal{M}_{\mathrm{l}_j}(t) \triangleq \{H_j \in \mathbb{R}^{n_u m} \ : \ -\varphi(t)^T H_j \leq -\tilde{y}_j(t) + \epsilon_{d_j} + \epsilon_{v_j}\}, \tag{3.7}$$

it is defined as

$$d_{\mathrm{l}_j} = \left\| \frac{-\tilde{y}_j(t) + \epsilon_{d_j} + \epsilon_{v_j} + \varphi(t)^T p^{\mathrm{P}}_{\mathrm{u}_j}}{\varphi(t)^T \varphi(t)} \right\|_2. \tag{3.8}$$

Finally we can define the cut ratios of the two faces of $\mathcal{M}_j(t)$ as follows

$$\kappa_{\mathrm{u}_j} \triangleq \frac{d_{\mathrm{u}_j}}{d_{\mathrm{PSS}_j}}, \qquad \kappa_{\mathrm{l}_j} \triangleq \frac{d_{\mathrm{l}_j}}{d_{\mathrm{PSS}_j}}. \tag{3.9}$$

Figure 3.2 illustrates the cut ratios κ_{u_j} and κ_{l_j} and the terms related to their definition, namely d_{u_j} and d_{l_j}, the MSP $\mathcal{F}_j(t-1)$, and the upper and lower faces of the MS $\mathcal{M}_j(t)$ (i.e., $\mathcal{M}_{\mathrm{u}_j}(t)$ and $\mathcal{M}_{\mathrm{l}_j}(t)$, respectively). For the case in Figure 3.2 the lower face $\mathcal{M}_{\mathrm{l}_j}(t)$ appears to be providing decent information with a cut ratio of 0.69 (it will be cutting away roughly 31% of the MSP $\mathcal{F}_j(t-1)$), whereas the upper face $\mathcal{M}_{\mathrm{u}_j}(t)$, having a cut ratio of 0.94, carries little information (it will be cutting away roughly only 6% of the MSP $\mathcal{F}_j(t-1)$).

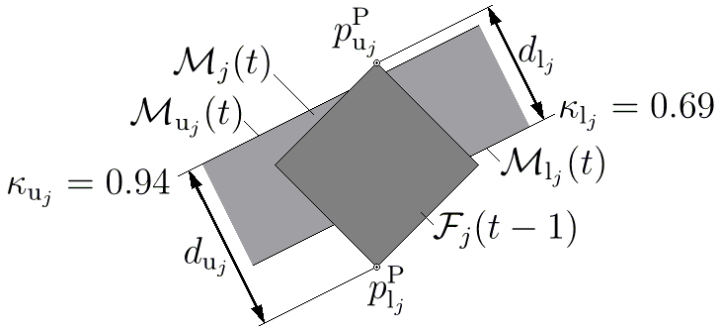

Figure 3.2: Cut ratios and terms related to their definition.

The cut ratios provide estimates of the information content of the faces constituting MS $\mathcal{M}_j(t)$ with regards to a possible polytopic update on the MSP $\mathcal{F}_j(t-1)$. It is then possible to base the decision about adding a face to the MSP on this estimate, by introducing a face filtering procedure during the polytopic update. With this method only those faces that are informative are added to the MSP. Faces with cut ratios less than 1 can be classified as informative in general, since they will be cutting away some part of the MSP, whereas those with cut ratios equal to or greater than 1 are redundant and will not provide any new information (i.e., they will yield no decrease in the polytope size).
To be able to tune the behaviour of the method we introduce a face filtering parameter Γ_{A_j}, which is to be chosen in the interval $[0, 1]$. A value of Γ_{A_j} equal to 1 will imply that every non-redundant face will be added to the MSP, whereas values less than 1 imply that faces should be providing new information (i.e., they should be decreasing polytope size) to be added. A face filter parameter Γ_{A_j} equal to 0.5, for example, means that a face

should be cutting away roughly half of the MSP in order to be classified as informative, which will specify a highly demanding criterion of informativeness.
The face filtering method can be formally expressed as follows: The faces $\mathcal{M}_{\mathrm{u}_j}(t)$ and $\mathcal{M}_{\mathrm{l}_j}(t)$ are informative if

$$\kappa_{\mathrm{u}_j} < \Gamma_{A_j}, \quad \kappa_{\mathrm{l}_j} < \Gamma_{A_j}. \tag{3.10}$$

The FFPU algorithm based on the face filtering method is summarized in Algorithm 4.

Algorithm 4 Face filtering polytopic update

0) Repeat for each output j, $j = 1, \ldots, n_y$.
1) At time step t, use the regressor vector $\varphi(t)$ and the measurement $\tilde{y}_j(t)$ to build the MS $\mathcal{M}_j(t) = \{H_j \in \mathbb{R}^{n_u m} \ : \ |\varphi(t)^T H_j - \tilde{y}_j(t)| \leq \epsilon_{d_j} + \epsilon_{v_j}\}$;
2) Check informativeness of the faces of $\mathcal{M}_j(t)$ with (3.10) using a predefined Γ_{A_j};
3) Update $\mathcal{F}_j(t-1)$ to $\mathcal{F}_j(t)$ by adding the informative faces to $A_j(t-1)H_j \leq b_j(t-1)$;
4) Remove redundant faces from $\mathcal{F}_j(t)$ to obtain $A_j(t)$ and $b_j(t)$.

Algorithm 4 can guarantee that the MSPs are non-expanding, i.e., $\mathcal{F}_j(t+1) \subseteq \mathcal{F}_j(t), j = 1, \ldots, n_y$, thus recursive feasibility and output constraint satisfaction are guaranteed with an adaptive MPC algorithm using the FFPU method for real-time polytopic SMI.

3.1.2 Self-tuning Face Filtering Polytopic Update

It is also possible to introduce a self-tuning mechanism for the face filtering parameter Γ_{A_j} of the FFPU method given in Algorithm 4, based on the extent to which the space for number of faces is occupied. A simple function for mapping this extent of occupation to a Γ_{A_j} value is needed, which can for example be chosen as follows

$$\Gamma_{A_j} = \Gamma^{\mathrm{m}}_{A_j} + \Gamma^{\mathrm{a}}_{A_j} \mathrm{erf}(2 - 4\frac{r_{A_j}(t) - r^0_{A_j}(t)}{M_{A_j} - r^0_{A_j}(t)}), \tag{3.11}$$

where $\Gamma^{\mathrm{m}}_{A_j}$ and $\Gamma^{\mathrm{a}}_{A_j}$ are design parameters defining the medium value and the amplitude of the mapping, respectively, erf is the error function defined as

$$\mathrm{erf}(x) \triangleq \frac{2}{\sqrt{\pi}} \int_0^x \mathrm{e}^{-k^2} \mathrm{d}k, \tag{3.12}$$

$r_{A_j}(t)$ is the number of faces in the $A_j(t)H_j \leq b_j(t)$ part of the current MSP $\mathcal{F}_j(t)$, $r^0_{A_j}(t)$ is the number of faces that were predefined as faces of $A_j(0)H_j \leq b_j(0)$ (i.e., the initial MSP $\mathcal{F}_j(0)$) that are still inside $\mathcal{F}_j(t)$, and M_{A_j} is the face number limit on the MSP.
The self-tuning FFPU operates first by calculating a Γ_{A_j} value using a function, e.g. the one given in (3.11). The face filter is thus tuned in real-time according to how close r_{A_j} is to the face number limit M_{A_j}, with Γ_{A_j} progressively decreasing (filter becoming more stringent) as the MSP is filled up with faces. The obtained Γ_{A_j} value is then used as the threshold for judging informativeness of MS faces (as opposed to an a priori defined fixed Γ_{A_j} as in the previous case with the FFPU given in Algorithm 4). The procedure for the self-tuning FFPU is given in Algorithm 5.
Similar to Algorithm 4, Algorithm 5 can also guarantee that the MSPs are non-expanding, i.e., $\mathcal{F}_j(t+1) \subseteq \mathcal{F}_j(t), j = 1, \ldots, n_y$, thus recursive feasibility and output constraint satisfaction are guaranteed with an adaptive MPC algorithm using the self-tuning FFPU method.

Algorithm 5 Self-tuning face filtering polytopic update

0) Repeat for each output j, $j = 1, \ldots, n_y$.
1) At time step t, use the regressor vector $\varphi(t)$ and the measurement $\tilde{y}_j(t)$ to build the MS $\mathcal{M}_j(t) = \{H_j \in \mathbb{R}^{n_u m} \ : \ |\varphi(t)^T H_j - \tilde{y}_j(t)| \leq \epsilon_{d_j} + \epsilon_{v_j}\}$;
2) Check informativeness of the faces of $\mathcal{M}_j(t)$ with (3.10) using the Γ_{A_j} value found by e.g. (3.11);
3) Update $\mathcal{F}_j(t-1)$ to $\mathcal{F}_j(t)$ by adding the informative faces to $A_j(t-1)H_j \leq b_j(t-1)$;
4) Remove redundant faces from $\mathcal{F}_j(t)$ to obtain $A_j(t)$ and $b_j(t)$.

3.2 Improvements on Bounded Complexity Polytopic Update

3.2.1 Gradual Learning Bounded Complexity Polytopic Update

Unlike the case with the BCPU described in 2.3.2, in the Gradual Learning (GL)-BCPU the set of direction vectors is not defined a priori. Instead, the method starts with an empty set when closed-loop operation begins and constructs the set progressively by adding those regressor vectors that come from informative faces, informativeness defined as given in (3.10). This enables the method to learn the set of direction vectors in accord with the plant dynamics, resulting in better bounded complexity style SMI opportunities. Instead of writing the MSP in the usual manner, i.e.

$$\mathcal{F}_j(t) = \{H_j \in \mathbb{R}^{n_u m} \ : \ A_j(t)H_j \leq b_j(t)\}, \tag{3.13}$$

in the GL-BCPU it is written as follows

$$\mathcal{F}_j(t) = \left\{ H_j \in \mathbb{R}^{n_u m} \ : \ \begin{bmatrix} A_j(t) \\ D_j(t) \end{bmatrix} H_j \leq \begin{bmatrix} b_j(t) \\ b_j^D(t) \end{bmatrix} \right\}. \tag{3.14}$$

Here, the set of vectors constituting the rows of the matrix $D_j(t)$ take the place of the set of direction vectors $\hat{D}_j$ in the BCPU method described in 2.3.2. Instead of a predefined set as in the case of the BCPU, here the matrix $D_j(t)$ is empty at time step $t = 0$. During operation, first the $A_j(t)H_j \leq b_j(t)$ part of the MSP is filled with faces using Algorithm 4 or 5. After the number of faces in $A_j(t)H_j \leq b_j(t)$ reaches its predefined limit M_{A_j}, the GL-BCPU method starts learning direction vectors using the regressor vectors belonging to informative MSs. Those faces that are informative are inserted to the $D_j(t)H_j \leq b_j^D(t)$ part of the MSP, thus they also gradually construct the set of direction vectors as rows of $D_j(t)$. Here, to be able to separately tune the behavior of the face filter of the learning algorithm, another face filtering parameter is introduced, denoted by Γ_{D_j}, with the same function as the Γ_{A_j}, but used for updating the $D_j(t)H_j \leq b_j^D(t)$ part of $\mathcal{F}_j(t)$. When the number of faces in $D_j(t)H_j \leq b_j^D(t)$, i.e. $r_{D_j}(t)$, reaches its predefined limit M_{D_j}, the method continues with updating the MSP in the style of the BCPU as given in Algorithm 3, with the difference that the set of directions consisting of the rows of the matrix $D_j(t)$ is used instead of a predefined D set. Afterwards, if $r_{D_j}(t)$ drops below M_{D_j} due to removal of redundant faces, the method can still continue to learn new directions by adding faces to $D_j(t)H_j \leq b_j^D(t)$. The GL-BCPU procedure is given in Algorithm 6.
Algorithm 6 can guarantee that the MSPs are non-expanding, i.e., $\mathcal{F}_j(t+1) \subseteq \mathcal{F}_j(t), j = 1, \ldots, n_y$, thus recursive feasibility and output constraint satisfaction are guaranteed.

Algorithm 6 Gradual learning bounded complexity polytopic update

0) Repeat for each output j, $j = 1, \ldots, n_y$.
1) At time step t, use the regressor vector $\varphi(t)$ and the measurement $\tilde{y}_j(t)$ to build the MS $\mathcal{M}_j(t) = \{H_j \in \mathbb{R}^{n_u m} \ : \ |\varphi(t)^T H_j - \tilde{y}_j(t)| \leq \epsilon_{d_j} + \epsilon_{v_j}\}$;
2) Using Algorithm 4 or 5 with $\mathcal{M}_j(t)$, if $r_{A_j}(t-1) < M_{A_j}$ update the $A_j(t-1)H_j \leq b_j(t-1)$ part of $\mathcal{F}_j(t-1)$ to $\mathcal{F}_j(t)$, otherwise if $r_{D_j}(t-1) < M_{D_j}$ update the $D_j(t-1)H_j \leq b_j^D(t-1)$ part of $\mathcal{F}_j(t-1)$ to $\mathcal{F}_j(t)$, otherwise update $\mathcal{F}_j(t-1)$ to $\mathcal{F}_j(t)$ with Algorithm 3 using the rows of $D_j(t)$ as the set of directions D;
3) Remove redundant faces from $\mathcal{F}_j(t)$ to obtain $A_j(t)$, $b_j(t)$, $D_j(t)$, and $b_j^D(t)$.

3.2.2 Batch Learning Bounded Complexity Polytopic Update

Batch Learning (BL)-BCPU is a variant of the GL method given in Algorithm 6. Here the matrix $D_j(t)$ is constructed within that time step t at which the number of faces in $A_j(t)H_j \leq b_j(t)$ reaches the total face number limit $M_{A_j} + M_{D_j}$: The best M_{D_j} faces (i.e., those with the lowest cut ratios) in $A_j(t)H_j \leq b_j(t)$ are selected to be transferred to $D_j(t)H_j \leq b_j^D(t)$; they are thus learned in batch during a single time step. Identification continues afterwards in the style of BCPU as in Algorithm 3, using the rows of $D_j(t)$ as the set of direction vectors. The procedure of the BL-BCPU method is summarized in Algorithm 7.

Algorithm 7 Batch learning bounded complexity polytopic update

0) Repeat for each output j, $j = 1, \ldots, n_y$.
1) At time step t, use the regressor vector $\varphi(t)$ and the measurement $\tilde{y}_j(t)$ to build the MS $\mathcal{M}_j(t) = \{H_j \in \mathbb{R}^{n_u m} \ : \ |\varphi(t)^T H_j - \tilde{y}_j(t)| \leq \epsilon_{d_j} + \epsilon_{v_j}\}$;
2) Using Algorithm 4 or 5 with $\mathcal{M}_j(t)$, if $r_{A_j}(t-1) < M_{A_j} + M_{D_j}$ update the $A_j(t-1)H_j \leq b_j(t-1)$ part of $\mathcal{F}_j(t-1)$ to $\mathcal{F}_j(t)$, otherwise if $r_{A_j}(t-1) = M_{A_j} + M_{D_j}$ transfer M_{D_j} faces with the lowest cut ratios from $A_j(t-1)H_j \leq b_j(t-1)$ to $D_j(t-1)H_j \leq b_j^D(t-1)$, otherwise update $\mathcal{F}_j(t-1)$ to $\mathcal{F}_j(t)$ with Algorithm 3 using the rows of $D_j(t)$ as the set of directions D;
3) Remove redundant faces from $\mathcal{F}_j(t)$ to obtain $A_j(t)$, $b_j(t)$, $D_j(t)$, and $b_j^D(t)$.

Similar to Algorithm 6, Algorithm 7 can also guarantee that the MSPs are non-expanding, i.e., $\mathcal{F}_j(t+1) \subseteq \mathcal{F}_j(t), j = 1, \ldots, n_y$, thus recursive feasibility and output constraint satisfaction are guaranteed.

Chapter 4

Extensions for Time-varying Systems

Polytopic methods for handling time-varying systems are presented in this chapter. Section 4.1 proposes inflating the polytope for handling slow variations in plant dynamics. A center projecting and shifting method for dealing with rapidly varying dynamics is shown in Section 4.2. Finally, Section 4.3 consolidates these two methods into an operational real-time polytopic SMI algorithm that can handle both slow and rapid variations in plant dynamics.

4.1 Polytopic Set Inflation

For handling plants with slowly varying dynamics, past information on the plant has to be gradually forgotten. The way to do this is to inflate the MSP by increasing disturbance and noise bounds of past MSs recorded inside the MSP as inequalities, which is the idea of the Polytopic Set Inflation (PSI) method presented in this section, inspired by the exponential bound inflation method of Veres and Norton in [10].
Consider the MSP at time $t-1$:

$$\mathcal{F}_j(t-1) = \{H_j \in \mathbb{R}^{n_u m} \ : \ A_j(t-1)H_j \leq b_j(t-1)\}, \tag{4.1}$$

where $A_j(t-1)H_j \leq b_j(t-1)$ has faces that were predefined for specifying the initial MSP $\mathcal{F}_j(0)$, alongside those that were added to it during operation as faces from the MSs with polytopic updates. For the inequalities defining the initial MSP $\mathcal{F}_j(0)$ there is no distinction between measurements $\tilde{y}_j(t)$ and bounds $\epsilon_{d_j} + \epsilon_{v_j}$, thus for these faces no inflation can be made via increasing disturbance and noise bounds. To make this distinction for the faces coming from MSs, we can write the MSP in the following form

$$\mathcal{F}_j(t-1) = \left\{ H_j \in \mathbb{R}^{n_u m} \ : \ \begin{bmatrix} A_j(0) \\ A_j^{\mathcal{M}}(t-1) \end{bmatrix} H_j \leq \begin{bmatrix} b_j(0) \\ b_j^{\tilde{y}}(t-1) + b_j^{\epsilon}(t-1) \end{bmatrix} \right\}, \tag{4.2}$$

where $A_j^{\mathcal{M}}(t-1)H_j \leq b_j^{\tilde{y}}(t-1) + b_j^{\epsilon}(t-1)$ contains the faces coming from MSs, with the vector $b_j^{\tilde{y}}(t-1)$ containing past measurements $\tilde{y}_j(k),\ k = 0, \ldots, t-1$ and the vector $b_j^{\epsilon}(t-1)$ containing the corresponding magnitude bounds $\epsilon_{d_j}(k) + \epsilon_{v_j}(k), k = 0, \ldots, t-1$. The PSI operation can then be given as the following inflation of the MSP

$$\hat{\mathcal{F}}_j(t-1) = \left\{ H_j \in \mathbb{R}^{n_u m} \ : \ \begin{bmatrix} A_j(0) \\ A_j^{\mathcal{M}}(t-1) \end{bmatrix} H_j \leq \begin{bmatrix} b_j(0) \\ b_j^{\tilde{y}}(t-1) + \Omega_{\text{PSI}} b_j^{\epsilon}(t-1), \end{bmatrix} \right\} \tag{4.3}$$

where $\hat{\mathcal{F}}_j(t-1)$ denotes the inflated version of $\mathcal{F}_j(t-1)$, and the PSI factor Ω_{PSI} is a design parameter to be chosen greater than 1. If we write the MSP $\mathcal{F}_j(t-1)$ as the intersection of the initial MSP $\mathcal{F}_j(0)$ and past MSs:

$$\mathcal{F}_j(t-1) = \mathcal{F}_j(0) \cap \left\{ \bigcap_{k=0,\dots,t-1} \left|\varphi(k)^T H_j - \tilde{y}_j(k)\right| \leq \epsilon_{d_j}(k) + \epsilon_{v_j}(k) \right\}, \tag{4.4}$$

then the inflated part of $\mathcal{F}_j(t-1)$ that comes from past MSs can also be written as

$$\begin{aligned} &\{H_j \in \mathbb{R}^{n_u m} \ : \ A_j^{\mathcal{M}}(t-1)H_j \leq b_j^{\tilde{y}}(t-1) + \Omega_{\text{PSI}} b_j^{\epsilon}(t-1)\} \\ &\qquad = \bigcap_{k=0,\dots,t-1} \left|\varphi(k)^T H_j - \tilde{y}_j(k)\right| \leq \Omega_{\text{PSI}}^{t-k}(\epsilon_{d_j}(k) + \epsilon_{v_j}(k)), \end{aligned} \tag{4.5}$$

and thus it can be seen that through PSI the faces from MSs that were recorded further away in the past will have their bounds increased more than those that are relatively recent; since the dynamics of the plant is slowly varying, old measurements are less relevant than new ones, so they also have to be forgotten more strongly. The PSI method makes it possible for polytopic methods to handle systems with slowly varying dynamics. The downside is that the property of MSPs being non-expanding, i.e., $\mathcal{F}_j(t+1) \subseteq \mathcal{F}_j(t), j = 1, \dots, n_y$, is lost with the PSI method, thus recursive feasibility and output constraint satisfaction cannot be guaranteed, which leads to the requirement of having to use MPC formulations with soft output constraints.

4.2 Polytopic Center Tracking

To handle plants with rapidly varying dynamics, countermeasures that are more radical than the PSI method have to be taken. We propose the Polytopic Center Tracking (PCT) method, inspired by the center-adjusting polyhedric tracker method of Piet-Lahanier and Walter in [11]. In the PCT method it is assumed that if there is an abrupt change in plant dynamics, the intersection of the MS $\mathcal{M}_j(t)$ and the MSP $\mathcal{F}_j(t-1)$ will be an empty set:

$$\mathcal{M}_j(t) \cap \mathcal{F}_j(t-1) = \varnothing, \tag{4.6}$$

which implies that an update will lead to an empty model set, and thus it should be avoided. The way to avoid this is to translate $\mathcal{F}_j(t-1)$ in accord with $\mathcal{M}_j(t)$, such that the intersection of $\mathcal{M}_j(t)$ with the translated $\mathcal{F}_j(t-1)$ will no longer be an empty set. Consider the median hyperplane of the MS $\mathcal{M}_j(t)$:

$$\mathcal{M}_j^{\text{m}}(t) = \{H_j \in \mathbb{R}^{n_u m} \ : \ \varphi(t)^T H_j = \tilde{y}_j(t)\}. \tag{4.7}$$

The translation vector $V_{\text{T}}^{\text{P}}(t)$ is the vector connecting the nominal model $H_{c,j}(t-1)$ with its projection on the median hyperplane $\mathcal{M}_j^{\text{m}}(t)$:

$$V_{\text{T}}^{\text{P}}(t) = \varphi(t) \frac{\tilde{y}_j(t) - \varphi(t)^T H_{c,j}(t-1)}{\varphi(t)^T \varphi(t)}. \tag{4.8}$$

Considering the MSP $\mathcal{F}_j(t-1)$ in the following form

$$\mathcal{F}_j(t-1) = \{H_j \in \mathbb{R}^{n_u m} \ : \ A_j(t-1)H_j \leq b_j(t-1)\}, \tag{4.9}$$

the PCT operation can be given as the following translation of the MSP and its nominal model

$$\begin{aligned} \bar{\mathcal{F}}_j(t-1) &= \{H_j \in \mathbb{R}^{n_u m} \ : \ A_j(t)H_j \leq b_j(t) + A_j(t-1)V_{\mathrm{T}}^{\mathrm{P}}(t)\} \\ \bar{H}_{c,j}(t-1) &= H_{c,j}(t-1) + V_{\mathrm{T}}^{\mathrm{P}}(t), \end{aligned} \tag{4.10}$$

where $\bar{\mathcal{F}}_j(t-1)$ and $\bar{H}_{c,j}(t-1)$ denote the translated versions of $\mathcal{F}_j(t-1)$ and $H_{c,j}(t-1)$, respectively.
The PCT method makes it possible for polytopic methods to handle systems with rapidly varying dynamics. The downside is that, similar to the case with the PSI method, the property of MSPs being non-expanding, i.e., $\mathcal{F}_j(t+1) \subseteq \mathcal{F}_j(t), j = 1, \ldots, n_y$, is lost with the PCT method. Recursive feasibility and output constraint satisfaction can thus no longer be guaranteed, which leads to the requirement of having to use MPC formulations with soft output constraints.

4.3 Polytopic Update Algorithm for Time-varying Systems

This section consolidates the PSI and PCT methods, presented in the previous sections of this chapter, into an algorithm that is able to handle time-varying systems with both slow and rapid variations in their dynamics. First the definition of Polytope Tight Strip (PTS) is presented since it is used in the algorithm for detecting MSP infeasibility (i.e., $\mathcal{M}_j(t) \cap \mathcal{F}_j(t-1) = \varnothing$).

Polytope Tight Strip

Given the MSP $\mathcal{F}_j(t-1) = \{H_j \in \mathbb{R}^{n_u m} \ : \ A_j(t-1)H_j \leq b_j(t-1)\}$ and the MS $\mathcal{M}_j(t) = \{H_j \in \mathbb{R}^{n_u m} \ : \ |\varphi^T H_j - \tilde{y}_j| \leq \epsilon_{d_j} + \epsilon_{v_j}\}$, the PTS is defined as $\mathcal{T}_j^{\mathrm{P}}(t) \triangleq \mathcal{M}_j(t) \cap \mathcal{S}_j^{\mathrm{P}}(t)$, where $\mathcal{S}_j^{\mathrm{P}}(t)$ is the PSS given in Equation (3.1).

Polytopic update for time-varying systems

The PU algorithm that employs the PSI and PCT methods, and thus can deal with plants that may have both slow and rapid variations in their dynamics, is summarized in Algorithm 8.

Algorithm 8 Polytopic update for time-varying systems

0) Repeat for each output j, $j = 1, \ldots, n_y$.
1) At time step t, use the regressor vector $\varphi(t)$ and the measurement $\tilde{y}_j(t)$ to build the MS $\mathcal{M}_j(t) = \{H_j \in \mathbb{R}^{n_u m} \ : \ |\varphi(t)^T H_j - \tilde{y}_j(t)| \leq \epsilon_{d_j} + \epsilon_{v_j}\}$;
2) If the PTS $\mathcal{T}_j^{\mathrm{P}}(t)$ (built as given in 4.3) is an empty set conduct PCT (as given in (4.10)), inflate the resulting MSP using (4.3) with $\Omega_{\mathrm{PSI}} = \Omega_{\mathrm{PE}}$, switch PSI method on, otherwise if the PTS has been empty less than N_{PCT} time steps ago, conduct PCT (as given in (4.10)) and inflate the resulting MSP using (4.3) with $\Omega_{\mathrm{PSI}} = \Omega_{\mathrm{TPE}}$, otherwise if PSI method is switched on inflate the MSP using (4.3) with $\Omega_{\mathrm{PSI}} = \Omega_{\mathrm{DPI}}$ and continue with updating the MSP with the MS using the PU method of choice;
3) Remove redundant faces from $\mathcal{F}_j(t)$ to obtain $A_j(t)$ and $b_j(t)$.

The third step of Algorithm 8 is the normal PCT step, where a PCT is conducted upon an infeasibility detection. In this step the MSP is inflated with polytopic explosion factor Ω_{PE}, since an abrupt change in plant dynamics necessitates an abrupt and strong

forgetting of past information. The Ω_{PE} parameter should thus be selected larger than what normally would be selected as Ω_{DPI}, which is for countering slow variations in dynamics. The fourth step is the trailing PCT step: Since the median hyperplane $\mathcal{M}_j^{\text{m}}$ employed in the PCT method is a function of the regressor vector $\varphi(t)$, it is expected that conducting a couple of PCT operations instead of a single one is more reasonable as $\varphi(t)$ will continue to be effected by the abrupt change in dynamics for a couple of time steps. The N_{PCT} parameter here specifies how many times the PCT will be conducted in a row after an infeasibility happens (i.e., PTS becomes empty), and in general should be selected with relation to the FIR length m, since the length of φ depends on this value. The Ω_{TPE} parameter is the trailing polytopic explosion factor, which should be selected less than Ω_{PE}, but should still be larger than Ω_{DPI} due to the same reasoning given for Ω_{PE}. The fifth step is a default PSI step that is for countering slow variations in dynamics, thus the polytopic inflation factor Ω_{DPI} should be selected such that it will specify a slow forgetting of past information. The speed of forgetting should be chosen according to the speed of change in plant dynamics.

Algorithm 8 makes it possible for polytopic methods to handle systems with both slowly and rapidly varying dynamics. The downside is that, since it employs the PSI and PCT methods, both of which lack the property of MSPs being non-expanding, i.e., $\mathcal{F}_j(t+1) \subseteq \mathcal{F}_j(t),\ j = 1, \ldots, n_y$, recursive feasibility and output constraint satisfaction cannot be guaranteed. This leads to the requirement of having to use MPC formulations with soft output constraints. For the QP (2.42), such a formulation is given with Equation (4.11):

$$
\begin{aligned}
\underset{U,\Lambda,\Xi}{\text{minimize}} \quad & \sum_{k=t}^{t+N-1} \left(\|\hat{e}(k+1|t)\|_Q^2 + \|u(k|t)\|_S^2 + \|\Delta u(k|t)\|_R^2 + \|\xi(k+1|t)\|_P^2 \right) \\
\text{subject to} \quad & \left.\begin{array}{l} Cu(k|t) \le d \\ L\Delta u(k|t) \le f \end{array}\right\} \quad \forall k \in [t, t+N-1] \\
& \left.\begin{array}{l} A(t)^T \lambda_l(k|t) = \begin{bmatrix} e_{l1}\varphi(k|t) \\ \vdots \\ e_{ln_y}\varphi(k|t) \end{bmatrix} \\ b(t)^T \lambda_l(k|t) \le p_l - \bar{d}_l + \xi_l(k|t) \\ \lambda_l(k|t) \ge \mathbf{0} \\ \xi_l(k|t) \ge 0 \end{array}\right\} \begin{array}{l} \forall l = 1, \ldots, n_o \\ \forall k \in [t+1, t+N] \end{array} \\
& \varphi(t+N|t) = W\varphi(t+N|t) + Zu(t+N-1|t),
\end{aligned}
\tag{4.11}
$$

with the slack variables vector $\Xi \in \mathbb{R}^{n_o(N-1)}$ for the soft output constraints defined as:

$$
\Xi \triangleq \begin{bmatrix} \xi(t+1|t) \\ \vdots \\ \xi(t+N|t) \end{bmatrix}, \tag{4.12}
$$

where $\xi(t) \in \mathbb{R}^{n_o}$ is defined as:

$$
\xi(t) \triangleq \begin{bmatrix} \xi_1(t) \\ \vdots \\ \xi_{n_o}(t) \end{bmatrix}, \tag{4.13}
$$

and $P \in \mathbb{S}_+^{n_o}$ is the weighting matrix on the slack variables.

Chapter 5

Zonotopic Methods

This chapter studies zonotopes and their use for representing model sets in real-time SMI. Section 5.1 provides an overview of zonotopes as geometrical objects. Sections 5.2 and 5.3 present some recent zonotopic SMI methods in the literature that have the order preserving property. Sections 5.4 and 5.5 propose zonotopic counterparts of the polytopic methods for handling time-varying systems given in Chapter 4. Finally, Section 5.6 consolidates these last methods into an operational real-time zonotopic SMI algorithm that can deal with both slowly and rapidly varying plant dynamics.

5.1 Zonotope Fundamentals

Zonotopes are a special class of convex polytopes. Specifically, an n-zonotope of order r is the linear image of an r-dimensional hypercube in $\mathbb{R}^n$; another definition is that it is formed by taking the Minkowski sum of r straight line segments in $\mathbb{R}^n$ [12]. The order r is a measure of the geometrical complexity of the zonotope [13].
An n-zonotope of order r can be expressed through a vector $w \in \mathbb{R}^n$ and a matrix $Z \in \mathbb{R}^{n \times r}$:

$$\begin{aligned} \mathcal{G} &= \{H \in \mathbb{R}^n \ : \ H = w \oplus Z\mathbf{B}^r\} \\ &= \{H \in \mathbb{R}^n \ : \ H = w + Zv, \ v \in \mathbf{B}^r\} \end{aligned} \tag{5.1}$$

where $\mathbf{B}^r$ is the r-dimensional hypercube centered at the origin and consisting of r one-dimensional unitary intervals $\mathbf{B} = [-1, 1]$. Thus, the matrix Z specifies the operator defining the shape of the zonotope, whereas the vector w denotes its center.
The zonotope has interesting properties that make it a computationally advantageous alternative to polytopes for representing model sets:

1. A Model Set Zonotope (MSZ) $\mathcal{G}_j(t) = w_j(t) \oplus Z_j(t)\mathbf{B}^r$ is fully defined by the vector $w_j(t)$ and the matrix $Z_j(t)$ (in fact the vector $w_j(t)$ directly provides the nominal model information, i.e., $w_j(t) = H_{c,j}(t)$). Thus, unlike polytopic methods, there is no need to remove redundant faces or compute nominal model, which point to savings in computational effort.

2. Certain zonotopic SMI methods have the property of preserving the zonotope order (i.e., r remains constant indefinitely); since the size of $w_j(t)$ and $Z_j(t)$ are fixed for a constant r, order preserving zonotopic methods are inherently bounded complexity methods, which have no need for special methods to bound complexity, e.g. the ones for polytopes given in Sections 2.3.2-3.2.1-3.2.2.

Following two sections describe some recent zonotopic SMI methods in the literature that have the order preserving property mentioned in the second point above.

5.2 Basic Zonotopic Update

The basic Zonotopic Update (ZU) calculates an overbounding zonotope $\hat{\mathcal{G}}$ that contains the intersection of a given strip $\mathcal{M}$ and a given zonotope $\mathcal{G}$. This section is based on the work of Bravo and coworkers in [13].

Zonotope Support Strip

Given a zonotope $\mathcal{G} = \{H \in \mathbb{R}^n \ : \ H = w \oplus Z\mathbf{B}^r\}$, and a vector φ, we define the Zonotope Support Strip (ZSS) as

$$\mathcal{S}^{\mathrm{Z}} = \{H \ : \ q_2^{\mathrm{Z}} \leq \varphi^T H \leq q_1^{\mathrm{Z}}\}, \tag{5.2}$$

where q_2^{Z} and q_1^{Z} are defined as

$$q_{\mathrm{l}}^{\mathrm{Z}} = \min_{H \in \mathcal{G}} \varphi^T H, \quad q_{\mathrm{u}}^{\mathrm{Z}} = \max_{H \in \mathcal{G}} \varphi^T H. \tag{5.3}$$

These can easily be calculated as follows

$$q_1^{\mathrm{Z}} = \varphi^T w + \left\| Z^T \varphi \right\|_1, \quad q_2^{\mathrm{Z}} = \varphi^T w - \left\| Z^T \varphi \right\|_1. \tag{5.4}$$

Zonotope Tight Strip

Given a zonotope $\mathcal{G}$ and a strip $\mathcal{M}$, the Zonotope Tight Strip (ZTS) is defined as

$$\mathcal{T}^{\mathrm{Z}} \triangleq \mathcal{M} \cap \mathcal{S}^{\mathrm{Z}}, \tag{5.5}$$

with $\mathcal{S}^{\mathrm{Z}}$ denoting the ZSS described in 5.2. We note that

$$\mathcal{S}^{\mathrm{Z}} \cap \mathcal{G} = \mathcal{G}, \tag{5.6}$$

hence

$$\mathcal{M} \cap \mathcal{G} = \mathcal{M} \cap \mathcal{S}^{\mathrm{Z}} \cap \mathcal{G} = \mathcal{T}^{\mathrm{Z}} \cap \mathcal{G}. \tag{5.7}$$

Since $\mathcal{T}^{\mathrm{Z}} \subseteq \mathcal{M}$, it is reported in [13] that taking the intersection of $\mathcal{G}$ with $\mathcal{T}^{\mathrm{Z}}$ instead of $\mathcal{M}$ leads to better identification.

Intersection of a Zonotope and a Strip

Here the order preserving zonotopic update method of [13] that bounds the intersection of a zonotope and a strip is presented. The following property lays out the method that obtains a family of zonotopes which contain the aforementioned intersection.

Property 1. Given an n-zonotope $\mathcal{G}$ of order r

$$\mathcal{G} = \{H \in \mathbb{R}^n \ : \ H = w \oplus Z\mathbf{B}^r\} \tag{5.8}$$

and the ZTS $\mathcal{T}^{\mathrm{Z}}$ obtained as shown in 5.2

$$\mathcal{T}^{\mathrm{Z}} \triangleq \mathcal{M} \cap \mathcal{S}^{\mathrm{Z}} = \{H \in \mathbb{R}^n \ : \ |\varphi^T H - \eta| \leq \sigma\}, \tag{5.9}$$

the family of zonotopes $v(k) \oplus T(k)\mathbf{B}^r$, parametrized with the integer $k,\ 0 \leq k \leq r$, contains the intersection of $\mathcal{G}$ and $\mathcal{T}^{\mathrm{Z}}$, i.e.:

$$\mathcal{G} \cap \mathcal{T}^{\mathrm{Z}} \subseteq v(k) \oplus T(k)\mathbf{B}^r, \quad 0 \leq k \leq r, \tag{5.10}$$

where

$$v(k) = \begin{cases} w + \left(\frac{\eta - \varphi^T w}{\varphi^T Z_k}\right) Z_k, & \text{if } 1 \leq k \leq r \text{ and } \varphi^T Z_k \neq 0 \\ w, & \text{otherwise} \end{cases}$$

$$T(k) = \begin{cases} [T_1^k\ T_2^k\ \ldots\ T_r^k], & \text{if } 1 \leq k \leq r \text{ and } \varphi^T Z_k \neq 0 \\ Z, & \text{otherwise} \end{cases}$$

$$T_i^k = \begin{cases} Z_i - \left(\frac{\varphi^T Z_i}{\varphi^T Z_k}\right) Z_k, & \text{if } i \neq k \\ \left(\frac{\sigma}{\varphi^T Z_k}\right) Z_k, & \text{if } i = k. \end{cases}$$

Proof. See [13]. □

Minimizing the Size of the Intersection

Using Property 1 with a given zonotope and a ZTS yields a family of zonotopes $v(k) \oplus T(k)\mathbf{B}^r,\ k = 0, \ldots, r$, that contain the intersection of the two. The selection of k should be made such that the resulting zonotope is the least conservative one among all possible choices; a reasonable method for this selection is to choose the integer k^* that yields the minimum volume overbounding zonotope:

$$k^* = \arg \min_{0 \leq k \leq r} \mathrm{Vol}(v(k) \oplus T(k)\mathbf{B}^r) \tag{5.11}$$

where $\mathrm{Vol}(v(k) \oplus T(k)\mathbf{B}^r)$ is the volume of the zonotope $v(k) \oplus T(k)\mathbf{B}^r$.
It is known (see [13, 14]) that the volume of a zonotope $v(k) \oplus T(k)\mathbf{B}^r \subset \mathbb{R}^n$ is given by:

$$\mathrm{Vol}(v(k) \oplus T(k)\mathbf{B}^r) = \sum_{i=1}^{M(n,r)} 2^n |\det[T_{s_1(i)}(k)\ \ldots\ T_{s_n(i)}(k)]| \tag{5.12}$$

where $T_i(k)$ is the i-th column of $T(k)$, $M(n, r)$ is the number of different ways of selecting n elements from a set containing r elements, and the integers $s_1(i), \ldots, s_n(i)$ represent the i-th choice among these $M(n, r)$ choices, with $i = 1, \ldots, M(n, r)$. Although (5.12) gives the exact volume of the zonotope, computing the volumes of the whole zonotope family $v(k) \oplus T(k)\mathbf{B}^r,\ k = 0, \ldots, r$ creates a heavy computational burden and is thus undesirable, especially when the values of n or r are large. To avoid these burdensome computations, an estimate for the zonotope volume can be used, as proposed by [13]:

$$\mathrm{Vol}(v(k) \oplus T(k)\mathbf{B}^r) = \det(T(k)T(k)^T) \tag{5.13}$$

In [13] it is reported that this method is computationally advantageous since it involves the calculation of only $r+1$ determinants (whereas (5.12) would require $(r+1) \times M(n, r)$ such calculations). It is further noted that even though the selected minimum volume zonotopes will be different from the ones that would be obtained if (5.12) was used, the zonotope volume estimate (5.13) strikes a good balance between computational effort and conservatism.

Basic Zonotopic Update Algorithm

The procedure of basic ZU using the method of [13] given in Property 1 is stated by Algorithm 9:

Algorithm 9 Basic zonotopic update

0) Repeat for each output j, $j = 1, \dots, n_y$.
1) At time step t, use the regressor vector $\varphi(t)$ and the measurement $\tilde{y}_j(t)$ to build the MS $\mathcal{M}_j(t) = \{H_j \in \mathbb{R}^{n_u m} \ : \ |\varphi(t)^T H_j - \tilde{y}_j(t)| \leq \epsilon_{d_j} + \epsilon_{v_j}\}$;
2) Using the MSZ $\mathcal{G}_j(t-1)$ and the MS $\mathcal{M}_j(t)$, build the ZTS $\mathcal{T}_j^{\mathrm{Z}}(t)$ as given in 5.2;
3) For $\mathcal{G}_j(t-1)$ and $\mathcal{T}_j^{\mathrm{Z}}(t)$, calculate the family of overbounding zonotopes $v(k) \oplus T(k)\mathbf{B}^r,\ k = 1, \dots, r$ using Property 1;
4) Calculate the volume of each member of this family using either (5.12) or (5.13);
5) Update the MSZ $\mathcal{G}_j(t-1)$ to $\mathcal{G}_j(t)$ by selecting the minimum volume overbounding zonotope $v(k^*) \oplus T(k^*)\mathbf{B}^r$ as $\mathcal{G}_j(t)$, i.e., $\mathcal{G}_j(t) = v(k^*) \oplus T(k^*)\mathbf{B}^r$.

The basic ZU algorithm is computationally very advantageous, as it does not involve heavy computations, e.g. solving optimization problems, but only requires some simple algebraic calculations and finding $r+1$ determinants. It also benefits from the advantages of zonotopes as model sets, in the sense that there is no need for removing redundant faces or computing the nominal model. A downside of the basic ZU is that the property of MSZs being non-expanding, i.e., $\mathcal{G}_j(t+1) \subseteq \mathcal{G}_j(t),\ j = 1, \dots, n_y$, is lost, since the intersection procedure obtains overbounding zonotopes. Thus, recursive feasibility and output constraint satisfaction cannot be guaranteed, which leads to the requirement of having to use MPC formulations with soft output constraints. Furthermore, it suffers from excessive conservatism due to a fundamental defect of the method given in Property 1 that is related to zonotopes collapsing into parallelotopes after some number of updates when certain conditions are met. The mechanism of this collapse is given in Appendix C for completeness.

5.3 Improved Zonotopic Update

The improved ZU is a step forward from the basic ZU method given in Section 5.2. This method also calculates a family of overbounding zonotopes containing the intersection of a given strip $\mathcal{M}$ with a given zonotope $\mathcal{G}$, but an intermediate step is introduced to reduce the conservatism present in the basic ZU. This section is based on the work of Chai and coworkers in [15].
The improved ZU differs from the basic ZU only in the method for calculating the family of overbounding zonotopes, thus only the property related to this calculation will be presented here as Property 2, which is a modification of Property 1.

Property 2. Given an n-zonotope $\mathcal{G}$ of order r

$$\mathcal{G} = \{H \in \mathbb{R}^n \ : \ H = w \oplus Z\mathbf{B}^r\} \tag{5.14}$$

and the ZTS $\mathcal{T}^{\mathrm{Z}}$ (obtained as shown in 5.2)

$$\mathcal{T}^{\mathrm{Z}} = \{H \in \mathbb{R}^n \ : \ |\varphi^T H - \eta| \leq \sigma\}, \tag{5.15}$$

obtain the minimal volume r-orthotope $q \oplus L\mathbf{B}^r$ that bounds the intersection of the strip given as

$$\mathcal{R} = \{f \in \mathbb{R}^r \ : \ |\varphi^T Z f - (\eta - \varphi^T w)| \leq \sigma\} \tag{5.16}$$

and the hypercube $\mathbf{B}^r$. Here $q \in \mathbb{R}^r$ is a vector defining the center of the orthotope $q \oplus L\mathbf{B}^r$, whereas $L \in \mathbb{R}^{r\times r}$ is a diagonal matrix defining its shape, with the elements on its diagonal given as $L_{ii},\ i = 1, \ldots, r$. Then, the family of zonotopes $\bar{v}(k) \oplus \bar{T}(k)\mathbf{B}^r$, parameterized with the integer $k,\ 0 \leq k \leq r$, contains the intersection of $\mathcal{G}$ and $\mathcal{T}^{\mathrm{Z}}$, i.e.:

$$\mathcal{G} \cap \mathcal{T}^{\mathrm{Z}} \subseteq \bar{v}(k) \oplus \bar{T}(k)\mathbf{B}^r, \quad 0 \leq k \leq r \tag{5.17}$$

where

$$\bar{v}(k) = \begin{cases} \bar{w} + \left(\frac{\eta - \varphi^T \bar{w}}{\varphi^T Z_k}\right) Z_k, & \text{if } 1 \leq k \leq r \text{ and } \varphi^T Z_k \neq 0 \\ w, & \text{otherwise,} \end{cases}$$

where $\bar{w}$ is defined as $\bar{w} = w + Zq$, and

$$\bar{T}(k) = \begin{cases} [\bar{T}_1^k\ \bar{T}_2^k\ \ldots\ \bar{T}_r^k], & \text{if } 1 \leq k \leq r \text{ and } \varphi^T Z_k \neq 0 \\ Z, & \text{otherwise} \end{cases}$$

$$\bar{T}_i^k = \begin{cases} L_{ii}\left(Z_i - \left(\frac{\varphi^T Z_i}{\varphi^T Z_k}\right) Z_k\right), & \text{if } i \neq k \\ \left(\frac{\sigma}{\varphi^T Z_k}\right) Z_k, & \text{if } i = k. \end{cases}$$

Proof. See [15]. □

Improved Zonotopic Update Algorithm

The procedure of improved ZU using the method of [15] given in Property 2 is summarized in Algorithm 10:

Algorithm 10 Improved zonotopic update

0) Repeat for each output j, $j = 1, \ldots, n_y$.
1) At time step t, use the regressor vector $\varphi(t)$ and the measurement $\tilde{y}_j(t)$ to build the MS $\mathcal{M}_j(t) = \{H_j \in \mathbb{R}^{n_u m} \ : \ |\varphi(t)^T H_j - \tilde{y}_j(t)| \leq \epsilon_{d_j} + \epsilon_{v_j}\}$;
2) Using the MSZ $\mathcal{G}_j(t-1)$ and the MS $\mathcal{M}_j(t)$, build the ZTS $\mathcal{T}_j^{\mathrm{Z}}(t)$ as given in 5.2;
3) For $\mathcal{G}_j(t-1)$ and $\mathcal{T}_j^{\mathrm{Z}}(t)$, calculate the family of overbounding zonotopes $\bar{v}(k) \oplus \bar{T}(k)\mathbf{B}^r,\ k = 1, \ldots, r$ using Property 2;
4) Calculate the volume of each member of this family using either (5.12) or (5.13);
5) Update the MSZ $\mathcal{G}_j(t-1)$ to $\mathcal{G}_j(t)$ by selecting the minimum volume overbounding zonotope $\bar{v}(k^*) \oplus \bar{T}(k^*)\mathbf{B}^r$ as $\mathcal{G}_j(t)$, i.e., $\mathcal{G}_j(t) = \bar{v}(k^*) \oplus \bar{T}(k^*)\mathbf{B}^r$.

Due to the additional computation required for $q \oplus L\mathbf{B}^r$, the improved ZU algorithm has more computational burden than its basic counterpart given in Algorithm 9, but it is also less conservative (see §5 in [15]), providing a tighter family of overbounding zonotopes containing the intersection of $\mathcal{G}_j(t-1)$ and $\mathcal{M}_j(t)$. It shares the order preserving property of the basic ZU and naturally has the general advantages of zonotopes over polytopes for representing model sets, having no need to remove redundant faces or compute nominal model. The improved ZU thus represents a computationally advantageous alternative to polytopic methods, while at the same time decreasing the conservatism present in the basic ZU. A downside of the improved ZU is that, similar to the case with the basic ZU, the property of MSZs being non-expanding, i.e., $\mathcal{F}_j(t+1) \subseteq \mathcal{F}_j(t), j = 1, \ldots, n_y$, is lost,

since the intersection procedure again obtains overbounding zonotopes, although with less conservatism. Thus, recursive feasibility and output constraint satisfaction cannot be guaranteed, which leads to the requirement of having to use MPC formulations with soft output constraints. Furthermore, it also suffers from excessive conservatism, due to the same fundamental defect plaguing the basic ZU method, related to zonotopes collapsing into parallelotopes. The mechanism of this defect is given in Appendix C for completeness.

5.4 Zonotopic Set Inflation

The Zonotopic Set Inflation (ZSI) is the zonotopic counterpart of the PSI method (given in Section 4.1) for handling plants with slowly varying dynamics. The idea here is also to forget past information simply by inflating the zonotope $w \oplus Z\mathbf{B}^r$ through increasing the length of the column vectors of Z. Since a zonotope is the Minkowski sum of the line segments defined by these column vectors, increasing the length of these vectors will increase the size of the zonotope.
Consider the MSZ at time t:

$$\mathcal{G}_j(t) = \{H_j \in \mathbb{R}^{n_u m} \ : \ H_j = w_j(t) \oplus Z_j(t)\mathbf{B}^r\} \tag{5.18}$$

where the matrix $Z_j(t)$ is as follows

$$Z_j(t) = [Z_{j,1}(t) \ Z_{j,2}(t) \ \ldots \ Z_{j,r}(t)], \tag{5.19}$$

with $Z_{j,i}(t), \ i = 1, \ldots, r$ denoting the columns of $Z_j(t)$. The length of these columns can be increased as follows

$$\hat{Z}_{j,i}(t) = \left(1 + \frac{\Omega_{\text{ZSI}}}{\|Z_{j,i}(t)\|_2}\right) Z_{j,i}(t), \ i = 1, \ldots, r, \tag{5.20}$$

where the ZSI factor Ω_{ZSI} is a design parameter to be chosen greater than 0. The inflated version of $Z_j(t)$ can then be written as follows

$$\hat{Z}_j(t) = [\hat{Z}_{j,1}(t) \ \hat{Z}_{j,2}(t) \ \ldots \ \hat{Z}_{j,r}(t)], \tag{5.21}$$

and the inflated MSZ then becomes

$$\hat{\mathcal{G}}_j(t) = \{H_j \in \mathbb{R}^{n_u m} \ : \ w_j(t) \oplus \hat{Z}_j(t)\mathbf{B}^r\}. \tag{5.22}$$

The ZSI method makes it possible for zonotopic methods to handle systems with slowly varying dynamics. The downside is that the property of MSZs being non-expanding, i.e., $\mathcal{G}_j(t+1) \subseteq \mathcal{G}_j(t), j = 1, \ldots, n_y$, is lost with the ZSI method, thus recursive feasibility and output constraint satisfaction cannot be guaranteed, which leads to the requirement of having to use MPC formulations with soft output constraints.

5.5 Zonotopic Center Tracking

The Zonotopic Center Tracking (ZCT) is the zonotopic counterpart of the PCT method (given in Section 4.2) for handling plants with rapidly varying dynamics. It is also assumed in the ZCT method that if there is an abrupt change in plant dynamics, the intersection of the MS $\mathcal{M}_j(t)$ and the MSZ $\mathcal{G}_j(t-1)$ will be an empty set:

$$\mathcal{M}_j(t) \cap \mathcal{G}_j(t-1) = \varnothing, \tag{5.23}$$

which implies that an update will lead to an empty model set, and thus it should be avoided. The way to avoid this is to translate $\mathcal{G}_j(t-1)$ in accordance with $\mathcal{M}_j(t)$, such that the intersection of $\mathcal{M}_j(t)$ with the translated $\mathcal{G}_j(t-1)$ will not be an empty set. Consider again the median hyperplane of the MS $\mathcal{M}_j(t)$:

$$\mathcal{M}_j^{\mathrm{m}}(t) = \{H_j \in \mathbb{R}^{n_u m} \ : \ \varphi(t)^T H_j = \tilde{y}_j(t)\}. \tag{5.24}$$

The translation vector $V_{\mathrm{T}}^{\mathrm{Z}}(t)$ is the vector linking the zonotope center $w_j(t-1)$ with its projection on the median hyperplane $\mathcal{M}_j^{\mathrm{m}}(t)$:

$$V_{\mathrm{T}}^{\mathrm{Z}}(t) = \varphi(t)\frac{\tilde{y}_j(t) - \varphi(t)^T w_j(t-1)}{\varphi(t)^T\varphi(t)}. \tag{5.25}$$

Considering the MSZ $\mathcal{G}_j(t-1)$ in the following form

$$\mathcal{G}_j(t-1) = \{H_j \in \mathbb{R}^{n_u m} \ : \ w_j(t-1) \oplus Z_j(t-1)\mathbf{B}^r\}, \tag{5.26}$$

the ZCT operation can be given as the following translation of the MSZ

$$\bar{\mathcal{G}}_j(t-1) = \{H_j \in \mathbb{R}^{n_u m} \ : \ \bar{w}_j(t-1) \oplus Z_j(t-1)\mathbf{B}^r\}, \tag{5.27}$$

where $\bar{w}_j(t-1)$ denotes the translated version of $w_j(t-1)$

$$\bar{w}_j(t-1) = w_j(t-1) + V_{\mathrm{T}}^{\mathrm{Z}}(t). \tag{5.28}$$

The ZCT method makes it possible for zonotopic methods to handle systems with rapidly varying dynamics. The downside is that, similar to the case with the ZSI method, the property of MSZs being non-expanding, i.e., $\mathcal{G}_j(t+1) \subseteq \mathcal{G}_j(t), j = 1, \ldots, n_y$, is lost with the ZCT method. Recursive feasibility and output constraint satisfaction can thus no longer be guaranteed, which leads to the requirement of having to use MPC formulations with soft output constraints.

5.6 Zonotopic Update Algorithm for Time-varying Systems

This section consolidates the ZSI and ZCT methods, presented in the previous sections of this chapter, into an algorithm that is able to handle time-varying systems with both slow and rapid variations in their dynamics.

Algorithm 11 Zonotopic update for time-varying systems

0) Repeat for each output j, $j = 1, \ldots, n_y$.
1) At time step t, use the regressor vector $\varphi(t)$ and the measurement $\tilde{y}_j(t)$ to build the MS $\mathcal{M}_j(t) = \{H_j \in \mathbb{R}^{n_u m} \ : \ |\varphi(t)^T H_j - \tilde{y}_j(t)| \leq \epsilon_{d_j} + \epsilon_{v_j}\}$;
2) Build the ZTS $\mathcal{T}_j^{\mathrm{Z}}$ as given in 5.2 using the MSZ $\mathcal{G}_j(t-1)$ and the MS $\mathcal{M}_j(t)$;
3) If the ZTS is an empty set, conduct ZCT as given in (5.27), inflate the resulting MSZ using (5.22) with $\Omega_{\mathrm{ZSI}} = \Omega_{\mathrm{ZE}}$, and switch PSI method on, otherwise go to 4);
4) If the ZTS has been empty less than N_{ZCT} time steps ago, conduct ZCT as given in (5.27) and inflate the resulting MSZ using (5.22) with $\Omega_{\mathrm{ZSI}} = \Omega_{\mathrm{TZE}}$, otherwise go to 5);
5) If ZSI method is switched on, inflate the MSZ using (5.22) with $\Omega_{\mathrm{ZSI}} = \Omega_{\mathrm{DZI}}$ and continue with updating the MSZ with the MS using a ZU method of choice.

Algorithm 11 is the zonotopic counterpart of Algorithm 8, thus the explanations for the steps of the latter, given in Section 4.3, are also valid for Algorithm 11 presented here.

Chapter 6

Simulation Results

6.1 Quadruple-Tank Process

Simulation studies conducted for the thesis use a nonlinear model of the QTP, hence this section is devoted to the study of its dynamics. The section is based on the work of Johansson in [16], where the QTP was first introduced.
Designed with the aim of emphasizing limitations in performance of multivariable control systems due to zero location, the QTP is built through interconnecting four liquid tanks and supplying them from a reservoir, with the flow generated by two pumps and distributed via two valves. A schematic diagram of the QTP is shown in Figure 6.1.

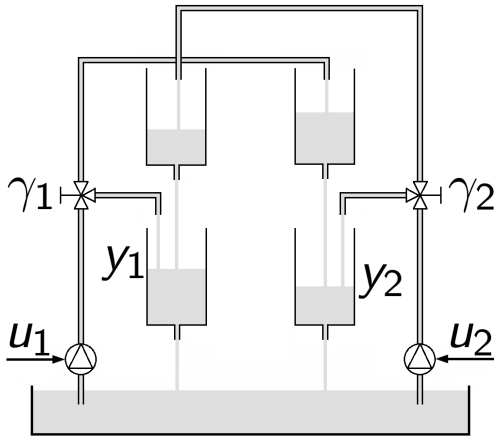

Figure 6.1: Schematic diagram of the QTP.

The control inputs are the voltages applied to the two pumps (v_1 and v_2), whereas the process outputs are the liquid levels in the lower tanks (h_1 and h_2). Using Bernoulli's law and mass balances, liquid level dynamics of the four tanks can be written as follows:

$$\begin{aligned}
\dot{h}_1 &= -\frac{s_1}{S_1}\sqrt{2gh_1} + \frac{s_3}{S_1}\sqrt{2gh_3} + \frac{\gamma_1 k_1}{S_1}v_1 \\
\dot{h}_2 &= -\frac{s_2}{S_2}\sqrt{2gh_2} + \frac{s_4}{S_2}\sqrt{2gh_4} + \frac{\gamma_2 k_2}{S_2}v_2 \\
\dot{h}_3 &= -\frac{s_3}{S_3}\sqrt{2gh_3} + \frac{(1-\gamma_2)k_2}{S_3}v_2 \\
\dot{h}_4 &= -\frac{s_4}{S_4}\sqrt{2gh_4} + \frac{(1-\gamma_1)k_1}{S_4}v_1,
\end{aligned} \tag{6.1}$$

where

S_i cross-section of tank i;

s_i cross-section of the outlet hole of tank i;

h_i liquid level in tank i,

with $i = 1, 2, 3, 4$. The parameters k_1 and k_2 are the pump flow constants, with $k_1 v_1$ and $k_2 v_2$ denoting then the flows generated by pump 1 and 2, respectively. Acceleration due to gravity is denoted by g. The valve coefficients γ_1,$\gamma_2 \in (0, 1)$ specify the positions of the valves and therefore determine how the flows from the pumps are distributed among the four tanks:

$$\begin{aligned} q_1 &= \gamma_1 k_1 v_1 \\ q_2 &= \gamma_2 k_2 v_2 \\ q_3 &= (1-\gamma_2) k_2 v_2 \\ q_4 &= (1-\gamma_1) k_1 v_1 \end{aligned} \tag{6.2}$$

with q_i denoting the flow to the tank i. Denoting the deviations of the liquid levels and pump voltages from their steady state values as $x_i = h_i - \bar{h}_i$ and $u_i = v_i - \bar{v}_i$, respectively, the linearized model of the process in state-space can be written as:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} \frac{-1}{T_1} & 0 & \frac{S_3}{S_1 T_3} & 0 \\ 0 & \frac{-1}{T_2} & 0 & \frac{S_4}{S_2 T_4} \\ 0 & 0 & \frac{-1}{T_3} & 0 \\ 0 & 0 & 0 & \frac{-1}{T_4} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} \frac{\gamma_1 k_1}{S_1} & 0 \\ 0 & \frac{\gamma_2 k_2}{S_2} \\ 0 & \frac{(1-\gamma_2) k_2}{S_3} \\ \frac{(1-\gamma_1) k_1}{S_4} & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{6.3}$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} k_c & 0 & 0 & 0 \\ 0 & k_c & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}, \tag{6.4}$$

where k_c is the sensor constant and the time constants T_i $(i = 1, 2, 3, 4)$ are:

$$T_i = \frac{A_i}{a_i} \sqrt{\frac{2\bar{h}_i}{g}}. \tag{6.5}$$

The transfer function matrix is:

$$G(s) = \begin{bmatrix} \frac{\gamma_1 c_1}{T_1 s + 1} & \frac{(1-\gamma_2) c_1}{(T_3 s + 1)(T_1 s + 1)} \\ \frac{(1-\gamma_1) c_2}{(T_4 s + 1)(T_2 s + 1)} & \frac{\gamma_2 c_2}{T_2 s + 1} \end{bmatrix} \tag{6.6}$$

where $c_1 = T_1 k_1 k_c / S_1$ and $c_2 = T_2 k_2 k_c / S_2$. The zeros of $G(s)$ are the zeros of the numerator polynomial of determinant of $G(s)$:

$$\det G(s) = \frac{c_1 c_2}{\gamma_1 \gamma_2 \prod_{i=1}^{4} (T_i s + 1)} \left[(T_3 s + 1)(T_4 s + 1) - \frac{(1-\gamma_1)(1-\gamma_2)}{\gamma_1 \gamma_2} \right]. \tag{6.7}$$

For $\gamma_1, \gamma_2 \in (0, 1)$ there exists two finite zeros of the transfer matrix $G(s)$, with one invariably located in the left half-plane while the other may switch half-planes with varying γ_1 and γ_2. This follows from the following root-locus argument that uses a parameter $\eta \in (0, \infty)$ defined as

$$\eta \triangleq \frac{(1-\gamma_1)(1-\gamma_2)}{\gamma_1 \gamma_2}. \tag{6.8}$$

For η tending to 0 the zeros approach the values of $-1/T_3$ and $-1/T_4$, whereas for η tending to ∞ one zero goes to $-\infty$ while the other goes to $+\infty$. One zero is at the origin for $\eta = 1$, which implies $\gamma_1 + \gamma_2 = 1$. Then it follows that the plant is nonminimum phase for

$$0 < \gamma_1 + \gamma_2 < 1 \tag{6.9}$$

and minimum phase for

$$1 < \gamma_1 + \gamma_2 < 2. \tag{6.10}$$

The QTP presents a difficult control problem, having both nonlinear characteristics given in Equation (6.1), and nonminimum phase behaviour for certain valve coefficient configurations. It is thus a challenging testbed for adaptive control algorithms [3].
Simulation experiments presented in the following sections use the simulation configuration given in Table 6.1.

Table 6.1: Simulation configuration.

S[cm^2]	s[cm^2]	$k_1 = k_2$[cm^3/sV]	k_c	$\bar{h}_1$[cm]	$\bar{h}_2$[cm]	$\bar{v}_1$[V]	$\bar{v}_2$[V]	g[cm/s^2]
15.52	0.178	3.3	1	11.7	10.8	8	8	981

For all simulation experiments, the simulation length (in discrete time steps) is selected as $T_{\text{op}} = 250$, whereas the sampling time is chosen as $T_{\text{s}} = 8\ s$ (see [17] for the reasoning). For the sensitivity analyses presented in the chapter, valve constants of the QTP are fixed at $\gamma_1 = 0.35$ and $\gamma_2 = 0.33$, and the simulation scenario (in the sense of the shape of the desired output references) given in Figure 6.2 is used for each individual simulation.

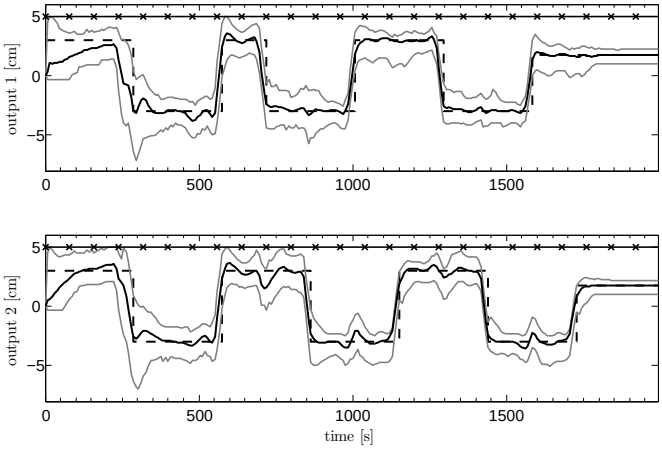

Figure 6.2: Simulation scenario used in the sensitivity analyses.

6.2 Performance Measures

In the simulation based sensitivity analyses, for both polytopic and zonotopic methods, the performance measures of Integral Square Error (ISE) and Integral Output Uncertainty Interval (IOUI) are employed. The ISE is the usual measure for quantifying tracking performance and it is defined for the output j as follows:

$$\text{ISE}_j \triangleq \sum_{t=0}^{T_{\text{op}}} \hat{e}_j^2(t),\ j = 1, \ldots, n_y. \tag{6.11}$$

IOUI, on the other hand, is related to the identification performance of the SMI engine. For IOUI we first need to compute the maximal and minimal values of predicted (at time step t) plant outputs by solving the following LP

$$\begin{aligned} \bar{y}_j(t) &= \underset{A_j(t)H_j \leq b_j(t)}{\text{maximize}} \; H_j^T \varphi(t), \\ \underline{y}_j(t) &= \underset{A_j(t)H_j \leq b_j(t)}{\text{minimize}} \; H_j^T \varphi(t), \; j = 1, \ldots, n_y, \end{aligned} \tag{6.12}$$

where $\bar{y}_j(t)$ and $\underline{y}_j(t)$ are the maximal and minimal values of the predicted plant outputs, respectively. The difference between these two specifies the output uncertainty interval, which the adaptive MPC algorithm tries to gradually decrease through real-time SMI. The IOUI can then be defined, for the output j, as the sum of the output uncertainty intervals over operation length:

$$\text{IOUI}_j \triangleq \sum_{t=0}^{T_{\text{op}}} (\bar{y}_j(t) - \underline{y}_j(t)), \; j = 1, \ldots, n_y. \tag{6.13}$$

Note that the IOUI measure directly relates to how strongly the SMI engine is able decrease the model set size during operation, since the output uncertainty interval will be large for large model sets.

In the sensitivity analyses a single scalar value for each performance measure is required to be able to easily compare different simulation experiment instances, thus the ISE and IOUI measures are defined as the sums of their respective values over the set of all plant outputs:

$$\text{ISE} \triangleq \sum_{j=1}^{n_y} \text{ISE}_j, \quad \text{IOUI} \triangleq \sum_{j=1}^{n_y} \text{IOUI}_j. \tag{6.14}$$

Furthermore, the performance measure for quantifying computational effort, the Total Computation Time (TCT), is chosen simply as the total time it takes in seconds for the simulation experiment to finish execution.

For polytopic methods the measure of Total Number of Faces (TNF) is defined for the output j as follows:

$$\text{TNF}_j \triangleq \sum_{t=0}^{T_{\text{op}}} (r_{A_j}(t) + r_{D_j}(t)), \; j = 1, \ldots, n_y. \tag{6.15}$$

whereas for a single simulation experiment, the TNF is defined as

$$\text{TNF} \triangleq \sum_{j=1}^{n_y} \text{TNF}_j. \tag{6.16}$$

For zonotopic methods the measure of Total Zonotope Order (TZO) is defined for the output j as follows:

$$\text{TZO}_j \triangleq \sum_{t=0}^{T_{\text{op}}} r_j(t), \; j = 1, \ldots, n_y, \tag{6.17}$$

whereas for a single simulation experiment, the TZO is defined as

$$\text{TZO} \triangleq \sum_{j=1}^{n_y} \text{TZO}_j. \tag{6.18}$$

6.3 Sensitivity Analyses of Face Filtering Methods

6.3.1 Sensitivity Analysis of FFPU

Variation of control performance with changing Γ_{A_j} parameter (given in (3.10)) of the FFPU method given in Algorithm 4 is analyzed here through a series of simulations conducted using the nonlinear model of the QTP given in Equation (6.1) with the model configuration given in Table 6.1. Controller parameters are selected as given in Table 6.2.

Table 6.2: Controller configuration for Γ_A sensitivity analysis.

$\bar{u}$	$\underline{u}$	$\Delta\bar{u}$	$\bar{y}$	ϵ_d	ϵ_v	L	μ	ρ	m	α	N	M_A	M_D
3.6	-3.6	7.2	5	0.35	0.25	1.8	2	0.78	12	0.01	17	120	0

For the 2 outputs of the QTP model the same M_A and Γ_{A_j} values are used, i.e. $M_{A_1} = M_{A_2} = M_A$ and $\Gamma_{A_1} = \Gamma_{A_2} = \Gamma_A$. In the simulations Γ_A values are varied from 0.45 to 1 in steps of 0.01, corresponding to a total of 56 individual simulation experiments. The results are consolidated in Figure 6.4, which shows the normalized values of ISE, IOUI, TCT, and TNF for the 56 simulations plotted against Γ_A. Normalizations are made via dividing each sequence by its maximum value.

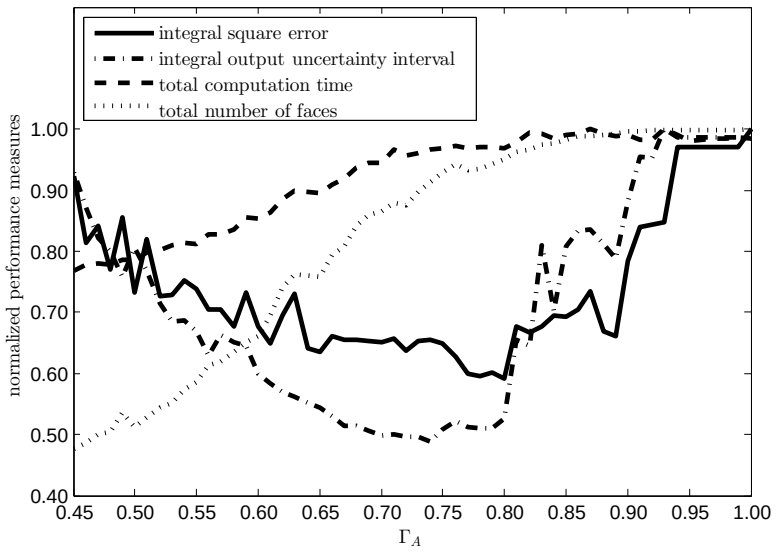

Figure 6.3: Performance sensitivity to Γ_A in FFPU.

Figure 6.3 clearly shows the effect of the Γ_A parameter on closed-loop operation: Smaller values of Γ_A indicate more stringent filtering, in the sense that during an SMI update the current MS $\mathcal{M}_j(t)$ has to contain more information (it has to be cutting a larger portion of the polytope) in order to be added to the MSP for a smaller Γ_A value. This in turn results in fewer faces being added to the MSPs, decreasing in general the TNF of the MSPs and in turn also the TCT, for the price of a degrading performance, as seen in increasing ISE and IOUI. This trend (decreasing computation for the price of degrading performance) is especially pronounced for Γ_A values between 0.45 and 0.75. On the other hand, in the interval $\Gamma_A^{\mathrm{m}} \in [0.75, 1]$, Γ_A^{m} values around 1 roughly correspond to the case without any face filtering, and they naturally have the worst performances (i.e., the performance

measures are around 1). With a Γ_A equal to 0.8, for example, tracking performance (i.e., ISE) and identification performance (i.e., IOUI) can be improved around $40-50\%$ with regards to the case with $\Gamma_A = 1$. This points to substantial improvements in control performance with the use of FFPU without paying any price in increased computational effort. In conclusion, real-time SMI can be improved significantly using the face filtering method, and the Γ_A parameter of the FFPU algorithm 4 specifies a tuning knob for the tradeoff between performance and computation. Although it is possible, through careful design of Γ_A, to yield substantial gains in performance without increasing computational effort (e.g., with $\Gamma_A = 0.78$), the FFPU method is in general superior to the PU given in Algorithm 2: Even for Γ_A values between 0.45 and 0.75 there are improvements in both control performance and reductions in computational effort at the same time compared to the case without using any face filtering (i.e., $\Gamma_A = 1$).

6.3.2 Sensitivity Analysis of Self-tuning FFPU

To analyze how control performance varies with the $\Gamma^{\mathrm{m}}_{A_j}$ parameter of the self-tuning FFPU method given in Algorithm 5, a series of simulations has been conducted using the nonlinear model of the QTP given in Equation (6.1) with the model configuration given in Table 6.1. Controller parameters are selected as given in Table 6.2, except that here the $\Gamma^{\mathrm{m}}_{A_j}$ and $\Gamma^{\mathrm{a}}_{A_j}$ parameters (given in (3.11)) of the self-tuning FFPU are set as follows: The $\Gamma^{\mathrm{m}}_{A_j}$ parameter is varied from 0.45 to 1 in steps of 0.01, whereas a constant value of 0.15 is chosen for the $\Gamma^{\mathrm{a}}_{A_j}$ parameter, corresponding to a total of 56 individual simulation experiments. For the 2 outputs of the QTP model the same M_A, $\Gamma^{\mathrm{m}}_{A_j}$, and $\Gamma^{\mathrm{a}}_{A_j}$ values are used, i.e., $M_{A_1} = M_{A_2} = M_A$, $\Gamma^{\mathrm{m}}_{A_1} = \Gamma^{\mathrm{m}}_{A_2} = \Gamma^{\mathrm{m}}_{A}$, and $\Gamma^{\mathrm{a}}_{A_1} = \Gamma^{\mathrm{a}}_{A_2} = 0.15$. The results are consolidated in Figure 6.4, which shows the normalized values of ISE, IOUI, TCT, and TNF for the 56 simulations plotted against Γ^{m}_{A}. Normalizations are made by dividing each sequence to its maximum value.

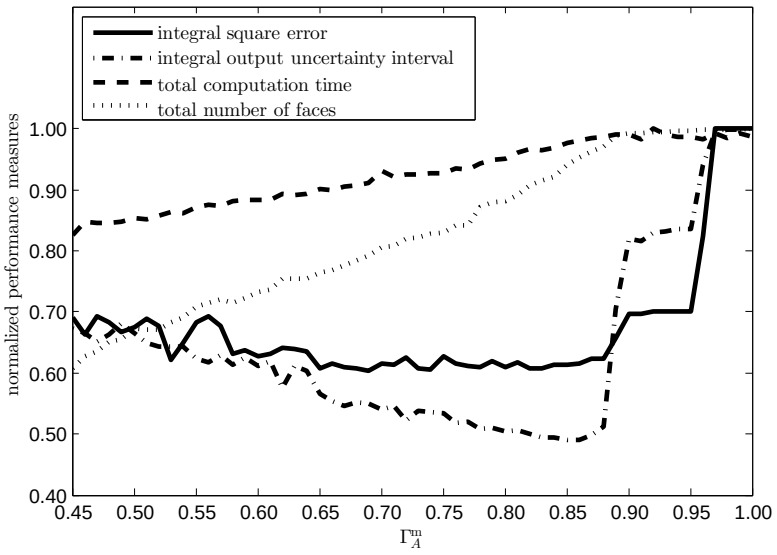

Figure 6.4: Performance sensitivity to Γ^{m}_{A} in self-tuning FFPU.

Comments similar to those given for Figure 6.3 can be made for Figure 6.4 as well: The trend of improving performance for the price of increased computation is pronounced here for Γ^{m}_{A} values between 0.45 and 0.85. Γ^{m}_{A} values around 1 (i.e., those without almost

no filtering) again have the worst performances, expectedly. A Γ_A^{m} equal to 0.85 yields around $40 - 50\%$ improvements in tracking performance (i.e., ISE) and identification performance (i.e., IOUI) without any increase in computational effort, with regards to the case with $\Gamma_A^{\mathrm{m}} = 1$. This again points to the capability of the self-tuning FFPU algorithm in substantially improving control performance. In conclusion, it is also possible to improve real-time SMI via the self-tuning face filtering method, and the Γ_A^{m} parameter of the self-tuning FFPU algorithm 5 specifies a tuning knob for the tradeoff between performance and computation. Through careful design of Γ_A^{m} it is possible to yield substantial gains in performance without increasing computational effort (e.g., with $\Gamma_A = 0.86$), but the self-tuning FFPU method is also in general superior to the PU given in Algorithm 2: Even for Γ_A^{m} values between 0.45 and 0.85 there are improvements in both control performance and reductions in computational effort at the same time compared to the case without using any face filtering (i.e., $\Gamma_A = 1$).

6.3.3 Comparison of FFPU and Self-tuning FFPU

Sensitivity analysis comparison of the two face filtering methods contributed by the thesis, namely the FFPU and the self-tuning FFPU, given in Algorithms 4 and 5, respectively, is presented here. The comparison, shown in Figure 6.5, is made via simply consolidating the two previous analyses shown in Figures 6.3 and 6.4, but the normalizations of the plots given here are made such that each plot shows a single performance measure sequence for the two algorithms, divided by the maximum value of the two, and the measures are plotted against Γ_A values for the FFPU and Γ_A^{m} for the self-tuning FFPU.

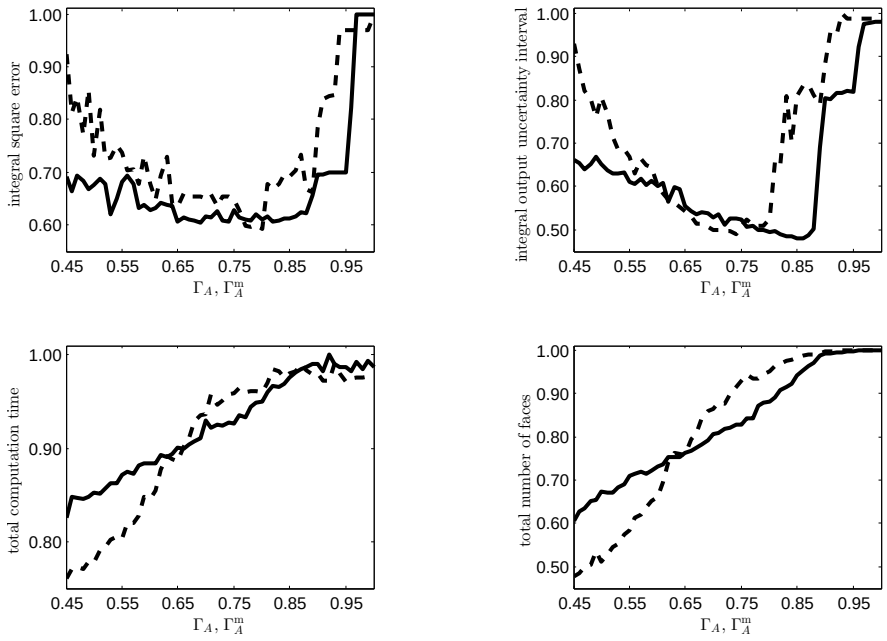

Figure 6.5: Comparison of FFPU (dashed lines) and self-tuning FFPU (solid lines).

Figure 6.5 shows that the self-tuning FFPU method is an improvement over the FFPU: With a self-tuning face filter it is possible to improve tracking performance (i.e., ISE) around 10% and identification performance (i.e., IOUI) around $20 - 30\%$, for some intervals of the face filter parameters. There are no meaningful differences between the computational efforts and polytope face usages of the two methods, however, as the plots

for TCT and TNF show. Thus it can be concluded that the self-tuning FFPU presents a slight improvement over the FFPU, while both methods clearly specify substantial improvements over the PU method given in Algorithm 2 (which again roughly corresponds to the case with $\Gamma_A = 1$ or $\Gamma_A^{\mathrm{m}} = 1$ in Figure 6.5).

6.4 Sensitivity Analysis of Gradual Learning BCPU

For the analysis of the performance improvement obtained by using the GL-BCPU method given in Algorithm 6, a series of simulations has been conducted using the nonlinear model of the QTP given in Equation (6.1). In the simulations M_A is decreased from 120 to 48 while M_D is increased from 0 to 72, both in steps of 2, corresponding to a total of 37 individual simulation experiments. Thus, the total face number limit $M_A + M_D$ is constant for all simulations, whereas the ratio M_D/M_A increases from 0 to 1.5. For the 2 outputs of the process model the same M_A and M_D values are used, i.e., $M_{A_1} = M_{A_2}$ and $M_{D_1} = M_{D_2}$. The simulations are conducted under the configurations given in Tables 6.1 and 6.2, except that for the FFPU phase of the GL-BCPU algorithm a default face filter with constant a parameter of $\Gamma_A = 1$ is used, whereas the GL phase employs an error function based self-tuning face filter with the parameters $\Gamma_D^{\mathrm{m}} = 0.8$ and $\Gamma_D^{\mathrm{a}} = 0.15$. The results are consolidated in Figure 6.6, which shows the normalized values of ISE, IOUI, TCT, and TNF for the 37 simulations plotted against M_D/M_A. Normalizations are made by dividing each sequence to its maximum value.

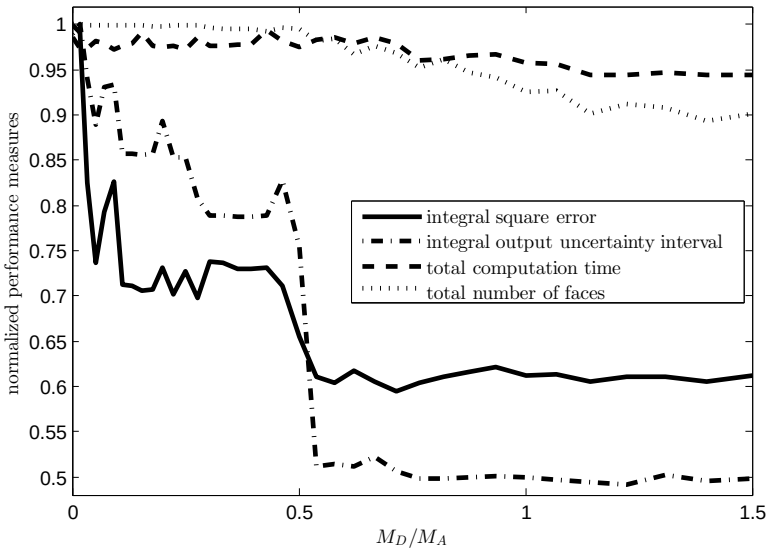

Figure 6.6: Performance sensitivity to M_D/M_A ratio in GL-BCPU.

The figure clearly shows that, using the GL-BCPU method, it is also possible to improve control performance without increasing computational effort. M_D/M_A values around 0 correspond to the case without any learning, since no portion of the face number limit is dedicated to learning (i.e., $M_D = 0$). For values between 0.5 and 1.5, it can be seen that both tracking and identification performance can be improved around $40 - 50\%$ with regards to the case with $M_D/M_A = 0$ without any increase in computational effort. It is even possible, for M_D/M_A close to 1.5, to yield small (around 5%) reductions in computation time while still retaining the $40 - 50\%$ improvement in control performance. In conclusion, real-time SMI can be improved significantly using the GL-BCPU method,

as showcased by the improvements in control performance obtained without increasing computational effort.

6.5 Comparison of Basic ZU and Improved ZU

This section compares the two order preserving ZU methods, namely the basic ZU and the improved ZU, given in Algorithms 9 and 10, respectively. The comparison investigates how the control performance changes with varying initial zonotope order $r(0)$, via a series of simulations conducted using the nonlinear model of the QTP given in Equation (6.1) with the model configuration given in Table 6.1. Controller parameters are selected as given in Table 6.2, except that there are no M_A and M_D parameters defined for zonotopes but instead the initial zonotope order $r(0)$ is varied from 24 to 48 in steps of 1, corresponding to a total of 25 individual simulation experiments, with $r(0) = 24$ corresponding to the case with a parallelotope as the MSZ. For the 2 outputs of the QTP model the same $r(0)$ value is used, i.e., $r_1(0) = r_2(0) = r(0)$. The results are consolidated in Figure 6.7, which shows the normalized values of ISE, IOUI and TCT for the 25 simulations plotted against $r(0)$, alongside final and average TZO values. Normalizations are made such that each plot shows a single performance measure sequence for the two algorithms, divided by the maximum value of the two, except the lower right plot showing the average values of $r(T_{\text{op}})$ (i.e., $(r_1(T_{\text{op}}) + r_2(T_{\text{op}}))/2$) and TZO (i.e., $\text{TZO}/2T_{\text{op}}$), which are not normalized with the intention to show the exact zonotope order values. The distinction between zonotope order $r(t)$ and its initial value $r(0)$ is important here, since the order preserving ZU methods analyzed here have a fundamental defect related to zonotopes collapsing into parallelotopes (as described in Appendix C). The $r(t)$ value tends to decrease over time as the zonotopes gradually collapse into parallelotopes (i.e., as the column vectors of the matrices Z_j defining the zonotope shape gradually degrade into zero vectors).

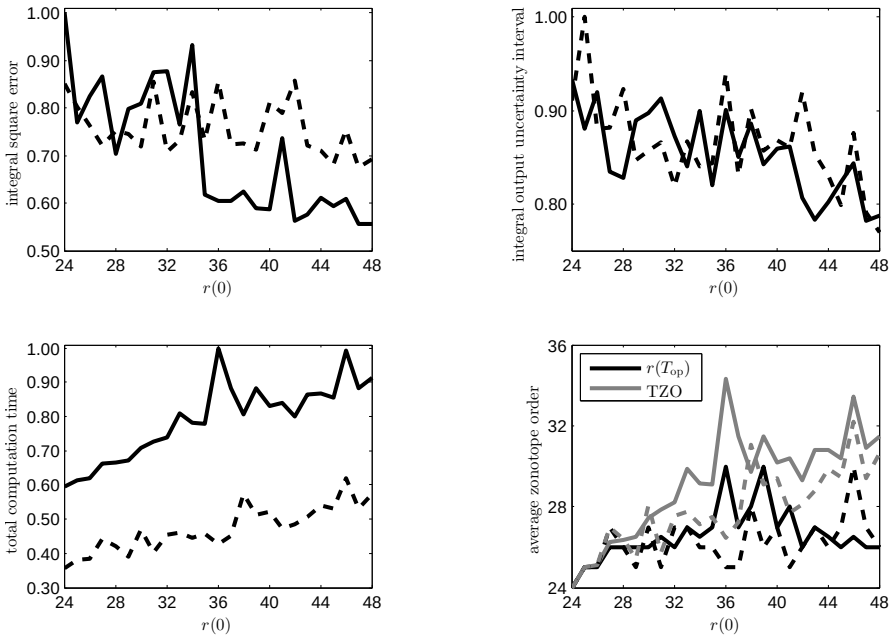

Figure 6.7: Comparison of basic ZU (dashed lines) and improved ZU (solid lines).

Figure 6.7 shows that the improved ZU is able to slightly improve tracking performance over the basic ZU, as the plot for ISE suggests, especially for $r(0)$ values in the interval

[36, 48]. This can be seen as evidence that the improved ZU is in fact less conservative than the basic ZU, although without substantial difference. The two methods have no meaningful difference in identification performance, as the plot for IOUI shows. Both measures expectedly show a trend of decrease with increasing $r(0)$, the price of which is paid with increased computational effort, as seen in the plot for TCT. The TCT plot is also interesting in showing the substantial difference in computational effort, as the improved ZU appears to need around 60% more computation time than the basic ZU. This stems from the intermediary step in the improved ZU needed for computing the r-orthotope $q \oplus L\mathbf{B}^r$, which decreases the conservatism of the method but increases in turn the computational effort. Finally, the average zonotope order plot provides a good example of how the zonotope collapse mechanism manifests itself in practice: The average $r(T_{\text{op}})$ values are close to $r = 24$ (i.e., the r value specifying a parallelotope), whereas the average TZO values are substantially different from 24. This implies that the zonotopes are indeed collapsing into parallelotopes, and the collapse happens over time as this difference suggests. In conclusion, although the improved ZU has a slightly higher tracking performance than the basic ZU, the fact that it is around 60% more burdensome computationally suggests in general that it is not a substantially better ZU method. Also, both methods suffer from the defect related to the zonotope collapse mechanism given in Appendix C.

6.6 Operations with Time-Varying Systems

Capabilities of the polytopic and zonotopic algorithms for handling time-varying systems, proposed with Algorithms 8 and 11, are showcased in this section. A scenario with a nonlinear QTP model with time-varying parameters is used for the simulations. In this scenario the valve constants γ_1 and γ_2 of the process model given in Equation (6.1) are varied in time as given in Figure 6.8.

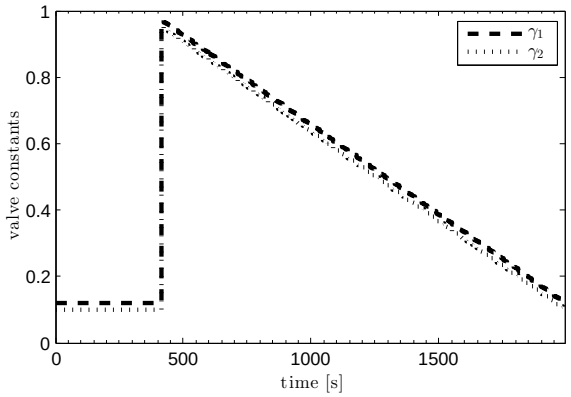

Figure 6.8: Variation of valve constants γ_1 and γ_2 with time.

The simulation scenario given with Figure 6.8 specifies severe changes in plant dynamics in the form of both jumps and drifts in the parameters, the severity coming from the fact that the QTP model is nonminimum phase for $0 < \gamma_1 + \gamma_2 < 1$ and minimum phase for $1 < \gamma_1 + \gamma_2 < 2$, as discussed at the end of Section 6.1. Thus, it provides good testing conditions for the algorithms to display their capabilities at handling time-varying systems.

6.6.1 Polytopic Operation with Time-Varying Systems

Performance of the polytopic methods for time-varying systems is illustrated here using the simulation results for the scenario given in Figure 6.8. Algorithm 8 is employed as the real-time SMI engine of the adaptive MPC controller, with the FFPU method with $\Gamma_A = 1$ as the PU method of choice. Configurations for the simulation and the controller are the same as the ones given in Tables 6.1 and 6.2, except that here the face number limits are selected as $M_{A_1} = M_{A_2} = 200$, and the valve constants γ_1 and γ_2 are varying in time. Parameters of the polytopic method related to the handling of time-varying systems are given in Table 6.3, whereas the results of the simulation are shown in Figure 6.9.

Table 6.3: Parameters of the polytopic method for operation with time-varying systems.

N_{PCT}	Ω_{PE}	Ω_{TPE}	Ω_{DPI}
8	1.4	1.2	1.11

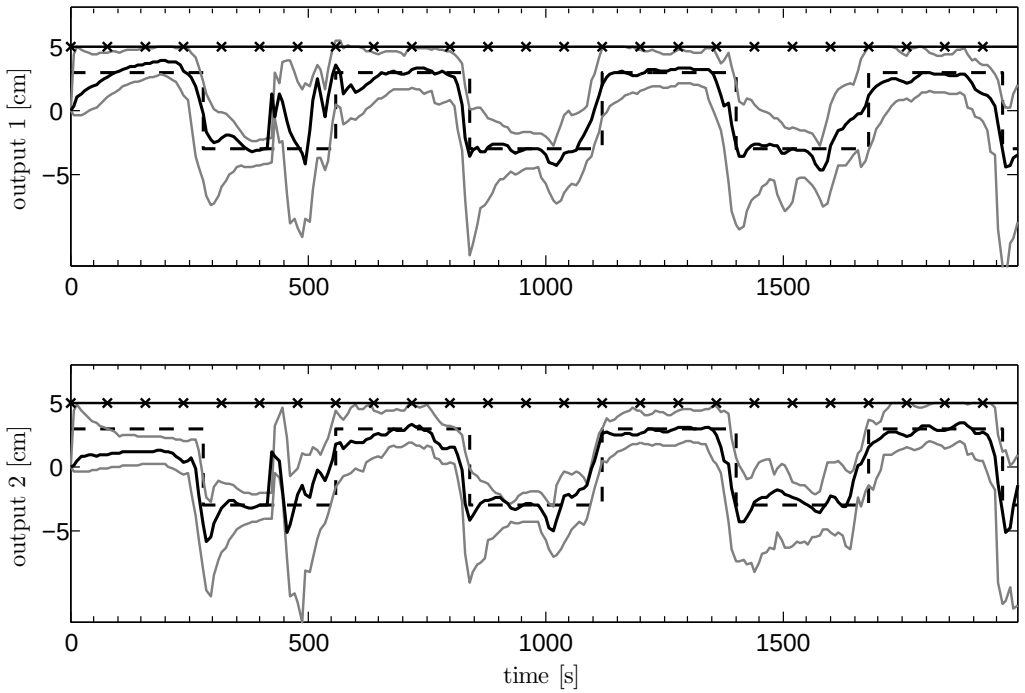

Figure 6.9: Simulation results obtained by applying the adaptive MPC algorithm (Algorithm 1), with Algorithm 8 as the SMI engine, to the nonlinear QTP model with valve constants varying in time as given in Figure 6.8. Comparison of desired output reference $y_{\mathrm{des}}(t)$ (dashed black lines) with the measured plant output $\tilde{y}(t)$ (solid black lines) is shown for outputs $y_1(t)$ (upper plot) and $y_2(t)$ (lower plot). Lower and upper bounds (solid gray) of output uncertainty intervals are also shown, alongside the constraints on plant outputs (solid black lines with $\times$).

Figure 6.9 makes it clear that the polytopic method for handling time-varying systems, given in Algorithm 8, is capable of dealing with both slowly and rapidly varying plant dynamics, even under severe conditions the valve constant variations given in Figure 6.8 specify. Considering that a reasonable choice of sampling time for the QTP is 8 seconds, the method is also practicable as the average computation time per step is around 3.06 seconds.

6.6.2 Zonotopic Operation with Time-Varying Systems

Performance of the zonotopic methods for time-varying systems is illustrated here using the simulation results for the scenario given in Figure 6.8. Algorithm 11 is employed as the real-time SMI engine of the adaptive MPC controller, with the improved ZU method as the ZU method of choice. Configurations for the simulation and the controller are the same as the ones given in Tables 6.1 and 6.2, except that there are no M_A and M_D parameters defined for zonotopes but instead the zonotope order r is selected as 30, and the valve constants γ_1 and γ_2 are varying in time. Parameters of the zonotopic method related to the handling of time-varying systems are given in Table 6.4, whereas the results of the simulation are shown in Figure 6.10.

Table 6.4: Parameters of the zonotopic method for operation with time-varying systems.

N_{ZCT}	Ω_{ZE}	Ω_{TZE}	Ω_{DZI}
8	0.024	0.006	0.00286

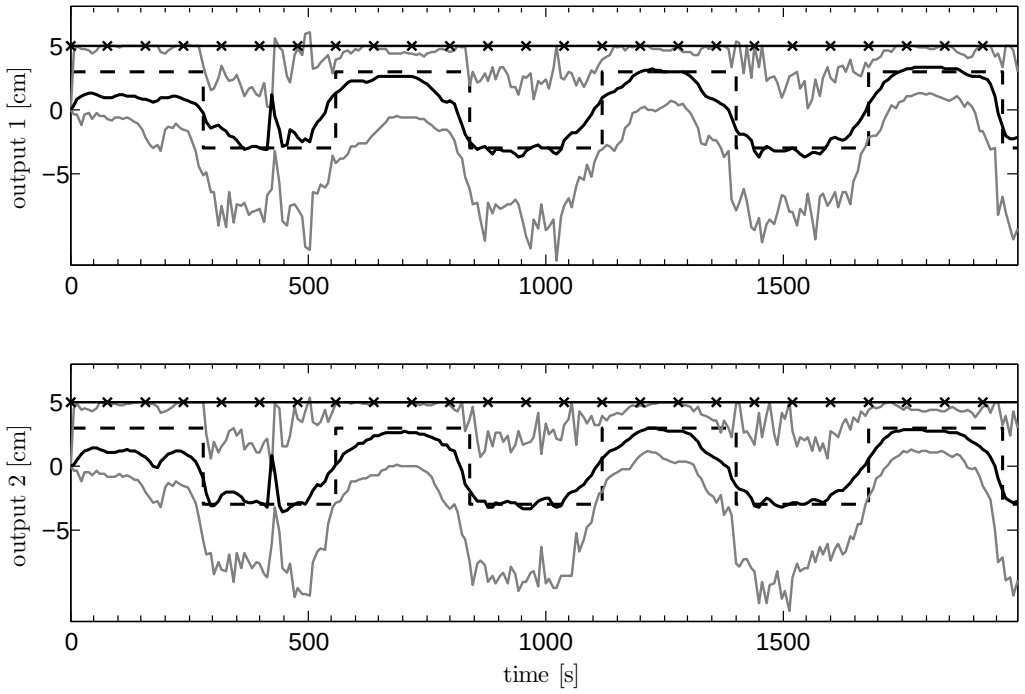

Figure 6.10: Simulation results obtained by applying the adaptive MPC algorithm (Algorithm 1), with Algorithm 11 as the SMI engine, to the nonlinear QTP model with valve constants varying in time as given in Figure 6.8. Comparison of desired output reference $y_{\mathrm{des}}(t)$ (dashed black lines) with the measured plant output $\tilde{y}(t)$ (solid black lines) is shown for outputs $y_1(t)$ (upper plot) and $y_2(t)$ (lower plot). Lower and upper bounds (solid gray) of output uncertainty intervals are also shown, alongside the constraints on plant outputs (solid black lines with $\times$).

Figure 6.10 makes it clear that the zonotopic method for handling time-varying systems, given in Algorithm 11, is capable of dealing with both slowly and rapidly varying plant dynamics, even under severe conditions the valve constant variations given in Figure 6.8 specify. It should be noted that the zonotopic operation is noticeably more conservative than its polytopic counterpart, as can be seen from the differences in the size of output

uncertainty intervals between Figures 6.9 and 6.10. This stems from the fact that the order preserving ZU methods have excessive conservatism, owing largely to their fundamental defect about zonotope collapse explained in Appendix C. On the other hand, the polytopic operation shown in Figure 6.9 had an average computation time per step of 3.06 seconds, whereas the zonotopic operation in Figure 6.10 has the same at 0.48 seconds. Zonotopic methods thus appear to be a viable and computationally highly advantageous, although somewhat conservative, alternative to polytopic SMI.

Chapter 7

Conclusion

This thesis focused on improving the real-time SMI engine of the recently proposed adaptive MPC algorithm of [3] for constrained linear MIMO systems, and developed methods for extending the algorithm to time-varying systems.

Firstly, real-time polytopic SMI has been improved with two main methods, namely the face filtering method for both PU and BCPU, and the learning method for BCPU. Face filtering method bases the decision about conducting polytopic updates on the information content of measurements, improving thus the quality of SMI and leading to higher control performance. The face filter itself is also improved with self-tuning capability. For bounded complexity methods the GL and BL methods are proposed, which tune the BCPU method to plant dynamics by learning regressor vectors in real-time, leading to better BCPU style SMI opportunities, thus also to improved control performance.

Extensions of polytopic SMI methods are proposed as the second part of the contributions of the thesis. The PSI method adds the capability of dealing with slowly varying plant dynamics, whereas the PCT method is for countering rapid variations. An operational PU algorithm is constructed, that consolidates the PSI and PCT methods, and thus enables the adaptive MPC algorithm to deal with both slow and rapid variations in plant dynamics.

Zonotopic SMI methods are explored as the third part, which represent model sets as zonotopes, a special class of convex polytopes. These methods specify a computationally advantageous alternative to polytopes. Two recent order preserving ZU methods from the literature are studied and applied to the adaptive MPC algorithm. Furthermore, zonotopic counterparts of the polytopic methods for dealing with time-varying systems are proposed. It is observed that although the investigated order preserving zonotopic SMI methods are excessively conservative (owing largely to their fundamental defects about zonotope collapse shown in Appendix C), these methods still represent a viable and computationally advantageous competition to polytopic methods as they are able to operate around an order of magnitude faster than their polytopic counterparts.

For future work, methods that can tune the parameters of the algorithms for handling time-varying systems can be considered. Investigating less conservative ZU methods might be of interest, since zonotopic methods appear to be computationally highly advantageous. Also, it may be possible to obtain extremely fast operation by combining zonotopic SMI and model parametrizations with orthonormal basis functions given in §5.1 of [3].

Bibliography

[1] K.J. Åström and B. Wittenmark. *Adaptive Control*. Addison-Wesley, 1995.

[2] I.D. Landau, R. Lozano, M. M'Saad, and A. Karimi. *Adaptive Control: Algorithms, Analysis and Applications*. Springer, New York, 2011.

[3] M. Tanaskovic, L. Fagiano, R. Smith, and M. Morari. Adaptive receding horizon control for constrained MIMO systems. *Automatica*, 2013.

[4] M. Tanaskovic, L. Fagiano, R. Smith, P. Goulart, and M. Morari. Adaptive model predictive control for constrained linear systems. In *European Control Conference*, Zurich, Switzerland, July 2013.

[5] M. Tanaskovic, L. Fagiano, R. Smith, and M. Morari. Adaptive model predictive control for constrained mimo systems. In *11th IFAC International Workshop on Adaptation and Learning in Control and Signal Processing*, July 2013.

[6] M. Tanaskovic, L. Fagiano, R.S. Smith, and M. Morari. Adaptive model predictive control with exploring property for constrained linear systems that uses basis function model parametrization. *CoRR*, abs/1309.5304, 2013.

[7] T.H. Mattheiss. An algorithm for determining irrelevant constraints and all vertices in system of linear inequalities. *Operations Research*, 21:247–260, 1973.

[8] S.M. Veres, H. Messaoud, and J.P. Norton. Limited-complexity model-unfalsifying adaptive tracking-control. *International Journal of Control*, 72:1417–1426, 1999.

[9] S. Maraoui and H. Messaoud. Design and comparative study of limited complexity bounding error identification algorithms. In *IFAC Symposium on System Structure and Control*, Prague, Czech Republic, 2001.

[10] S.M. Veres and J.P. Norton. Predictive self-tuning control by parameter bounding and worst-case design. *Automatica*, 29(4):911–928, 1993.

[11] H. Piet-Lahanier and E. Walter. Bounded-error tracking of time-varying parameters. *IEEE Transactions on Automatic Control*, 39:1661–1664, 1994.

[12] C. Combastel. A state bounding observer based on zonotopes. In *European Control Conference*, January 2003.

[13] J.M. Bravo, T. Alamo, and E.F. Camacho. Bounded error identification of systems with time-varying parameters. *IEEE Transactions on Automatic Control*, 51(7):1144–1150, July 2006.

[14] H. L. Montgomery. Computing the volume of a zonotope. *The American Mathematical Monthly*, 96:431, 1989.

[15] W. Chai, X. Sun, and J. Qiao. Improved zonotopic method to set membership identification for systems with time-varying parameters. *IET Control Theory and Applications*, 5:2039–2044, 2011.

[16] K.H. Johansson. The quadruple-tank process: a multivariable laboratory process with an adjustable zero. *IEEE Transactions on Control Systems Technology*, 8(3):456–465, May 2000.

[17] Lawrence Minnetian. Implementation of adaptive model predictive control on the quad tank system. Semester project, IfA, August 2013.

[18] J. Löfberg. YALMIP : A toolbox for modeling and optimization in MATLAB. In *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

Appendix A

Algorithm for Removing Redundant Inequalities

The procedure for removing redundant faces from a polytope, given in [7], is described here. Consider the polytope, the redundant inequalities of which are to be removed, that consists of the set of inequalities defined by the matrix A and vector b:

$$Ax \leq b. \tag{A.1}$$

Denoting the inequality to be checked for redundancy as follows

$$\alpha^T x \leq \beta, \tag{A.2}$$

A and b can be written as

$$A = \begin{bmatrix} A' \\ \alpha^T \end{bmatrix} \quad b = \begin{bmatrix} b' \\ \beta \end{bmatrix}. \tag{A.3}$$

The following LP can then be solved to detect the redundancy of the inequality A.2:

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & \alpha^T x \\ \text{subject to} \quad & A'x \leq b' \\ & \alpha^T x \leq \beta + 1. \end{aligned} \tag{A.4}$$

If it holds for the optimal value $f^* = \alpha^T x^*$ of the above LP that $f^* \leq \beta$, then A.2 is redundant and should be removed. Through solving the LP given above and repeating the procedure for all inequalities constituting A.1, the set of non-redundant inequalities describing the polytope can be found.

Appendix B

Nominal Model Computation

Polytopic model set updates require the selection of a nominal model from the model set after the update for the use of the controller. A reasonable method involves inscribing the maximum volume 2-norm ball inside the MSP $\mathcal{F}_j(t)$ and selecting the center of this ball to be the nominal model. An LP can be solved for this, however the solution is not unique in general; as remedy, the change in nominal model (the displacement of the new one from the previous one) can be penalized via a regularization term, which yields the following LP [3]:

$$\begin{aligned} \underset{\xi_j(t), H_{c,j}(t)}{\text{maximize}} \quad & \sum_{j=1}^{n_y} \xi_j(t) - \alpha \left\| H_{c,j}(t-1) - H_{c,j}(t) \right\|_1 \\ \text{subject to} \quad & a_{ji}(t) H_{c,j}(t) + \xi_j(t) \left\| a_{ji}(t) \right\|_2 \leq b_{ji}(t) \\ & \forall j = 1, \ldots, n_y, \ \forall i = 1, \ldots, n_u, \end{aligned} \tag{B.1}$$

where $\xi_j(t)$ is the radius of the maximum volume 2-norm ball inscribed inside the MSP $\mathcal{F}_j(t)$ with the nominal model $H_{c,j}(t) \in \mathbb{R}^{n_u m}$ as the center of this ball, $\alpha > 0$ is a design variable specifying the weight on the regularization term, $a_{ji}(t)$ is the i-th row of the matrix $A_j(t)$, and $b_{ji}(t)$ is the i-th row of the vector $b_j(t)$. Provided that the LP (B.1) is feasible, the solution $H_{c,j}(t)$ is selected as the nominal model of the j-th MSP. Infeasibility, on the other hand, would suggest that the j-th MSP is an empty set and that the data gathered from the plant invalidate the Assumptions 1 and 2.

Appendix C

Mechanism of Zonotope Collapse

C.1 Mechanism of Zonotope Collapse in Basic ZU

The zonotopic update method of Bravo and coworkers, proposed in [13] and described in this thesis in Section 5.2 with the name of basic ZU, suffers from a fundamental defect in the calculation of the family of overbounding zonotopes given in Property 1. The defect results from the MSZ $\mathcal{G}$ collapsing into a parallelotope over time as the real-time zonotopic SMI progresses. The mechanism of this collapse can be explained as follows: The shape of the MSZ $\mathcal{G}$ is defined by the matrix $Z \in \mathbb{R}^{n\times r}$. During the calculation of the family of overbounding zonotopes $v(k) \oplus T(k)\mathbf{B}^r, k = 0, \ldots, r$, given in Property 1, the columns T_i^k of the matrices $T(k)$ defining the members of the family are calculated using Z as follows

$$T_i^k = \begin{cases} Z_i - \left(\frac{\varphi^T Z_i}{\varphi^T Z_k}\right) Z_k, & \text{if } i \neq k \\ \left(\frac{\sigma}{\varphi^T Z_k}\right) Z_k, & \text{if } i = k. \end{cases} \tag{C.1}$$

Then the minimum volume member of the family, $v(k^*) \oplus T(k^*)$, is selected as the new MSZ. This results in turn that the column $T_{k^*}^{k^*}$ of $T(k^*)$ of the new MSZ is parallel to the column $Z_{j_k^*}$ of Z_j of the old MSZ, whereas all the other columns $T_k^{k^*}, k = 0, \ldots, r, k \neq k^*$ become perpendicular to the regressor vector φ, since

$$\varphi^T Z_i - \left(\frac{\varphi^T Z_i}{\varphi^T Z_k}\right) \varphi^T Z_k = 0. \tag{C.2}$$

For an n-zonotope of order r, with $r > n$, this implies that after the start of real-time zonotopic SMI, at a time step t when n unique integers $k, k = 0, \ldots, r$ has been selected as k^* (discounting occurrences of $k^* = 0$, since these imply that no update actually took place, i.e., $\mathcal{G}(t) = \mathcal{G}(t-1)$), $r-n$ of the columns of Z of the MSZ will become perpendicular to n regressor vectors that occurred during the last n time steps. If these regressor vectors form a basis for $\mathbb{R}^n$ (which is not unlikely to happen in practice) this will result in those $r - n$ columns getting reduced to zero vectors, since only the zero vector is perpendicular to all the vectors forming a basis for $\mathbb{R}^n$. When this reduction to zero vectors happens, the MSZ is left with n vectors as columns of Z that are not zero vectors. An n-zonotope $\mathcal{G} = \{H_j \ : \ H = w \oplus Z\mathbf{B}^r\}$ of order r with $r - n$ zero vectors as columns of Z specifies a parallelotope and cannot return to being a zonotope, since if a column vector Z_i is a zero vector then the corresponding column vector T_i^k of a member of the overbounding zonotope family calculated with (C.1) will also be a zero vector, regardless of what the vectors Z_k and φ are. Thus we say that the zonotope has collapsed into a parallelotope. Examples 1 and 2 illustrate the collapse mechanism of the basic ZU method.

Example 1. (*Symbolical example for zonotope collapse*)
Consider the MSZ at time $t = 0$, given as a 3-zonotope of order 4 as $\mathcal{G}(0) = w(0) \oplus Z(0)\mathbf{B}^4$. $Z(0)$ has thus 4 columns:

$$Z(0) = [Z_1(0)\ Z_2(0)\ Z_3(0)\ Z_4(0)]. \tag{C.3}$$

Consider that at time $t = 1$ a regressor vector $\varphi(1)$ occurs and k^* is selected as 1, i.e., $k^*(1) = 1$. This means that, due to the calculations given in Property 1, the first column $Z_1(1)$ of $Z(1)$ $\mathcal{G}(1)$ will be parallel to $Z_1(0)$:

$$Z_1(1) = \left(\frac{\sigma}{\varphi(1)^T Z_1(0)}\right) Z_1(0) \parallel Z_1(0), \tag{C.4}$$

whereas the other columns of $Z(1)$ become perpendicular to $\varphi(1)^T$:

$$Z_i(1) = Z_i(0) - \left(\frac{\varphi(1)^T Z_i(0)}{\varphi(1)^T Z_1(0)}\right) Z_1(0) \perp \varphi(1),\ i \in 2, 3, 4. \tag{C.5}$$

It may happen that an integer $k, k = 0, \ldots, r$ may be selected as k^* more than once for two different zonotopic updates. Thus let us consider that, at time $t = 2$, k^* is again selected to be 1, and also a regressor vector $\varphi(2)$ occurs. Then the first column $Z_1(2)$ of $Z(2)$ of $\mathcal{G}(2)$ will be parallel to $Z_1(1)$ and $Z_1(0)$:

$$Z_1(2) = \left(\frac{\sigma}{\varphi(2)^T Z_1(1)}\right) Z_1(1) \parallel Z_1(1) \parallel Z_1(0), \tag{C.6}$$

whereas the other columns of $Z(2)$ become perpendicular to $\varphi(2)$ this time:

$$Z_i(2) = Z_i(1) - \left(\frac{\varphi(2)^T Z_i(1)}{\varphi(2)^T Z_1(1)}\right) Z_1(1) \perp \varphi(2),\ i \in 2, 3, 4. \tag{C.7}$$

Note that now $Z_i(2), i \in 2, 3, 4$ are not forced to stay perpendicular to $\varphi(1)$ at time $t = 2$. Continue with considering time $t = 3$, when a previously unseen k value of 2 is selected as k^*, and a regressor vector $\varphi(3)$ occurs. Then the second column $Z_2(3)$ of $Z(3)$ of $\mathcal{G}(3)$ will be parallel to $Z_2(2)$, which we remember as becoming perpendicular to $\varphi(2)$ at time $t = 2$, thus $Z_2(3)$ will also be perpendicular to $\varphi(2)$:

$$Z_2(3) = \left(\frac{\sigma}{\varphi(3)^T Z_2(2)}\right) Z_2(2) \parallel Z_2(2) \perp \varphi(2), \quad Z_2(3) \perp \varphi(2). \tag{C.8}$$

The other columns of $Z(3)$ become perpendicular to $\varphi(3)$:

$$Z_i(3) = Z_i(2) - \left(\frac{\varphi(3)^T Z_i(2)}{\varphi(3)^T Z_2(2)}\right) Z_2(2) \perp \varphi(3),\ i \in 1, 3, 4, \tag{C.9}$$

where we note that $Z_i(2), i \in 2, 3, 4$ became perpendicular to $\varphi(2)$ at time $t = 2$. Since the previous equation obtains $Z_i(3), i \in 1, 3, 4$ by subtracting $Z_2(2)$, scaled by some value, from $Z_i(2), i \in 1, 3, 4$, it is also the case that $Z_i(3), i \in 3, 4$ are perpendicular to both $\varphi(2)$ and $\varphi(3)$. Note here that since being perpendicular to $\varphi(2)$ and $\varphi(3)$ at the same time (assuming that $\varphi(2)$ and $\varphi(3)$ are not parallel) in $\mathbb{R}^3$ means lying on the same line, the columns $Z_3(3)$ and $Z_4(3)$ of $Z(3)$ are parallel, and the zonotope $Z(3)$ has already

practically collapsed into a parallelotope, as an n-zonotope $\mathcal{G} = \{H \ : \ H = w \oplus Z\mathbf{B}^r\}$ of order r with $r - n + 1$ parallel vectors as columns of Z also specifies a parallelotope.
Consider finally time $t = 4$, when again a previously unseen k value of 4 is selected as k^*, and a regressor vector $\varphi(4)$ occurs. Now the fourth column $Z_4(4)$ of $Z(4)$ of $\mathcal{G}(4)$ will be parallel to $Z_4(3)$, which we remember as becoming perpendicular to $\varphi(2)$ at time $t = 2$ and to $\varphi(3)$ at time $t = 3$, thus $Z_4(4)$ will also be perpendicular to both $\varphi(2)$ and $\varphi(3)$:

$$Z_4(4) = \left(\frac{\sigma}{\varphi(4)^T Z_4(3)}\right) Z_4(3) \parallel Z_4(3), \quad Z_4(3) \perp \varphi(2), \varphi(3) \perp Z_4(4), \tag{C.10}$$

whereas the other columns of $Z(4)$ become perpendicular to $\varphi(4)$:

$$Z_i(4) = Z_i(3) - \left(\frac{\varphi(4)^T Z_i(3)}{\varphi(4)^T Z_4(3)}\right) Z_4(3) \perp \varphi(4), \ i \in 1, 2, 3, \tag{C.11}$$

where we note that $Z_i(3), i \in 1, 3, 4$ became perpendicular to $\varphi(3)$ at time $t = 3$. Since the previous equation obtains $Z_i(4), i \in 1, 2, 3$ by subtracting $Z_4(3)$, scaled by some value, from $Z_i(3), i \in 1, 2, 3$, it is also the case that $Z_i(4), i \in 1, 3$ are perpendicular to both $\varphi(3)$ and $\varphi(4)$. Specifically, the column $Z_3(4)$ was obtained by the previous equation via subtracting $Z_4(3)$, scaled by some value, from $Z_3(3)$. Since $Z_i(3), i \in 3, 4$ became perpendicular to both $\varphi(2)$ and $\varphi(3)$ at time $t = 3$, now at time $t = 4$ the column $Z_3(4)$ is perpendicular to $\varphi(2)$, $\varphi(3)$, and $\varphi(4)$ at the same time.
In conclusion, time $t = 4$ specifies the time step when $n = 3$ unique integers $k, k = 1, 2, 4$ has been selected as k^*. Here, $r - n = 1$ of the columns of $Z(4)$ of the MSZ $\mathcal{G}(4)$, namely $Z_3(4)$, becomes perpendicular to $n = 3$ regressor vectors that occurred during the last n time steps, namely $\varphi(2)$, $\varphi(3)$, and $\varphi(4)$. Assuming that these vectors form a basis for $\mathbb{R}^3$, the column $Z_3(4)$ is reduced to zero vector, since only the zero vector in $\mathbb{R}^3$ is perpendicular to all the vectors forming a basis for $\mathbb{R}^3$. When this happens the MSZ is left with $n = 3$ vectors as columns of $Z(4)$, namely $Z_i(4), i \in 1, 2, 4$, that are not zero vectors. Since a 3-zonotope of order 4 with $r - n = 1$ zero vectors as columns of Z_j specifies a parallelotope, we conclude that the initial 3-zonotope $\mathcal{G}(0)$ of order 4 has collapsed into a 3-parallelotope $\mathcal{G}(4)$ after 4 time steps.

Example 2. (*Numerical example for zonotope collapse*) (Figures are generated using YALMIP developed by Löfberg in [18].)
Consider the MSZ at time $t = 0$, shown in Figure C.1,

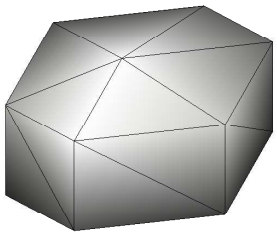

Figure C.1: Zonotope $\mathcal{G}(0)$ at time $t = 0$.

and given as a 3-zonotope of order 4 as $\mathcal{G}(0) = w(0) \oplus Z(0)\mathbf{B}^4$ with $w(0)$ and $Z(0)$ as follows:

$$w(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad Z(0) = \begin{bmatrix} 0.58 & 1 & 0 & 0 \\ 0.58 & 0 & 1 & 0 \\ 0.58 & 0 & 0 & 1 \end{bmatrix}. \tag{C.12}$$

Consider that at time $t = 1$ a regressor vector $\varphi(1)$ occurs as

$$\varphi(1) = \begin{bmatrix} 1 \\ 6 \\ 2 \end{bmatrix}, \tag{C.13}$$

and k^* is selected as 1, i.e., $k^*(1) = 1$. The basic ZU yields the MSZ $\mathcal{G}(1) = w(1) \oplus Z(1)\mathbf{B}^4$ with $w(1)$ and $Z(1)$ as follows:

$$w(1) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad Z(1) = \begin{bmatrix} 0.67 & 0.89 & -0.67 & -0.22 \\ 0.67 & -0.11 & 0.33 & -0.22 \\ 0.67 & -0.11 & -0.67 & 0.78 \end{bmatrix}. \tag{C.14}$$

The zonotope $\mathcal{G}(1)$ is shown in Figure C.2.

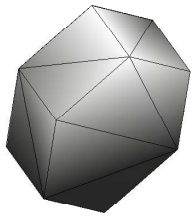

Figure C.2: Zonotope $\mathcal{G}(1)$ at time $t = 1$.

It may happen that an integer $k, k = 0, \ldots, r$ may be selected as k^* more than once for two different zonotopic updates. Thus let us consider that, at time $t = 2$, k^* is again selected to be 1. A regressor vector $\varphi(2)$ occurs as

$$\varphi(2) = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}. \tag{C.15}$$

The basic ZU yields the MSZ $\mathcal{G}(2) = w(2) \oplus Z(2)\mathbf{B}^4$ with $w(2)$ and $Z(2)$ as follows:

$$w(2) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad Z(2) = \begin{bmatrix} 0.56 & 0.89 & -0.33 & -0.56 \\ 0.56 & -0.11 & 0.67 & -0.56 \\ 0.56 & -0.11 & -0.33 & 0.44 \end{bmatrix}. \tag{C.16}$$

The zonotope $\mathcal{G}(2)$ is shown in Figure C.3.

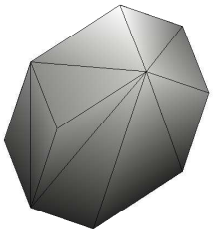

Figure C.3: Zonotope $\mathcal{G}(2)$ at time $t = 2$.

We continue with considering time $t = 3$, when a previously unseen k value of 4 is selected as k^*, and a regressor vector $\varphi(3)$ occurs as

$$\varphi(3) = \begin{bmatrix} 6 \\ 2 \\ 1 \end{bmatrix}. \tag{C.17}$$

The basic ZU yields the MSZ $\mathcal{G}(3) = w(3) \oplus Z(3)\mathbf{B}^4$ with $w(3)$ and $Z(3)$ as follows:

$$w(3) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad Z(3) = \begin{bmatrix} -0.33 & 0.89 & -0.16 & 0.16 \\ 0.67 & -0.11 & 0.64 & -0.64 \\ 0.67 & -0.11 & -0.36 & 0.36 \end{bmatrix}. \tag{C.18}$$

The zonotope $\mathcal{G}(3)$ is shown in Figure C.4.

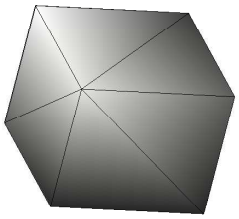

Figure C.4: Zonotope $\mathcal{G}(3)$ at time $t = 3$.

Note that with $\mathcal{G}(3)$ the MSZ has already practically collapsed into a parallelotope, as $Z_3(3)$ and $Z_4(3)$ are parallel.
Consider finally time $t = 4$, when again a previously unseen k value of 2 is selected as k^*, and a regressor vector $\varphi(4)$ occurs as

$$\varphi(4) = \begin{bmatrix} 3 \\ 0 \\ 6 \end{bmatrix}. \tag{C.19}$$

The basic ZU yields the MSZ $\mathcal{G}(4) = w(4) \oplus Z(4)\mathbf{B}^4$ with $w(4)$ and $Z(4)$ as follows:

$$w(4) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad Z(4) = \begin{bmatrix} -0.51 & 0.77 & 0 & 0.14 \\ 1.41 & 0.38 & 0 & -0.59 \\ 0.26 & -0.38 & 0 & 0.33 \end{bmatrix}. \tag{C.20}$$

The zonotope $\mathcal{G}(4)$ is shown in Figure C.5.

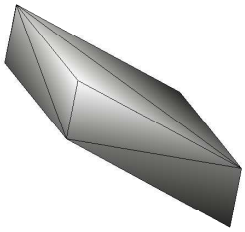

Figure C.5: Zonotope $\mathcal{G}(4)$ at time $t = 4$.

With $\mathcal{G}(4)$ the zonotope collapse is complete, as can be seen from the column vector $Z_3(4)$ of $Z(4)$ becoming a zero vector with the ZU update at time $t = 4$. We note that this happened at the time step where $n = 3$ unique integers, namely 1, 2, and 4, has been selected as k^*, and the condition that the regressor vectors occurred during the last n time steps, namely $\varphi(2)$, $\varphi(3)$, and $\varphi(4)$, form a basis for $\mathbb{R}^3$ is fulfilled:

$$\begin{vmatrix} 3 & 6 & 1 \\ 0 & 2 & 3 \\ 6 & 1 & 5 \end{vmatrix} = 117. \tag{C.21}$$

The collapse mechanism explains the excessive conservatism of the basic ZU method given in Algorithm 9, and specifies a fundamental defect of the zonotopic update method proposed in [13].

C.2 Mechanism of Zonotope Collapse in Improved ZU

The collapse mechanism of the zonotopic update method of Chai and coworkers, proposed in [15] and described in this thesis in Section 5.3 with the name of improved ZU, is the same as that of the basic ZU method. This is because the two methods differ in the calculation of the columns of the family of overbounding zonotopes only in the additional L_{ii} term present in the improved ZU method, but this term does not change anything regarding the zonotope collapse mechanism. This can clearly be seen if we rewrite Equation C.2 for the improved ZU:

$$L_{ii}\varphi^T Z_i - L_{ii}\left(\frac{\varphi^T Z_i}{\varphi^T Z_j}\right)\varphi^T Z_j = 0. \tag{C.22}$$