KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT INGENIEURSWETENSCHAPPEN
DEPARTEMENT ELEKTROTECHNIEK
Kasteelpark Arenberg 10, B-3001 Leuven (Heverlee)

ROBUST MODEL BASED PREDICTIVE CONTROL
– AN INVARIANT SET APPROACH –

Promotoren:
Prof. dr. ir. B. De Moor
Prof. dr. ir. J. Suykens

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de ingenieurswetenschappen

door

Bert PLUYMERS

May 2006

KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT INGENIEURSWETENSCHAPPEN
DEPARTEMENT ELEKTROTECHNIEK
Kasteelpark Arenberg 10, B-3001 Leuven (Heverlee)

ROBUST MODEL BASED PREDICTIVE CONTROL
– AN INVARIANT SET APPROACH –

Jury:
Prof. dr. ir. Y. Willems, voorzitter
Prof. dr. ir. B. De Moor, promotor
Prof. dr. ir. J. Suykens, promotor
Prof. dr. ir. J. Vandewalle
Prof. dr. ir. J. Van Impe
Prof. dr. ir. J. Swevers
Prof. dr. ir. J. Maciejowski

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de ingenieurswetenschappen

door

Bert PLUYMERS

U.D.C. 519.71

May 2006

ISBN 90-5682-715-4

U.D.C. 519.71

D/2006/7515/49

Dankwoord

In het voorjaar van 2001 ontving Prof. Suykens een mail van twee gemotiveerde, maar besluiteloze studenten met de vraag om zelf een thesisonderwerp te mogen voorstellen. Na wat zoeken werd het onderwerp modelgebaseerde predictieve controle – een term die ook terug te vinden is in de titel van deze thesis – en gingen Luc en ik aan de slag. Ongeveer één jaar, heel wat gezwoeg en gezweet, maar gelukkig toch ook enkele vreugdedansjes later konden we met enige trots een afgewerkte thesis voorleggen. Deze vormde een ideale springplank om een doctoraat te beginnen over hetzelfde onderwerp en daar heb ik tot op heden nog geen moment spijt van gehad.

Nu, vier jaar, nog wat meer gezwoeg en gezweet, maar toch ook weer enkele vreugdesprongen later ben ik de laatste hand aan het leggen aan mijn doctoraatsthesis en moet ik toegeven dat het opnieuw met enige trots is. Het moment is aangebroken om vooruit te durven kijken en nieuwe uitdagingen aan te gaan, maar tegelijkertijd zou ik ook willen terugkijken en alle mensen bedanken, die ervoor gezorgd hebben dat de laatste vier jaar zo'n unieke ervaring zijn geworden.

Vooreerst bedank ik het IWT-Vlaanderen (Instituut voor de Aanmoediging van Innovatie door Wetenschap en Technologie in Vlaanderen) dat mijn onderzoek financierde in de vorm van een specialisatiebeurs en het op die manier mogelijk maakte om 4 jaar voltijds aan onderzoek te doen.

Ik zou ook Prof. De Moor willen bedanken voor de kans die hij me gaf om binnen zijn onderzoeksgroep in alle vrijheid mijn doctoraatsonderzoek uit te voeren. Bart, je enthousiasme heeft me steeds gemotiveerd om de lat telkens een beetje hoger proberen te leggen. De zinsneden "Geeft er een lap op!" en "Make it happen!" zullen me altijd bijblijven; hopelijk blijven ze hun werk doen ...

Ook wil ik Prof. Suykens bedanken voor de vele besprekingen, die steeds in een informele sfeer verliepen. Johan, bedankt voor de talrijke raadgevingen, die steeds blijk gaven van een objectieve en kritische wetenschappelijke ingesteldheid en die me ook telkens terug wat afstand deden nemen om daarna met een betere doelgerichtheid verder te kunnen werken.

Daarnaast wil ik ook mijn twee assessoren, Prof. Vandewalle en Prof. Van Impe, danken voor hun waardevolle inbreng en feedback en de andere juryleden Prof. Swevers en Prof. Maciejowski (Cambridge Univ., UK) voor hun bereidwilligheid om in mijn jury te zetelen.

I would like to address a special word of gratitude to Prof. Maciejowski. Your work has had a significant influence on the research presented in this thesis. Thank you for your willingness to be a member of my Ph.D. jury and for making the effort to travel to Leuven; your presence is very much appreciated.

Vele resultaten in deze thesis werden bekomen in samenwerking met andere onderzoekers uit andere universiteiten, zonder wiens hulp dit doctoraat niet zou geweest zijn wat het nu is. Ik zou hen hiervoor speciaal willen bedanken.

I would like to thank Prof. Rossiter (Sheffield Univ., UK) for his cooperation, his patience and for the helpful discussions and insights. Anthony, thank you for pointing me the way to some of the research topics I've been working on. This has had a decisive influence on the topics discussed in this thesis and has proven to be very rewarding. I would also like to thank Lars Imslad (SINTEF, Norway) and Il-Seop Choi (Sheffield Univ., UK), with whom I came into contact via Anthony, for the enjoyable cooperation and the interesting results we obtained together.

A word of gratitude also goes to Prof. Kothare (Lehigh Univ., USA) and Zhaoyang Wan (Lehigh Univ., USA) for their constructive cooperation on our papers. Mayuresh, thanks for your openness to comments and your preparedness to listen. I have always been and still am impressed by your work and am honored to be able to defend our colors with our mutual paper at the next ACC.

Jeroen Buijs heeft ook, zowel rechtstreeks als onrechtstreeks, een bijdrage geleverd tot dit doctoraat onder meer door de erg goede begeleiding tijdens het eindwerk van Luc Roobrouck en mezelf. Ik zou samen met Jeroen ook Luc willen bedanken voor de goede sfeer waarin ons eindwerk verlopen is, wat me de motivatie gegeven heeft om in deze richting verder te gaan.

Zoals mijn collega's zullen beamen is een doctoraat geen *9 to 5* job; je neemt je onderzoek onvermijdelijk mee naar huis en dit vraagt daarom ook enig uithoudingsvermogen van de mensen waarmee je samenleeft, al zullen zij met plezier nog heel wat extra argumenten geven hiervoor. Ik ben jullie erg dankbaar voor het geduld en de steun.

In eerste instantie denk ik dan aan mijn ouders Willy en Monique, mijn zus, schoonbroer en neefje Wendie, Mario en Wout en mijn hele familie, voor de steun die ze me altijd geboden hebben en voor de bezorgdheid van sommigen van hen (of moet ik het 'angsten' noemen, Wendie?) wanneer ik weer eens moest vliegen.

Daarnaast denk ik aan mijn huisgenoten Gert, Wout, Dieter en Tinne, die regelmatig moeten aanhoren over welke wiskundige problemen ik weer aan het nadenken ben. Roomies, bedankt voor de feestjes die we al organiseerden en bedankt voor de ontelbare liters wijn die in de tussenperiodes vloeiden. Jullie hebben me hierdoor ongetwijfeld meer inspiratie gebracht dan dat ik er werktijd door ben verloren, ook al lijkt dat moeilijk. Ik hoop jullie op 23 mei evenveel inspiratie te kunnen teruggeven!

Natuurlijk vergeet ik ook niet mijn ex-huisgenoten, waarmee ik een huis deelde tijdens de eerste jaren van mijn doctoraat. Bedankt, Liesbeth, Koen, Marion en Kathleen, voor de steun en de babbels op de cruciale momenten.

Ik denk ook aan mijn vrienden uit Westerlo (Jef ‘den acht’ Wouters, Christiaan ‘brandjes blussen’ Torfs, Freddy ‘juuuu!’ Byl, Maarten ‘afterburner’ Pauwels, Bob ‘passela’ Hebb, Johannes ‘crucifix’ Pyck, . . .), die ik vaak te lang niet zie, maar met wie de feestjes altijd verlopen met een enthousiasme alsof we elkaar jaren niet hebben gezien en met een band alsof we elkaar de dag ervoor nog zijn tegengekomen. Ik vergeet ook niet de rest van de bende uit Leuven (Freddy J, Wim A, Pieter, Lore, Raf, Nathalie V, Wim & Carolien, Bart & Sofie, Filip & Marleen, . . .) omwille van de vele escapades die we samen meemaakten. Ik vergeet hier ongetwijfeld vele namen, waarvoor excuses, maar ik wil toch iedereen bedanken voor de vele ontspannende momenten. Zonder jullie had ik mijn doctoraat een half jaar vroeger af kunnen hebben, maar had ik het waarschijnlijk ook niet volgehouden.

Aan Boudewijn, onze ex-pat, zou ik nog een apart dankwoordje willen richten. Bowie, jongen, bedankt om pas in juni naar België te komen. Mei zou een probleem geweest zijn . . .

Tenslotte wil ik ook mijn collega’s doctoraatsstudenten bedanken. Binnen de onderzoeksgroep hangt er een hechte sfeer, die het aangenaam toeven maakt en die ervoor heeft gezorgd dat heel wat van hen ondertussen echte vrienden zijn geworden. Raf, Tom, Nathalie, Ruth, Pieter, Frizo, Thomas, Karen, Steven, Niels, Marcelo, Carlos, Toni, Mauricio, Fabian en alle anderen, ik hoop dat ik jullie nog ga blijven zien eens mijn en jullie doctoraat definitief achter de rug is!

Ik zou Tom ook nog apart willen bedanken voor onze *intensieve* samenwerking van de laatste drie jaar. Tom, ik ken weinig mensen met jouw gemotiveerdheid en ik bewonder jouw drang om steeds nieuwe dingen te willen bijleren. *“Go for it !!!”*

Special thanks also go out to Nathalie and Marcelo for agreeing to make parallel plans for the final stages of our Ph.D.’s. It’s been very helpful to have a detailed time schedule so we could stick to it step by step without having to worry about anything.

Ongetwijfeld ben ik, mezelf kennende, ook heel wat mensen vergeten te vermelden, waarvoor alvast mijn excuses. Ik zou immers iedereen die op eender welke manier geholpen heeft bij het tot stand komen van dit doctoraat, mij tijdens de voorbije vier jaar op eender welke manier heeft bijgestaan of mij het leven aangenamer heeft gemaakt, oprecht willen bedanken.

Tot slot zou ik mijn collega’s die nog volop aan het werken zijn aan hun doctoraat een hart onder de riem willen steken . . .

“Het beste moet áltijd nog komen!”

Bert

Leuven
6 mei 2006

Abstract

Model based Predictive Control (MPC) is an automatic control paradigm that has gained widespread acceptance in industry due to its unique advantages compared to classic control methods. The main distinguishing features are the ability to efficiently control large scale interconnected systems and the inherent ability to cope with physical and other constraints of the controlled system.

MPC controllers are designed on the basis of a dynamical model of the system that has to be controlled (i.e., the plant) and apply mathematical optimization techniques in order to obtain the optimal inputs to be applied to the plant. Crucial aspects are hence the accuracy of the dynamical model and the computational burden of the optimization that has to be performed. In this thesis the focus is on robust MPC algorithms, i.e., MPC algorithms that can take model uncertainty into account and can guarantee stable behavior and acceptable performance despite mismatches between the real plant behavior and the dynamical model that is used. More specifically, the main aim of this thesis is the development of MPC algorithms with improved computational efficiency and improved scaling behavior compared to existing algorithms, while enabling less conservative constraint handling. These aims are achieved in two ways: by making mathematical and conceptual contributions to existing MPC algorithms and by formulating improved algorithms for the construction of invariant sets, a mathematical concept used in the design phase of MPC controllers.

On the MPC-algorithmic level, contributions are made with respect to the type of predictions that er made by the controller. Closed-loop predictions are shown to be preferrable above open-loop predictions. Existing algorithms are converted from the latter to the former type of predictions in order to guarantee recursive feasibility of the optimization problems and new algorithms with closed-loop predictions are presented, resulting in significantly improved constraint handling.

On the level of invariant set synthesis a new class of polyhedral invariant sets is introduced, that makes a trade-off between maximal volume and minimal complexity. In this way the computational complexity of the construction of the invariant sets and their application in MPC algorithms is reduced significantly. It is shown that under certain conditions the exponential scaling behavior of existing algorithms is reduced to linear scaling behavior, enabling their application to larger scale systems and enabling the use of larger prediction horizons.

Several numerical examples and simulation on models of two industrial processes show the improved properties of the obtained algorithms.

Korte Inhoud

Modelgebaseerde Predictieve Controle (MPC) is een regeltechnische methode die in een breed spectrum aan industriële toepassingen gebruikt wordt omwille van de specifieke voordelen ten opzichte van klassieke technieken. De belangrijkste voordelen zijn de mogelijkheid om grootschalige systemen te regelen en om fysische beperkingen expliciet in rekening te brengen.

MPC regelaars worden ontworpen op basis van een dynamisch model van het te regelen systeem en gebruiken wiskundige optimalisatie om optimale ingangen te bepalen die aan het systeem aangelegd worden. Cruciale aspecten hierbij zijn de nauwkeurigheid van het model en de rekenkundige complexiteit van de op te lossen optimalisatieproblemen. In deze thesis zal de nadruk liggen op robuuste MPC algoritmes, dewelke modelonzekerheid in rekening kunnen brengen en zo stabiliteit en aanvaardbaar regelgedrag kunnen garanderen ondanks verschillen tussen het dynamisch gedrag van het werkelijke systeem en het gebruikte model. Meer specifiek zullen nieuwe MPC algoritmes ontwikkeld worden met verbeterde rekenkundige efficiëntie ten opzichte van bestaande algoritmes, terwijl opgelegde beperkingen minder conservatief afgehandeld zullen kunnen worden. Deze doelen worden bereikt door wiskundige en conceptuele bijdragen te leveren tot bestaande MPC algoritmes enerzijds en tot algoritmes voor het opstellen van invariante verzamelingen, een wiskundig concept dat gebruikt wordt tijdens het ontwerp van MPC regelaars, anderzijds.

Op MPC-algorithmisch gebied zijn bijdragen geleverd met betrekking tot het type predicties dat gemaakt wordt door de regelaar. Er wordt aangetoond dat gesloten-lus predicties de voorkeur genieten boven open-lus predicties. Bestaande algoritmes worden aangepast naar gesloten-lus predicties om recursieve oplosbaarheid te kunnen garanderen en bovendien worden nieuwe algoritmes geformuleerd die significant beter kunnen omgaan met opgelegde beperkingen.

Anderzijds wordt een nieuwe klasse van polyhedrale invariante verzamelingen geïntroduceerd, die een afweging maakt tussen een maximaal volume en een minimale complexiteit. Op deze manier wordt de rekencomplexiteit van zowel de constructie van dergelijke verzamelingen als hun toepassing in MPC algoritmes gereduceerd. Er wordt aangetoond dat onder bepaalde voorwaarden exponentieel schalingsgedrag gereduceerd kan worden tot linear schalingsgedrag, wat het mogelijk maakt om de robuuste MPC algoritmes op meer grootschalige systemen toe te passen en om een langere predictiehorizon te gebruiken.

Verscheidene numerieke voorbeelden en simulaties op modellen van twee industriële processen tonen de verbeterde eigenschappen van de bekomen algoritmes.

Notation

Variables

$\alpha, \beta, \gamma \in \mathbb{R}$	Greek symbols denote scalar variables
$a, b, c \in \mathbb{R}^n$	Lower case roman symbols denote scalar or vector variables
$A, B, C \in \mathbb{R}^{m \times n}$	Upper case roman symbols denote matrix variables
$A(i,j), A \in \mathbb{R}^{m \times n}$	Element at the i^{th} row and j^{th} column of A
$A(i,:), A \in \mathbb{R}^{m \times n}$	i^{th} row of a matrix A
$A(:,j), A \in \mathbb{R}^{m \times n}$	j^{th} column of a matrix A
$A(i:j, k:l), A \in \mathbb{R}^{m \times n}$	Submatrix spanning rows i through j and columns k through l of matrix A
$[a; b; c]$	Stacked vectors: $[a^{\mathrm{T}}\ b^{\mathrm{T}}\ c^{\mathrm{T}}]^{\mathrm{T}}$
$x_{1\ldots n}$	Enumeration: $x_{1\ldots n} \geq 0 \quad \Leftrightarrow \quad x_i \geq 0,\ i = 1, \ldots, n$

Scalar sets

$\mathbb{R}$, $\mathbb{R}^+$	Set of real numbers and positive real numbers
$\mathbb{Z}$, $\mathbb{Z}^+$, $\mathbb{Z}_0^+$	Set of integers, positive integers and strictly positive integers respectively
$\mathbb{N}$	Set of positive integers

Vector sets

$\mathcal{A}, \mathcal{B}, \mathcal{C} \subseteq \mathbb{R}^n$	Sets in n-dimensional space
$\mathcal{P} \subseteq \mathbb{R}^n$	Polyhedral set in n-dimensional space $\{x \in \mathbb{R}^n \vert A_{\mathcal{P}} x \leq b_{\mathcal{P}}\}$
$\mathcal{E} \subseteq \mathbb{R}^n$	Ellipsoidal set in n-dimensional space $\{x \in \mathbb{R}^n \vert x^{\mathrm{T}} Z^{-1} x \leq 1\}$ with $Z = Z^{\mathrm{T}} \succ 0$
$\mathbf{1}$	Vector of appropriate dimensions containing only ones: $[1; \ldots; 1]$

Matrix sets

$\mathbb{S}^n$	Set of symmetric $n \times n$ 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

Operators

$\triangleq$	Definition
$\equiv$	Equivalence
$:=$	Assignment $a := b \Leftrightarrow$ the value of b is assigned to variable a
$>, \geq$	(Strict) scalar inequality $A > B \Leftrightarrow A - B$ has strictly positive elements
$\succ, \succeq$	(Strict) matrix inequality $A \succ B \Leftrightarrow A - B$ is strictly positive definite

Inputs, states, outputs

$u(k) \in \mathbb{R}^{n_u}$	n_u-dimensional input vector at discrete time k
$x(k) \in \mathbb{R}^{n_x}$	n_x-dimensional state vector at discrete time k
$y(k) \in \mathbb{R}^{n_y}$	n_y-dimensional output vector at discrete time k
$w(k) \in \mathbb{R}^{n_x}$	Additive state disturbance vector at discrete time k
$u_{\mathrm{ref}}(k), x_{\mathrm{ref}}(k), y_{\mathrm{ref}}(k)$	Reference values for inputs, states and outputs
$u(k+i\|k), x(k+i\|k)$	Input and state vectors at discrete time $k+i$ as predicted / calculated at time k
$\mathcal{U} \subseteq \mathbb{R}^{n_u}$	Constraint set to be imposed on the input vectors
$\mathcal{X} \subseteq \mathbb{R}^{n_x}$	Constraint set to be imposed on the state vectors
$\mathcal{Y} \subseteq \mathbb{R}^{n_y}$	Constraint set to be imposed on the output vectors
m_x, m_u	Number of constraints defining $\mathcal{X}, \mathcal{U}$ respectively
$\mathcal{W} \subseteq \mathbb{R}^{n_x}$	Constraint set bounding the disturbance vectors

Model based predictive control

$N \in \mathbb{Z}_0^+$	Prediction horizon length
$\kappa_N(\cdot) : \mathbb{R}^{n_x} \to \mathbb{R}^{n_u}$	Terminal control law
$\mathcal{X}_N \subseteq \mathbb{R}^{n_x}$	Terminal constraint set
$Q_N \in \mathbb{R}^{n_x \times n_x}$	Terminal cost matrix
$\mathbf{u}_N(k)$	Open-loop input sequence $[u(k\|k); \ldots; u(k+N-1\|k)]$
$\mathbf{u}_{\mathrm{cl},N}$	Closed-loop input sequence
$\mathbf{x}_N(k)$	Nominal state prediction sequence $[x(k+1\|k); \ldots; x(k+N\|k)]$
$\mathcal{X}_{\mathbf{u}_N}(k+i\|k)$	Open-loop state prediction set
$\mathcal{X}_{\mathbf{u}_{\mathrm{cl},N}}(k+i\|k)$	Closed-loop state prediction set

Matrix operations

A^{T}	Transpose of matrix A
$\mathrm{Tr}(A)$	Trace of a matrix i.e. sum of its diagonal elements
$\mathrm{rows}(A)$	number of rows in matrix A
$\mathrm{cols}(A)$	number of columns in matrix A

Norms

$\|x\|_2, x \in \mathbb{R}^n$	2-norm of a vector: $\sqrt{x^{\mathrm{T}}x}$
$\|x\|_p, x \in \mathbb{R}^n$	p-norm of a vector: $(\sum_{i=1}^{n} \|x_i\|^p)^{1/p}$
$\|x\|_Q, x \in \mathbb{R}^n$	Weighted 2-norm of a vector: $\sqrt{x^{\mathrm{T}}Qx}$ with $Q \in \mathbb{S}^n_{++}$
$\hat{\rho}(\cdot)$	Joint Spectral Radius (see Appendix C)

Optimization

$\min_x$	Function minimization over x, optimal function value is returned
$\arg\min_x$	Function minimization over x, optimal value of x is returned
s.t.	Subject to constraints

Convex functions, convex sets

$\mathrm{Co}\{\cdot\} \subseteq \mathbb{R}^n$	Convex hull of a set of points or sets in $\mathbb{R}^n$
$\mathrm{epi}(f) \subseteq \mathbb{R}^{n+1}$	Epigraph of a function $f : \mathbb{R}^n \to \mathbb{R}$
$\mathrm{dom}(f) \subseteq \mathbb{R}^n$	Domain of a function $f : \mathbb{R}^n \to \mathbb{R}^m$

Geometric operations

$\cup$	Union of sets
$\cap$	Intersection of sets
$\oplus$	Minkowski sum of sets
$\ominus$	Minkowski (or Pontryagin) difference of two sets
$\mathrm{proj}(\mathcal{P}), \mathcal{P} \subset \mathbb{R}^n$	Projection of a polytope along the n-th dimension
$\mathrm{elim}(\mathcal{P}), \mathcal{P} \subset \mathbb{R}^n$	Elimination of a polytope along the n-th dimension
$e_k \in \mathbb{R}^n, k \in \{1, \ldots, n\}$	k-th unit vector in $\mathbb{R}^n$: $[0; \ldots; 1; \ldots; 0]$ (k-th component)

Acronyms

APC	Advanced Process Control
FIR	Finite Impulse Response
IIR	Infinite Impulse Response
JSR	Joint Spectral Radius
LP	Linear Program(ming)
LPV	Linear Parameter-Varying
LMI	Linear Matrix Inequality

Acronyms (continued)

LTI	Linear Time-Invariant
LTV	Linear Time-Varying
LQR	Linear Quadratic Regulator
MAS	Maximal Admissible Set
MIMO	Multiple-Input / Multiple-Output
MPC	Model based Predictive Control
NLP	Non-Linear Program(ming)
PID	Proportional / Integral / Differential
QP	Quadratic Program(ming)
RHC	Receding Horizon Control
SDP	Semi Definite Program(ming)
SQP	Sequential Quadratic Programming
SISO	Single-Input / Single-Output
SOCP	Second-Order Cone Program(ming)

Contents

Robuuste modelgebaseerde predictieve controle: ontwerp via invariante verzamelingen

Hoofdstuk 1: Inleiding

Dit hoofdstuk geeft een algemene inleiding tot deze thesis en introduceert concepten zoals procescontrole, modelgebaseerde predictieve controle, convexe optimalisatie, computationele geometrie, ... Deze samenvatting beperkt zich tot het geven van een algemene inleiding tot procescontrole waarna de basisconcepten van Modelgebaseerde Predictieve Controle (MPC) zullen toegelicht worden. Voor meer details hieromtrent en voor meer informatie wat betreft de specifieke wiskundige technieken die gebruikt zullen worden in latere hoofdstukken, wordt verwezen naar de Engelstalige versie van dit hoofdstuk.

Procescontrole

In 1788 introduceerde James Watt een *centrifugale regelaar* ("centrifugal governor", zie Figuur 1.1) voor het regelen van de rotatiesnelheid van zijn stoommachine. Dit leidde tot een grotere betrouwbaarheid van zijn machine, waardoor deze algemene ingang kon vinden in de industrie. Deze centrifugale regelaar kan dus gezien worden als een belangrijke factor die de industriële revolutie aan het einde van de 18e eeuw mogelijk maakte.

Tegelijkertijd kan de centrifugale regelaar gezien worden als het eerste industrieel toegepaste voorbeeld van stabilisatie door middel van expliciete terugkoppeling. De stoomtoevoer naar de machine (ingang) wordt immers geregeld op basis van het toerental (uitgang) van de machine. De uitgang van het systeem wordt dus teruggekoppeld naar de ingang teneinde een stabieler systeem te bekomen. Dergelijke technieken vallen onder de bredere noemer *procesregeling* of *procescontrole*.

In de moderne procesindustrie wordt de techniek van terugkoppeling veelvuldig toegepast in de vorm van PID regelaars voor het aansturen van allerhande processen. In de huidige praktijk dienen er echter veel grotere systemen geregeld te worden,

waarbij meerdere ingangen moeten aangestuurd worden om meerdere uitgangen een bepaalde gewenste waarde te laten aannemen of een bepaald traject te laten volgen. Hierbij treden er vaak interacties op tussen de verschillende ingangen en uitgangen, waardoor meer geavanceerde technieken, die deze interacties in rekening kunnen brengen, gebruikt moeten worden.

Bovendien zijn er vaak fysische beperkingen aanwezig in het systeem (een klep kan bv. niet verder dan 100% open staan), dienen er vaak veiligheidsvoorschriften in acht genomen te worden (bv. maximale druk binnen in een reactor) en gelden er economische en logistieke beperkingen (maximale toevoersnelheid van bepaalde grondstoffen). Typisch opereren productieprocessen dicht tegen deze beperkingen met het oog op het maximaliseren van het rendement, het reduceren van de kost, ...Deze beperkingen moeten dus ook in rekening gebracht kunnen worden bij het regelaarontwerp.

De enige techniek die aan deze voorwaarden voldoet, is Modelgebaseerde Predictieve Controle (MPC), dewelke dan ook meer en meer gebruikt wordt in de industrie.

Modelgebaseerde predictieve controle

Modelgebaseerde Predictieve Controle (MPC) is een op optimalisatie gebaseerde techniek, die op elke discrete tijdstap (bv. elke minuut) een optimaal ingangstraject bepaalt binnen een eindig toekomstig tijdsvenster, waarna van dit traject enkel de eerste waarden effectief worden aangelegd aan het systeem. Op de volgende discrete tijdstap wordt de procedure herhaald, gebaseerd op nieuwe metingen van de toestand van het systeem. Deze werkwijze is weergegeven in Figuur 1.4.

Teneinde te kunnen voorspellen wat het effect zal zijn van het aanleggen van bepaalde ingangen aan het te regelen systeem, wordt typisch een zogenaamd toestandsruimte-model gebruikt om dit dynamisch systeem (Figuur 1.3) te beschrijven:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k), \\ y(k) &= Cx(k), \end{aligned} \qquad k \in \mathbb{N},$$

waarbij $u(k) \in \mathbb{R}^{n_u}, x(k) \in \mathbb{R}^{n_x}$ en $y(k) \in \mathbb{R}^{n_y}$ respectievelijk de ingangen, toestanden en uitgangen van het systeem voorstellen op discrete tijdstap k. In volgende hoofdstukken zal naar modellen van dit type verwezen worden als lineaire, tijdsinvariante (LTI) modellen. De beperkingen die beschreven zijn in de vorige sectie, worden wiskundig uitgedrukt als

$$u(k) \in \mathcal{U}, \qquad x(k) \in \mathcal{X}, \qquad y(k) \in \mathcal{Y}, \qquad k \in \mathbb{N},$$

waarbij $\mathcal{U} \subseteq \mathbb{R}^{n_u}, \mathcal{X} \subseteq \mathbb{R}^{n_x}, \mathcal{Y} \subseteq \mathbb{R}^{n_y}$ respectievelijk de ingangs-, toestands- en uitgangsbeperkingen voorstellen.

Gegeven dit model en deze beperkingen kan nu het MPC algoritme dat geïntroduceerd werd in [135], maar beter bekend staat onder de vorm beschreven in [131], geformuleerd worden. Dit algoritme, waarnaar vaak verwezen wordt als *MPC met quasi-oneindige horizon*, lost op elk tijdstip k, gegeven de huidige toestand $x(k) \equiv$

$x(k|k)$, het volgende optimalisatieprobleem op:

$$\min_{\mathbf{x}(k),\mathbf{u}(k)} \quad \sum_{i=0}^{N-1} \|x(k+i|k)\|_Q^2 + \sum_{i=0}^{N-1} \|u(k+i|k)\|_R^2 + \|x(k+N|k)\|_{Q_N}^2,$$

$$\begin{aligned} \text{s.t.} \quad & x(k+i|k) \in \mathcal{X}, && i = 1, \ldots, N-1, \\ & x(k+N|k) \in \mathcal{X}_N, \\ & u(k+i|k) \in \mathcal{U}, && i = 0, \ldots, N-1, \\ & x(k+i+1|k) = Ax(k+i|k) + Bu(k+i|k), && i = 0, \ldots, N-1, \end{aligned}$$

waarna $u^{\text{o}}(k|k)$ aangelegd wordt aan het systeem. Hierin stellen $Q \in \mathbb{S}_{++}^{n_x}$, $R \in \mathbb{S}_{++}^{n_u}$ gewichtsmatrices voor die het relatieve belang van de toestanden en de ingangen voorstellen in de te minimaliseren regelkost. $\mathcal{X}_N$ en Q_N stellen respectievelijk de zogenaamde *eindbeperking* en *eindkost* voor. $x(k+i|k)$ stelt de toestand voor op tijdstip $k+i$ zoals voorspeld op tijdstip k. Een gelijkaardige definitie geldt voor $u(k+i|k)$.

Eenvoudig gezegd zorgt de eindbeperking ervoor dat de opgelegde beperkingen gerespecteerd worden op tijdstippen 'voorbij' de horizon, terwijl de eindkost de regelkost 'voorbij' de horizon in rekening brengt. Op deze manier wordt bij benadering een optimaal regelprobleem met oneindige horizon opgelost, waaraan het algoritme dan ook zijn naam te danken heeft.

Het resulterende algoritme leidt gegarandeerd tot een stabiel gesloten lus systeem indien aan voorwaarden (1.12) is voldaan. Voorwaarde (1.12c) eist dat $\mathcal{X}_N$ een *invariante verzameling* is ten opzichte van het gesloten lus systeem gevormd door het te regelen dynamisch systeem en een lokaal stabiliserende *eindregelaar*. Het bepalen van een eindbeperking $\mathcal{X}_N$ is een cruciale factor in de uiteindelijke eigenschappen van de resulterende MPC regelaar. De grootte van deze verzameling bepaalt de grootte van het werkingsgebied waarbinnen de MPC regelaar geldig is. De manier waarop deze invariante verzameling beschreven wordt, is bepalend voor de computationele vereisten van de regelaar. Dit is meteen de belangrijkste reden waarom onderzoek naar algoritmes voor het opstellen van dergelijke verzamelingen voor verschillende klassen van dynamische systemen een groeiend onderzoeksveld is gebleken in het afgelopen decennium.

Hoofdstuk 2: Polyhedrale Invariante Verzamelingen

In dit hoofdstuk wordt de stand van zaken wat betreft het opstellen van invariante verzamelingen voor lineaire, onzekere systemen toegelicht. Zoals uitgelegd in Hoofdstuk 1, vormen invariante verzamelingen een essentieel element in het ontwerp van een MPC regelaar. De voornaamste algoritmes voor het construeren van invariante verzamelingen worden toegelicht alsook hun voornaamste eigenschappen, met het oog op het verfijnen van deze algoritmes in Hoofdstuk 5. Deze samenvatting beperkt zich tot het beschrijven van het belangrijkste algoritme en schetst kort de belangrijkste eigenschappen.

Model en definities

Zoals uitgelegd in Hoofdstuk 1, is het bepalen van invariante verzamelingen van cruciaal belang bij het ontwerp van een MPC regelaar. In dit hoofdstuk wordt aangetoond hoe polyhedrale invariante verzamelingen kunnen opgesteld worden voor autonome lineaire, onzekere systemen, dewelke een belangrijk hulpmiddel zijn voor het bekomen van verbeterde robuuste MPC algoritmes, waarover de volgende 2 hoofdstukken zullen handelen.

In dit hoofdstuk beschouwen we autonome, lineaire, onzekere systemen van de volgende vorm:

$$x(k+1) = \Phi(k)x(k), \qquad k \in \mathbb{N},$$

waarbij de matrices $\Phi(k) \in \mathbb{R}^{n_x \times n_x}$ tot een onzekerheidspolytoop $\Omega' \subset \mathbb{R}^{n_x \times n_x}$ behoren, die gedefinieerd is als

$$\Phi(k) \in \Omega' \equiv \mathrm{Co}\{\Phi_1, \ldots, \Phi_r\}, \qquad k \in \mathbb{N},$$

of equivalent

$$\Phi(k) \in \left\{ \sum_{j=1}^{r} \lambda_j(k)\Phi_j \;\middle|\; \lambda_{1\ldots r}(k) \geq 0, \sum_{j=1}^{r} \lambda_j(k) = 1 \right\}, \qquad k \in \mathbb{N}.$$

Op elk tijdstip k ligt de systeemmatrix $\Phi(k)$ dus binnen een onzekerheidspolytoop Ω'. De exacte waarden van deze systeemmatrices liggen niet à priori vast, kunnen op elke tijdstip verschillend zijn en zijn ook niet gekend door de regelaar. In wat volgt zal verwezen worden naar deze systemen als autonome LPV systemen. Een verzameling $\mathcal{S}$ wordt invariant genoemd met betrekking tot bovenvermeld autonoom systeem als en enkel als

$$\Phi x \in \mathcal{S}, \qquad \forall x \in \mathcal{S}, \quad \forall \Phi \in \Omega'.$$

Bovendien voldoet een verzameling $\mathcal{S}$ aan de opgelegde beperkingen als en enkel als

$$\mathcal{S} \subseteq \mathcal{X}.$$

Er kunnen twee belangrijke klassen van invariante verzamelingen onderscheiden worden, namelijk *ellipsoïdale* en *polyhedrale* invariante verzamelingen, respectievelijk genoteerd als $\mathcal{E}$ en $\mathcal{P}$ en gedefinieerd als:

$$\mathcal{E} = \{x | x^{\mathrm{T}} Z^{-1} x \leq 1\},$$
$$\mathcal{P} = \{x | A_{\mathcal{P}} x \leq \mathbf{1}\},$$

met $Z \in \mathbb{S}^{n_x}_{++}$ en $A_{\mathcal{P}} \in \mathbb{R}^{m \times n_x}$. Ellipsoïdale invariante verzamelingen kunnen eenvoudig opgesteld worden door het oplossen van een convex optimalisatieprobleem (zie [23]). Deze hebben echter als nadeel dat ze niet in staat zijn om te gaan met asymmetrische beperkingen, dat ze typisch een kleiner volume hebben dan polyhedrale verzamelingen en dat ze leiden tot MPC regelaars met een hogere rekencomplexiteit. Om deze redenen zijn polyhedrale invariante verzamelingen verkiesbaar boven ellipsoïdale. Polyhedrale verzamelingen kunnen echter niet opgesteld worden door het oplossen van een convex optimalisatieprobleem, maar moeten iteratief geconstrueerd worden. Voor het geval $r = 1$ zijn de technieken uit [52] algemeen gekend, maar voor het meer algemene geval $r > 1$ zijn meer geavanceerde technieken vereist.

Polyhedrale invariante verzamelingen voor LPV systemen

In deze sectie wordt de methode die beschreven wordt in [13,15,63,98] kort behandeld, met nadruk op de structuur die vervat ligt in de resulterende verzameling.

Van het volgende algoritme kan aangetoond worden dat het de grootst mogelijke invariante verzameling bepaalt die voldoet aan de opgelegde beperkingen $\mathcal{X} \triangleq \{x | A_x x \leq \mathbf{1}\}$:

1. Initialiseer $A_{\mathcal{S}} := A_x,\ i := 1$.
2. Voer de volgende stappen iteratief uit totdat $i >$ rijen$(A_{\mathcal{S}})$:
   (a) Stel $a := A_{\mathcal{S}}(i, :)$.
   (b) Controleer de redundantie van de beperkingen $a\Phi_i x \leq 1,\ i = 1, \ldots, r$ met betrekking tot $\mathcal{S} \triangleq \{x | A_{\mathcal{S}} x \leq \mathbf{1}\}$. Voor alle $i = 1, \ldots, r$, voeg, indien $\mathrm{sig}_{\mathcal{S}}(a\Phi_i) > 1$, de beperking $a\Phi_i x \leq 1$ toe aan $A_{\mathcal{S}}$ door $A_{\mathcal{S}} := [A_{\mathcal{S}}; a\Phi_i]$ te stellen.
   (c) Voer indien nodig[1] *vuilopruiming* uit, m.a.w. controleer voor elke rij van $A_{\mathcal{S}}$ of de overeenkomstige beperking redundant is ten opzichte van de andere rijen van $A_{\mathcal{S}}$ en zo ja, verwijder die rij uit $A_{\mathcal{S}}$.
   (d) Stel $i := i + 1$.
3. Geef de resulterende verzameling $\mathcal{S} \triangleq \{x | A_{\mathcal{S}} x \leq \mathbf{1}\}$ terug als resultaat.

Doordat in elke iteratie enkel die beperkingen toegevoegd worden die significant zijn, wordt een significante tijdswinst bekomen voor het bepalen van de invariante

[1] Als vuistregel kan vuilopruiming toegepast worden telkens rijen$(A_{\mathcal{S}})$ met 50% is toegenomen.

verzameling. Dit werd reeds aangekaart in [15]. Echter, voor grootschalige systemen kan empirisch worden vastgesteld dat het aantal redundante beperkingen afneemt, waardoor bovenstaand algoritme vooral zijn nut heeft voor lager dimensionale systemen. Hoofdstuk 5 beschrijft varianten van dit algoritme, die beter geschikt zijn voor het bepalen van invariante verzamelingen voor grootschalige systemen.

De belangrijkste bijdrage van dit hoofdstuk, naast het beschrijven van de basisprincipes wat betreft invariante verzamelingen. is tweeledig. Enerzijds wordt de boomstructuur geanalyseerd die aanwezig is in invariante verzamelingen. Anderzijds wordt convergentie van bovenstaand algoritme gekoppeld aan de *gemeenschappelijke spectrale radius* (Joint Spectral Radius, JSR).

Sectie 2.3.2 beschrijft de boomstructuur waarin men de lineaire ongelijkheidsbeperkingen van de polyhedrale invariante verzameling kan onderbrengen. Dit wordt geïllustreerd aan de hand van Figuren 2.2 en 2.4. De beperkingen die aan A_S toegevoegd worden, worden toegekend aan een bepaalde laag van de boom afhankelijk van de iteratie waarin ze werden toegevoegd. De takken van de boom duiden aan welke beperkingen van welke beperkingen werden afgeleid. Een belangrijk kenmerk is dat indien een beperking redundant is, automatisch ook alle kinderen van de beperking redundant zullen zijn. Het aantal kinderen dat een beperking kan hebben, wordt bepaald door het aantal hoekpunten r in de onzekerheidspolytoop. Hoofdstuk 5 zal technieken beschrijven die enerzijds het aantal kinderen van elke beperking rachten te beperken en die anderzijds tot doel hebben de diepte van de boom te verminderen.

Sectie 2.3.1 legt een verband tussen de convergentie van bovenstaand algoritme en de JSR van de gesloten lus matrices $\Phi_1, \ldots, \Phi_r$, die genoteerd wordt als $\hat{\rho}(\Omega')$. Voor meer uitleg wat betreft de JSR, verwijzen we naar Appendix C. Theorema 2.2 toont aan dat indien

$$\hat{\rho}(\Omega') < 1,$$

het bovenstaande algoritme convergeert in een eindig aantal iteraties. Aangezien $\hat{\rho}(\Omega') < 1$ een nodige en voldoende voorwaarde is voor asymptotische stabiliteit van het autonome, lineaire, onzekere systeem dat hier beschouwd wordt, garandeert dit resultaat dat bovenstaand algoritme convergeert voor alle asymptotisch stabiele autonome systemen die binnen deze klasse vallen. Bovendien zal dit theorema toelaten om in Hoofdstuk 5 kwantitatieve resultaten op te stellen wat betreft convergentie van de nieuwe algoritmes die daar beschreven worden.

Hoofdstuk 3: Robuuste Modelgebaseerde Predictieve Controle

In dit hoofdstuk wordt een inleiding gegeven wat betreft robuuste MPC, waarna een overzicht gegeven wordt van verschillende methodes om met opgelegde beperkingen om te gaan. Er wordt aangetoond dat zogenaamde gesloten lus predicties *gebruikt moeten worden indien men recursieve oplosbaarheid van de MPC optimalisatieproblemen wil kunnen bewijzen. Tenslotte wordt aangetoond dat twee algoritmes uit de literatuur zogenaamde* open-lus predicties *gebruiken, ten gevolge waarvan recursieve oplosbaarheid niet kan aangetoond worden. De algoritmes worden gecorrigeerd en er wordt aangetoond dat de nieuwe algoritmes wel deze eigenschap hebben.*

Robuuste MPC

In Hoofdstuk 1 werd een introductie gegeven over MPC gebaseerd op lineaire, tijdsinvariante (LTI) modellen. In de praktijk is het werkelijke model van het te regelen systeem echter nooit perfect gekend. Oorzaken hiervan zijn de eindigheid van de data aan de hand waarvan de modellen opgesteld zijn, meetruis, modelreductietechnieken, enz... Het feit dat er een verschil is tussen het gebruikte predictiemodel en het werkelijke systeem (*mismatch*) kan ertoe leiden dat de performantie van de regelaar degradeert of dat er stabiliteitsproblemen optreden. Het doel van robuuste MPC is het in rekening brengen van deze onzekerheid op het gebruikte model om tot een goede regeling te komen ondanks deze modelfouten. Dergelijke lineaire, onzekere modellen kunnen als volgt beschreven worden:

$$x(k+1) = A(k)x(k) + B(k)u(k), \qquad k \in \mathbb{N},$$

met

$$[A(k)\ B(k)] \in \Omega \equiv \mathrm{Co}\{[A_1\ B_1], \ldots, [A_r\ B_r]\}, \qquad k \in \mathbb{N}.$$

Het specifieke probleem dat robuuste MPC bemoeilijkt is het feit de werkelijke waarden van de matrices $[A(k)\ B(k)]$ ongekend zijn voor de regelaar, dus er moet rekening gehouden worden met alle mogelijke waarden. Dit geeft aanleiding tot een *boom* van toestandsvoorspellingen, zoals weergegeven in Figuren 3.1 en 3.2. Het verschil tussen deze twee figuren is essentieel en wordt verduidelijkt in de volgende sectie.

Open-lus en gesloten-lus predicties

Bij robuuste MPC kan een onderscheid gemaakt worden tussen *open-lus predicties* en *gesloten-lus predicties*, afhankelijk van het soort ingangsequentie waarover geoptimaliseerd wordt.

Figuur 3.1 stelt schematisch voor hoe open-lus predicties geconstrueerd worden. Er wordt nog steeds geoptimaliseerd over een unieke sequentie van ingangen, net zoals bij niet-robuuste MPC. De boom van toestandspredicties wordt opgesteld door gebruik te maken van de r verschillende hoekpunten van de onzekerheidspolytoop.

Figuur 3.2 stelt schematisch voor hoe gesloten-lus predicties geconstrueerd worden. Gesloten-lus predicties houden rekening met het feit dat op tijdstippen $k+i, i \in \{1, \ldots, N-1\}$ men bijkomende kennis zal hebben over de toestand van het systeem op dat moment (doordat er nieuwe metingen binnenkomen) en men dus de aangelegde ingang hier kan aan aanpassen. Dit wordt in rekening gebracht door voor de r^i verschillende toestandspredicties op tijdstip $k+i$ aparte ingangsvariabelen te beschouwen waarover geoptimaliseerd kan worden.

In beide gevallen wordt de maximale waarde (over alle mogelijke predicties) van de regelkost geminimaliseerd met als beperking dat alle mogelijke predicties aan de opgelegde beperkingen moeten voldoen, hetgeen aanleiding geeft tot min-max optimalisatieproblemen.

Gesloten-lus predicties hebben als voordeel dat er meer vrijheidsgraden beschikbaar zijn om over te optimaliseren, waardoor een betere regeling mogelijk wordt. Anderzijds heeft deze methode als nadeel dat het aantal optimalisatievariabelen exponentieel toeneemt in functie van de horizon lengte N, wat al snel voor problemen kan zorgen wat betreft de rekencomplexiteit. Toch is het aangeraden om gesloten-lus predicties te gebruiken om redenen die verduidelijkt worden in de volgende sectie. Bovendien zal in Hoofdstuk 5 blijken dat het computationele nadeel van gesloten-lus predicties ongedaan kan gemaakt worden door gebruik te maken van nieuwe algoritmes voor het opstellen van invariante verzamelingen.

Correcties van bestaande algoritmes

In de laatste twee secties van Hoofdstuk 3 worden twee robuuste MPC algoritmes uit de literatuur behandeld. Enerzijds gaat het om het algoritme dat geïntroduceerd werd in [142]; anderzijds gaat het om het algoritme dat beschreven wordt in [31]. Beide algoritmes maken gebruik van open-lus predicties. Dit zorgt er bij deze twee specifieke algoritmes echter voor dat recursieve oplosbaarheid van de algoritmes niet kan bewezen worden. Recursieve oplosbaarheid garandeert dat als het MPC optimalisatieprobleem oplosbaar is op tijdstip k, het ook oplosbaar zal zijn op tijdstip $k+1$ en is een nodige voorwaarde voor het bekomen van MPC algoritmes met gegarandeerde stabiliteitseigenschappen. In beide artikels bevindt zich een vergissing in het bewijs van recursieve oplosbaarheid, waardoor deze eigenschap foutief wordt geopperd.

Sectie 3.4 beschrijft hoe het MPC algoritme uit [142] gecorrigeerd kan worden door het gebruik van gesloten-lus predicties, terwijl Sectie 3.5 beschrijft hoe het MPC algoritme uit [31] op een gelijkaardige manier gecorrigeerd kan worden. Omwille van plaatsgebrek kunnen hier geen verdere details gegeven worden en wordt de lezer doorverwezen naar de Engelstalige versie van Hoofdstuk 3. Als algemene conclusie kan men zeggen dat het gebruik van gesloten-lus predicties noodzakelijk is om recursieve oplosbaarheid te kunnen garanderen bij de regeling van onzekere systemen met ingangs- en toestandsbeperkingen.

Hoofdstuk 4: Robuuste MPC met Polyhedrale Invariante Verzamelingen

De nadruk van dit hoofdstuk ligt op het uitbreiden van bestaande robuuste MPC algoritmes en aanverwante technieken tot het gebruiken van polyhedrale invariante verzamelingen. Op deze manier wordt minder conservatief omgegaan met de opgelegde beperkingen en kan in een aantal gevallen een significante reductie van de rekencomplexiteit bekomen worden. Er worden drie belangrijke nieuwe technieken behandeld, die hier elk kort samengevat worden.

Synthese van robuuste lineaire terugkoppelwetten gebruik makende van LMIs en polyhedrale invariante verzamelingen

Zoals kort vermeld in Hoofdstuk 1, is het noodzakelijk bij MPC met quasi-oneindige horizon om een stabiliserende *eindregelaar* te ontwerpen, waarna men kan overgaan tot het bepalen van een geldige eindkost en eindbeperking. Deze eindregelaar is typisch van de vorm $u(k) = -Kx(k)$, waardoor het ontwerp ervan neerkomt op het vinden van een geschikte matrix K, waardoor het gegeven LTI systeem gestabiliseerd wordt. Men kan hiervoor eenvoudigweg een LQR regelaar ontwerpen, dewelke meteen garandeert dat de uiteindelijke regelaar lokaal optimaal regelgedrag heeft. In het geval van robuuste MPC moet er echter voor gezorgd worden dat de uiteindelijke regelaar robuust stabiliserend werkt voor het gegeven LPV systeem. Hiervoor wordt typisch de methode van Kothare et al. [68] gebruikt, dewelke gebaseerd is op *Lineaire Matrix Ongelijkheden* (Linear Matrix Inequalities, LMIs). Om te kunnen garanderen dat vanuit een bepaalde initiële toestand $\bar{x}$ de opgelegde beperkingen gerespecteerd worden, maakt deze methode gebruik van ellipsoïdale invariante verzamelingen, hetgeen omwille van redenen die reeds in Hoofdstuk 2 aangehaald werden, suboptimaal is.

Sectie 4.1 breidt de methode uit [68] uit naar meer algemene regelobjectieven en beperkingen, waarbij kruistermen tussen toestanden en ingangen ook toegelaten zijn (Algoritme 4.1). Ten tweede wordt de methode verder uitgebreid naar het gebruik van polyhedrale in plaats van ellipsoïdale invariante verzamelingen. Dit gebeurt op twee manieren:

1. Een eerste nieuw algoritme (Algoritme 4.2) past eerst de oorspronkelijke methode van [68] toe, waarna voor het resulterende gesloten-lus systeem een polyhedrale invariante verzameling wordt bepaald met behulp van Algoritme 2.4. Op deze manier wordt een minder conservatieve karakterisering bekomen van het werkingsgebied van de bekomen regelaar.

2. Een tweede algoritme (Algoritme 4.3) gaat een stap verder en voert Algoritme 4.2 iteratief uit, telkens met herschaalde beperkingen. Deze techniek zorgt ervoor dat de initiële toestand $\bar{x}$, waarvoor de regelaar optimaal moet zijn, exact op de rand van de resulterende polyhedrale invariante verzameling terecht komt. Hierdoor wordt de conservativiteit minimaal. Het nadeel van deze techniek is

de significant verhoogde rekencomplexiteit. Maar, aangezien dit een techniek is die typisch gebruikt wordt in de ontwerpfase van een MPC regelaar, is dit geen significant nadeel.

Figuren 4.1 en 4.2 tonen duidelijk de verbeterde resultaten van de twee nieuwe algoritmes.

Interpolatie-gebaseerde robuuste MPC

Interpolatie-gebaseerde MPC algoritmes bieden een alternatief voor de meer klassieke MPC algoritmes, zoals MPC met quasi-oneindige horizon. Interpolatie-gebaseerde MPC werd initieel geïntroduceerd in [3] en vertrekt van n verschillende lineaire terugkoppelregelaars $K_i, i = 1, \ldots, n$ en n invariante verzamelingen $\mathcal{S}_i, i = 1, \ldots, n$ overeenkomstig de gesloten-lus systemen gevormd door het te regelen systeem en de n verschillende regelaars. In elke tijdstap wordt de huidige toestand $x(k)$ opgesplitst in n deelcomponenten $\hat{x}_i, i = 1, \ldots, n$ als volgt:

$$x(k) = \sum_{i=1}^{n} \hat{x}_i(k), \quad \text{waarbij} \quad \begin{cases} \sum_{i=1}^{n} \lambda_i(k) = 1,\ \lambda_i \geq 0, \\ \hat{x}_i(k) \in \lambda_i(k)\mathcal{S}_i. \end{cases}$$

Op basis van deze opsplitsing in deelcomponenten wordt dan een ingang berekend, die aangelegd wordt aan het systeem:

$$u(k) = -\sum_{i=1}^{n} K_i \hat{x}_i(k).$$

De manier waarop de opsplitsing in componenten gebeurt, geeft vrijheidsgraden waarover in elke tijdstap kan geoptimaliseerd worden, teneinde een vooropgestelde kwadratische kostfunctie te minimaliseren.

Deze methode wordt op twee manieren verbeterd:

1. Enerzijds wordt de methode uitgebreid naar polyhedrale invariante verzamelingen. In het robuuste geval [3] werd er in de literatuur tot op heden enkel ellipsoïdale invariante verzamelingen beschouwd. Het gebruik van polyhedrale verzamelingen zorgt voor een reductie van de rekenkost, zorgt voor minder conservatieve behandeling van beperkingen en laat ook toe om efficiënt om te gaan met asymmetrische beperkingen. Deze werkwijze wordt beschreven in Algoritme 4.5. De voordelen worden geïllustreerd in Figuren 4.5 en 4.6.

2. Anderzijds wordt de methode verder verbeterd door ook de interacties tussen de verschillende regelaars in rekening te brengen. Hierdoor wordt het werkingsgebied van de resulterende regelaar verder vergroot. De opsplitsing in deelcomponenten, zoals beschreven hierboven, maakt in dit geval geen gebruik meer van de individuele invariante verzamelingen $\mathcal{S}_i$, maar maakt gebruik van één enkele invariante verzameling voor het volgende *uitgebreide systeem* (augmented system):

$$x_{\text{aug}}(i+1) = \Psi_{\text{aug}}(i) x_{\text{aug}}(i), \qquad i \in \mathbb{N},$$

waarbij $\Psi_{\text{aug}}(i) \in \Omega_{\text{aug}}, \forall i \in \mathbb{N}$, met Ω_{aug} gedefinieerd als

$$\Omega_{\text{aug}} \triangleq \text{Co}\{\Psi_{\text{aug},1}, \ldots, \Psi_{\text{aug},r}\},$$

met

$$\Psi_{\text{aug},j} \triangleq \begin{bmatrix} A_j - B_j K_n & B_j(K_n - K_1) & \cdots & B_j(K_n - K_{n-1}) \\ 0 & A_j - B_j K_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_j - B_j K_{n-1} \end{bmatrix}, \quad j = 1, \ldots, r,$$

en hetwelke onderhevig is aan volgende beperkingen

$$A_x \Gamma_x x_{\text{aug}}(i) \leq \mathbf{1}, \qquad A_u \Gamma_u x_{\text{aug}}(i) \leq \mathbf{1}, \qquad \forall i \in \mathbb{N},$$

met $\Gamma_x = [I\ 0\ \ldots\ 0], \Gamma_u = [-K_n \quad (K_n - K_1)\ \ldots\ (K_n - K_{n-1})]$. Lemma 4.4 toont aan dat als de uitgebreide toestandsvector $x_{\text{aug}} \triangleq [x(k); \hat{x}_1; \ldots; \hat{x}_{n-1}]$ binnen de resulterende invariante verzameling ligt, de overeenkomstige trajecten aan de opgelegde beperkingen voldoen, hetgeen deze werkwijze (Algoritme 4.6) rechtvaardigt. Figuren 4.8-4.11 illustreren de verbeterde resultaten bekomen met dit algoritme.

Het is belangrijk op te merken dat Algoritme 4.6 de opgelegde beperkingen respecteert door gebruik te maken van één enkele invariante verzameling. Dit toont het belang aan van performante algoritmes voor het opstellen van dergelijke verzamelingen, alsook het belang van efficiënte representaties voor zulke verzamelingen, aangezien deze in grote mate de computationele efficiëntie bepalen van het resulterende algoritme.

Robuuste MPC met quasi-oneindige horizon

In deze sectie wordt de robuuste MPC methode uit [70] uitgebreid naar het gebruik van polyhedrale invariante verzamelingen. Deze methode is een robuuste versie van het MPC algoritme met quasi-oneindige horizon beschreven in Hoofdstuk 1 en maakt gebruik van gesloten-lus voorspellingen. Om het exponentieel toenemend aantal variabelen te vermijden, wordt echter een herparametrisatie van de ingangsequentie gebruikt [123]. Deze is schematisch weergegeven in Figuur 4.12. Hierdoor wordt terug een lineair opschalend aantal optimalisatievariabelen bekomen. Bovendien wordt het hierdoor ook mogelijk, net zoals bij Algoritme 4.6, om de beperkingen van het MPC optimalisatieprobleem te bepalen als een invariante verzameling van een uitgebreid systeem, namelijk:

$$x(i+1)_{\text{aug}} = \Psi_{\text{aug}}(k+i) x(i)_{\text{aug}}, \qquad i \in \mathbb{N},$$

met $\Psi_{\text{aug}}(k+i) \in \Omega_{\text{aug}} \triangleq \text{Co}\{\Psi_{\text{aug},1}, \ldots, \Psi_{\text{aug},r}\}, i \in \mathbb{N}$, en

$$\Psi_{\mathrm{aug},j} = \begin{bmatrix} A_j - B_j K & [B_j E\, 0\, 0\, \ldots\, 0] \\ 0 & S_{N,n_c} \end{bmatrix}, \qquad S_{N,n_c} = \begin{bmatrix} 0 & I & 0 & \ldots & 0 \\ 0 & 0 & I & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \\ 0 & 0 & 0 & & I \\ 0 & 0 & 0 & \ldots & 0 \end{bmatrix}, \quad j = 1, \ldots, r,$$

waarbij $S_{N,n_c} \in \mathbb{R}^{N.n_c \times N.n_c}$ de $N.n_c$-dimensionale doorschuifmatrix met doorschuifoperaties van lengte n_c is. Het systeem is onderhevig aan beperkingen

$$A_x \Gamma_x x_{\mathrm{aug}}(i) \leq \mathbf{1}, \qquad A_u \Gamma_u x_{\mathrm{aug}}(i) \leq \mathbf{1}, \qquad \forall i \in \mathbb{N},$$

waarbij $\Gamma_x = [I\ 0\ \ldots\ 0], \Gamma_u = [-K\ E\ 0\ \ldots\ 0]$ en $x_{\mathrm{aug}}(i) = [x(k+i|k); c(k+i|k); \ldots; c(k+i+N-1|k)]$. Het resulterende algoritme (Algoritme 4.7) heeft een significant groter werkingsgebied dan de bestaande algoritmes (bv. [70]) en kan geformuleerd worden als een *Kwadratisch Programma* (*Quadratic Program*, QP) in plaats van een *Semi-Definiet Programma* (*Semi-Definite Program*, SDP). Het nadeel is dat het aantal beperkingen van het resulterende QP nog steeds exponentieel kan toenemen als een functie van N. Dit zal in Hoofdstuk 5 verholpen worden. Figuren 4.14-4.17 geven de performantie weer van het nieuwe algoritme. Tabel 4.3 toont aan dat het aantal beperkingen exponentieel kan toenemen als functie van de lengte van de horizon N.

Het is belangrijk op te merken dat ook hier de beperkingen van het MPC optimalisatieprobleem kunnen bepaald worden als een invariante verzameling voor een uitgebreid systeem. Het verschil met Algoritme 4.6 is de manier waarop het uitgebreide systeem opgebouwd wordt en de definities van Γ_x en Γ_u. Dit toont enerzijds opnieuw het belang aan van het bestaan van efficiënte algoritmes voor het opstellen van invariante verzamelingen en suggereert anderzijds het bestaan van een unificerend theoretisch raamwerk waarin beide methodes kunnen ondergebracht worden.

Hoofdstuk 5: Invariante Verzamelingen met Gereduceerde Complexiteit voor Robuuste MPC

Hoofdstuk 2 introduceerde methodes voor het opstellen van polyhedrale invariante verzamelingen voor LPV systemen, dewelke in Hoofdstuk 4 gebruikt werden voor het verbeteren van de performantie en het vergroten van het werkingsgebied van enkele robuuste MPC algoritmes. In een aantal gevallen bleek het aantal beperkingen dat nodig is voor het beschrijven van de bekomen invariante verzameling ongunstig op te schalen. Dit hoofdstuk verkent twee belangrijke technieken voor het reduceren van de complexiteit van de bekomen verzamelingen: snoeien *(pruning) en* trimmen *(trimming). Er wordt aangetoond dat op deze manier onder bepaalde voorwaarden de complexiteit significant kan gereduceerd worden. Deze samenvatting beperkt zich tot het geven van een korte beschrijving van de twee belangrijkste algoritmes en het toelichten van de implicaties van deze algoritmes voor het robuuste MPC algoritme met quasi-oneindige horizon beschreven in het vorige hoofdstuk (Algoritme 4.7).*

Complexiteitsreductie door middel van snoeien

Zoals uiteengezet in Sectie 2.3.2 kunnen de beperkingen die een invariante verzameling definiëren ondergebracht worden in een boomstructuur. Het belangrijke verschil tussen LTI en LPV systemen is het feit dat bij LTI systemen geen vertakkingen kunnen voorkomen in deze boomstructuur. Bij LPV systemen kunnen maximaal r vertakkingen optreden per knoop. Dit is de belangrijkste oorzaak van het typisch significant hogere aantal beperkingen dat nodig is om polyhedrale invariante verzamelingen voor LPV systemen te beschrijven. In deze sectie wordt Algoritme 5.2 kort toegelicht. Dit algoritme heeft als doel om het aantal vertakkingen per knoop te reduceren, zodat het totaal aantal beperkingen daalt.

Figuur 5.1 toont deze structuur aan de hand van een numeriek voorbeeld en geeft de geometrische interpretatie. Het is duidelijk dat bij het optreden van een splitsing in de boomstructuur twee erg gelijkaardige beperkingen gegenereerd worden. Bovendien is het duidelijk dat indien een van beide beperkingen met een kleine factor strikter genomen wordt (d.i. parallel verplaatst worden in de richting van de oorsprong), de andere beperking redundant wordt en kan weggelaten worden. Lemma 5.2 toont aan dat deze factor exact kan bepaald worden door het oplossen van een LP. Algoritme 5.2 maakt hiervan gebruik om te komen tot polyhedrale invariante verzamelingen met een gereduceerd aantal beperkingen. Een parameter γ wordt gebruikt om aan te geven wat de maximale factor is waarmee een beperking mag aangepast worden. Op deze manier kan een afweging gemaakt worden tussen een maximaal volume en een minimale complexiteit.

Figuren 5.2a en 5.2b geven het resultaat weer van deze methodiek, wanneer toegepast op het numerieke voorbeeld van Figuur 5.1. Sectie 5.4.2 bespreekt de werking van dit algoritme wanneer het toegepast wordt voor het bepalen van invariante verzamelingen

voor de robuuste MPC algoritmes uit het vorige hoofdstuk. Een belangrijke vaststelling is het feit dat in bepaalde gevallen (zie Figuur 5.9) het aantal beperkingen dat met Algoritme 5.2 bekomen wordt voor $\mathcal{P}$-RMPC (Algoritme 4.7) asymptotisch lineair toeneemt als een functie van N, terwijl met Algoritme 2.4 uit Hoofdstuk 2 een exponentieel stijgend aantal beperkingen bekomen wordt voor $\mathcal{P}$-RMPC. Later in dit hoofdstuk wordt aangetoond onder welke voorwaarden dit schalingsgedrag gegarandeerd kan worden.

Complexiteitsreductie door middel van trimmen

Terwijl in de vorige sectie aangetoond werd hoe het aantal beperkingen van polyhedrale invariante verzamelingen voor LPV systemen kan gereduceerd worden door het verminderen van het aantal vertakkingen in de boomstructuur van de verzameling, wordt in deze sectie uitgelegd hoe een gelijkaardig effect kan bekomen worden door het reduceren van de diepte van deze boomstructuur.

Lemma 5.4 toont aan dat, gegeven twee autonome LPV systemen gedefinieerd door de volgende onzekerheidspolytopen:

$$\Omega_1 \triangleq \mathrm{Co}\{\Phi_1, \ldots, \Phi_r\},$$
$$\Omega_2 \triangleq \mathrm{Co}\{\Phi'_1, \ldots, \Phi'_r\},$$

waarbij de matrices $\Phi'_i, i = 1, \ldots, r$ gedefinieerd zijn als

$$\Phi'_i = (1 + c)\Phi_i - cI, \qquad i = 1, \ldots, r,$$

met $c \in \mathbb{R}^+$, elke verzameling $\mathcal{S} \in \mathbb{R}^{n_x}$ die invariant is voor het LPV systeem gedefinieerd door Ω_2, ook gegarandeerd invariant is voor het LPV systeem gedefinieerd door Ω_1.

Dit lemma laat dus toe, gegeven een LPV systeem gedefinieerd door onzekerheidspolytoop Ω_1, om een invariante verzameling voor dit systeem te bepalen door Algoritme 2.4 of Algoritme 5.2 toe te passen op het systeem gedefinieerd door onzekerheidspolytoop Ω_2. Dit geeft een bijkomende vrijheidsgraad c, die door de gebruiker vrij kan gekozen worden en die toelaat om de waarde van de JSR van het systeem te beïnvloeden, dewelke door middel van Theorema 2.2 gekoppeld kan worden aan de diepte van de boomstructuur van de verzameling. De parameter c kan dus gebruikt worden om de diepte van de boomstructuur van de bekomen verzameling aan te passen en op die manier het aantal beperkingen dat de verzameling beschrijft. Het nadeel is dat het volume van de bekomen verzameling kleiner is.

Figuren 5.2c en 5.2d geven het resultaat weer van deze methodiek, wanneer toegepast op het numerieke voorbeeld van Figuur 5.1. Sectie 5.4.2 bespreekt de werking van dit algoritme wanneer het toegepast wordt voor het bepalen van invariante verzamelingen voor de robuuste MPC algoritmes uit het vorige hoofdstuk.

Lineair opschalen van $\mathcal{P}$-RMPC (Algoritme 4.7)

Zoals reeds opgemerkt eerder in dit hoofdstuk, leidt het gebruik van Algoritme 5.2 voor het bepalen van invariante verzamelingen voor $\mathcal{P}$-RMPC tot significante

complexiteitsreducties vergeleken met Algoritme 2.4. Deze empirische vaststelling kan echter ook bewezen worden vanuit theoretisch standpunt. Het belang van dit resultaat wordt weergegeven in Figuur 5.3. Deze figuur verduidelijkt dat dit resultaat ertoe leidt dat zowel het aantal variabelen als het aantal beperkingen van het MPC optimalisatie probleem van $\mathcal{P}$-RMPC lineair opschaalt in functie van N, wanneer Algoritme 5.2 gebruikt voor het bepalen van de nodige invariante verzamelingen.

Lemma 5.3 toont aan dat de factor waarmee beperkingen strikter gemaakt worden in Algoritme 5.2 begrensd is, indien dit algoritme toegepast wordt op het uitgebreide autonome systeem dat gebruikt wordt tijdens de ontwerpfase van $\mathcal{P}$-RMPC. Het belangrijkste aspect is dat deze bovengrens wel onafhankelijk is van N, maar dat deze bovengrens enkel geldt indien de onzekerheid die aanwezig is in het te regelen systeem voldoende klein is en indien de JSR van de systeemmatrices ook voldoende klein is in vergelijking met deze onzekerheid.

Theorema 5.5 maakt gebruik van Lemma 5.3 om daarna te bewijzen dat het aantal beperkingen van $\mathcal{P}$-RMPC linear toeneemt in functie van N.

Het dient vermeld te worden dat de bekomen voorwaarden erg conservatief zijn en dat in vele gevallen ook lineair schalingsgedrag geobserveerd kan worden indien aan deze voorwaarde niet voldaan is. Het nadeel is dat dit het schalingsgedrag van dit algoritme in deze brede klasse van de gevallen erg onvoorspelbaar maakt.

De belangrijkste implicatie van dit resultaat is dat in vele gevallen een significant grotere horizon kan gebruikt worden, waardoor het werkingsgebied van de regelaar kan vergroot worden. Op deze manier wordt de volumereductie die gepaard gaat met de complexiteitsreductie van de invariante set gecompenseerd en kan zelfs een significant groter werkingsgebied bekomen worden dan wanneer geen complexiteitsreductie zou toegepast worden.

Hoofdstuk 6: Controle-Invariante Verzamelingen met Gereduceerde Complexiteit

Het doel van dit hoofdstuk is het uitbreiden van de algoritmes van Hoofdstuk 5 naar het construeren van controle-invariante verzamelingen. Dergelijke verzamelingen kunnen gezien worden als een uitbreiding van invariante verzamelingen naar systemen met ingangen, in plaats van autonome systemen, dewelke geen ingangen hebben. Zoals in Hoofdstuk 7 aangetoond wordt, kunnen controle-invariante verzamelingen ook gebruikt worden in MPC algoritmes, wat de reden is voor het verfijnen van de bestaande algoritmes voor het opstellen van dergelijke verzamelingen. De moeilijkheid in het construeren van controle-invariante verzamelingen is dat er een projectiestap moet ingebouwd worden in de algoritmes waardoor de rekencomplexiteit opnieuw toeneemt. Naast het uitbreiden van de algoritmes uit Hoofdstuk 5 naar deze probleemstelling, zal dan ook getracht worden de rekencomplexiteit van deze bijkomende projectiestap te reduceren.

Definities

In dit hoofdstuk beschouwen we LPV modellen van dezelfde vorm als deze beschreven in Hoofdstuk 3, onderhevig aan lineaire ingangs- en toestandsbeperkingen respectievelijk beschreven door verzamelingen $\mathcal{U}$ en $\mathcal{X}$. Een verzameling S is een controle-invariante verzameling met betrekking tot dit systeem indien aan volgende voorwaarde voldaan is:

$$\forall x \in \mathcal{S}, \exists u(x) \in \mathcal{U}: \quad Ax + Bu \in \mathcal{S}, \qquad \forall [A\ B] \in \Omega.$$

In woorden is een verzameling $\mathcal{S}$ dus controle-invariant als voor elke huidige toestand binnen deze verzameling er een overeenkomstige toelaatbare ingangsvector bestaat die de toestand binnen de verzameling houdt. In dit hoofdstuk is het de bedoeling om, gegeven een LPV systeem en ingangs- en toestandsbeperkingen, de maximale controle-invariante verzameling $\mathcal{S}$ te vinden, waarvoor geldt dat $\mathcal{S} \subseteq \mathcal{X}$.

Dergelijke verzamelingen kunnen op gelijkaardige manier als invariante verzamelingen iteratief geconstrueerd worden, met als verschilpunt dat in elke iteratie een projectiestap moet uitgevoerd worden die een $(n_x + n_u)$-dimensionale verzameling projecteert naar een n_x-dimensionale verzameling. Deze methodologie is conceptueel samengevat in Algoritme 6.1 en schematisch weergegeven in Figuur 6.1. De methodes van het snoeien en trimmen, die reeds uitgewerkt werden in Hoofdstuk 5, kunnen gebruikt worden voor complexiteitsreducties tijdens het genereren van de $(n_x + n_u)$-dimensionale verzamelingen vanuit de n_x-dimensionale verzamelingen uit de vorige iteratie. Projecties met gereduceerde complexiteit worden gebruikt voor de andere stappen van het algoritme afgebeeld in Figuur 6.1.

Complexiteitsreductie door middel van snoeien

Alhoewel het niet meteen mogelijk is bij controle-invariante verzamelingen om de beperkingen onder te brengen in een boomstructuur, omwille van de extra projectiestap, is het nog steeds mogelijk om de methode van het snoeien toe te passen. Het is immers mogelijk om te detecteren wanneer een enkele beperking in iteratie i aanleiding geeft tot meerdere beperkingen in iteratie $i+1$. In dit geval kan identiek dezelfde methode toegepast worden om beperkingen redundant te maken door een andere beperking strikter te maken. Hiervoor kan men opnieuw een beroep doen op Lemma 5.2 om de factor te bepalen waarmee een bepaalde beperking strikter moet gemaakt worden. Deze werkwijze is beschreven in Algoritme 6.3.

Een belangrijk verschilpunt met Algoritme 5.2 is dat het in dit geval minder voor de hand ligt om een convergentiebewijs te geven. Simulaties leren echter dat er zelden convergentieproblemen optreden voor gangbare parameterwaarden.

Complexiteitsreductie door middel van trimmen

Naar analogie aan de vorige sectie kan ook de methode van het trimmen uitgebreid worden naar de constructie van controle-invariante verzamelingen. Theorema 6.2, dat een uitbreiding is van Theorema 5.4, toont aan dat, gegeven twee LPV systemen gedefinieerd door de volgende onzekerheidspolytopen

$$\Omega_1 \triangleq \mathrm{Co}\{[A_1\ B_1], \ldots, [A_r\ B_r]\},$$
$$\Omega_2 \triangleq \mathrm{Co}\{[A_1'\ B_1'], \ldots, [A_r'\ B_r']\},$$

waarbij de matrices $[A_i'\ B_i'], i = 1, \ldots, r$ gedefinieerd zijn als

$$[A_i'\ B_i'] = (1+c)[A_i\ B_i] - c[I\ 0], \qquad i = 1, \ldots, r,$$

met $c \in \mathbb{R}^+$, elke verzameling $\mathcal{S}$ die controle-invariant is voor het LPV systeem gedefinieerd door onzekerheidspolytoop Ω_2, ook gegarandeerd controle-invariant is voor het LPV systeem gedefinieerd door Ω_1.

Net zoals bij de constructie van invariante verzamelingen kan hier de parameter c gebruikt worden voor het beïnvloeden van de complexiteit van de resulterende verzameling.

Projecties met gereduceerde complexiteit

Zoals eerder in dit hoofdstuk aangegeven, dient bij het construeren van controle-invariante verzamelingen een bijkomende stap uitgevoerd te worden in elke iteratie, namelijk een projectiestap. In de praktijk blijkt dat deze stap de belangrijkste factor is die de rekencomplexiteit bepaalt. De meest gebruikte methode voor het berekenen van projecties van polyhedrale verzamelingen die beschreven zijn als de doorsnede van halfruimten, is *Fourier-Motzkin* eliminatie. Deze methode projecteert de opgegeven verzameling dimensie per dimensie en is beschreven in Appendix B. De rekencomplexiteit stijgt exponentieel naargelang de dimensionaliteit van de verzamelingen toeneemt. Een alternatief is de ESP-methode [62]. Deze is in staat

rechtstreeks meer-dimensionale projecties uit te voeren, maar heeft ook een ongunstig schalingsgedrag.

Sectie 6.2.4 introduceert een methode om inwendige en uitwendige benaderingen te bepalen van projecties van polyhedrale verzamelingen. Het doel is om zowel de complexiteit van de resulterende verzameling te reduceren, alsook om de rekencomplexiteit van het bepalen van deze projecties te reduceren. In deze samenvatting gaan we enkel in op het bepalen van inwendige benaderingen omdat deze het meeste nut hebben voor het bepalen van controle-invariante verzamelingen.

Algoritme 6.5 beschrijft een methode voor het bepalen van inwendige benaderingen van projecties door gebruik te maken van de methode van het snoeien. Telkens nieuwe beperkingen opgesteld worden, wordt gecontroleerd met behulp van Lemma 5.2 of deze beperking niet strikter kan gemaakt worden teneinde andere beperkingen redundant te maken. Simulaties tonen aan dat voor het bepalen van inwendige benaderingen van willekeurige polytopen dit algoritme geen significante voordelen biedt ten opzichte van het bepalen van exacte projecties.

Algoritme 6.5 kan echter ook gebruikt worden voor het uitvoeren van de projectiestap tijdens het bepalen van controle-invariante verzamelingen, zoals Algoritme 6.2 beschrijft. Figuren 6.2 en 6.3 tonen aan dat in deze context het gebruik van projecties met gereduceerde complexiteit nuttig kan blijken. Voor hoger-dimensionale systemen, zoals deze beschreven in Hoofdstuk 8, blijken de nieuwe algoritmes echter nog onvoldoende efficiënt om praktisch bruikbaar te zijn.

Hoofdstuk 7: Robuuste MPC met Controle-Invariante Verzamelingen

In dit hoofdstuk wordt besproken op welke manier de controle-invariante verzamelingen uit het vorige hoofdstuk kunnen gebruikt worden voor het formuleren van verbeterde MPC algoritmes. In een eerste deel wordt interpolatie-gebaseerde MPC uitgebreid naar niet-lineaire regelwetten, wat toelaat om te interpoleren tussen regelaars die geïnduceerd zijn door controle-invariante verzamelingen en de MPC regelaars beschreven in Hoofdstuk 4. Hierdoor kan het werkingsgebied van de regelaars significant uitgebreid worden, indien we beschikken over een controle-invariante verzameling voor het te regelen systeem. In een tweede deel wordt een eenvoudige methode voorgesteld om controle-invariante verzamelingen te gebruiken voor het omgaan met opgelegde beperkingen bij regelproblemen met volgobjectief.

Veralgemeende interpolatie tussen niet-lineaire regelaars

De algoritmes voor veralgemeende interpolatie die tot nu toe beschreven werden in de literatuur (zie bv. [3, 122]) gelden enkel voor lineaire systemen en lineaire regelwetten. In Sectie 7.2 wordt echter aangetoond dat veralgemeende interpolatie ook mogelijk is bij niet-lineaire regelwetten, zolang het te regelen systeem lineair is. Deze techniek kan gebruikt worden voor het vergroten van het werkingsgebied van de robuuste MPC algoritmes uit Hoofdstuk 4, omwille van de volgende twee vaststellingen:

1. Enerzijds toont Sectie 7.3.1 aan dat voor elke controle-invariante verzameling een regelaar kan gedefinieerd worden ten opzichte waarvan deze verzameling een invariante verzameling is. Deze regelwet kan ook op een voor de hand liggende manier geconstrueerd worden en wordt de geïnduceerde regelwet genoemd. We verwijzen naar Lemma 7.2 voor verdere details.

2. Anderzijds toont Sectie 7.3.2 aan dat de werkingsgebieden van MPC regelaars, waarvan recursieve oplosbaarheid gegarandeerd is, invariante verzamelingen zijn voor deze regelaars. We verwijzen naar Lemma 7.3 voor verdere details.

Deze twee resultaten tonen dus aan dat controle-invariante verzamelingen en hun geïnduceerde regelaars enerzijds en recursief oplosbare MPC algoritmes en hun werkingsgebied anderzijds, bruikbaar zijn als invariante verzameling en niet-lineaire regelaar voor gebruik in veralgemeende interpolatie voor niet-lineaire regelaars. Dit wordt verder uitgewerkt in Sectie 7.3.1, waardoor een groter werkingsgebied bekomen wordt.

Dit resultaat toont aan hoe het werkingsgebied van recursief oplosbare MPC regelaars kan uitgebreid worden tot het grootst mogelijke werkingsgebied dat kan bekomen worden voor het te regelen systeem en de opgelegde beperkingen. Het is echter wel duidelijk (zie Figuur 7.2) dat naarmate de toestand van het systeem zich verder van de oorsprong bevindt, men minder rekening kan houden met het te

minimaliseren regelobjectief terwijl de opgelegde beperkingen een steeds grotere rol gaan spelen.

Controle-invariante verzamelingen voor regelproblemen met volgobjectief

Deze sectie toont voor de volledigheid aan hoe met behulp van controle-invariante verzamelingen op een theoretisch onderbouwde manier kan omgegaan worden met opgelegde beperkingen bij regelproblemen met volgobjectief. De klassieke stabiliteitstheorie, die gebruik maakt van invariante verzamelingen voor het garanderen van recursieve oplosbaarheid, is enkel geldig voor regelproblemen waar het systeem naar een vast instelpunt geregeld wordt. Voor het geval men een regelprobleem met volgobjectief tracht op te lossen is het niet mogelijk om met hetzelfde theoretische raamwerk recursieve oplosbaarheid te garanderen.

In Sectie 7.4 wordt aangetoond dat ongeacht de gebruikte regelaar, het mogelijk is om met behulp van controle-invariante verzamelingen te garanderen dat aan alle opgelegde beperkingen voldaan wordt, ook in het geval van regelproblemen met volgobjectief. In essentie zorgt de nieuwe methode (Algoritme 7.3) ervoor dat er enkel ingangen kunnen aangelegd worden aan het systeem die de toestand binnen de controle-invariante verzameling houden, hetgeen garandeert dat aan alle ingangs- en toestandsbeperkingen voldaan wordt. Asymptotische stabiliteit is niet gegarandeerd in het algemeen en hangt af van de specifieke regelaar die gebruikt wordt.

Figuren 7.9-7.12 tonen de goede werking van deze methode en maken een vergelijking met meer eenvoudige technieken. Bovendien heeft de nieuwe methode slechts een kleine verhoging in rekencomplexiteit tot gevolg. In elke iteratie moet er een QP van aanvaardbare afmetingen opgelost worden.

Het moet wel opgemerkt worden dat Algoritme 7.3 geen allesomvattende methode is en slechts een eerste aanzet vormt tot meer algemene methodes, die ook stabiliteit en goed volggedrag garanderen.

Hoofdstuk 8: Gevallenstudies

Dit hoofdstuk heeft als doel de verschillende algoritmes die besproken worden in deze thesis te testen op meer realistische en meer grootschalige voorbeelden. Twee voorbeelden worden beschouwd: een mechanische installatie voor het regelen van de spanning in staalplaten tijdens het walsproces enerzijds en een chemische reactor voor de aanmaak van copolymeren anderzijds.

Regeling van de trekspanning in metaalplaten

Sectie 8.1 beschouwt het model van een mechanische installatie die gebruikt wordt in staalwalserijen voor het regelen van de trekspanning van staalplaten tussen twee opeenvolgende stadia van het walsproces. De spanning van twee elektrische motoren kan aangepast worden met als doel een bepaalde gewenste trekspanning te bekomen. Figuren 8.1 en 8.2 geven dit systeem schematisch weer. Een te lage spanning leidt tot een onstabiele doorvoer van de staalplaten, terwijl een te hoge spanning kan leiden tot ongewenste afwijkingen in de dikte en breedte van de platen. Bovendien is het wenselijk om de hoek van de mechanische arm (de *looper*) die de trekspanning helpt regelen rond een bepaalde vaste gewenste waarde te houden. Het proces wordt beschreven door een model met 2 ingangen, 6 toestanden en 2 uitgangen. Afhankelijk van de hoek van de mechanische arm heeft het systeem een ander dynamisch gedrag. Om dit op te vangen wordt een LPV model opgesteld gebaseerd op linearisaties van het systeemgedrag rond twee verschillende waarden van deze hoek en wordt op basis hiervan een regelaar ontworpen.

Een vergelijking tussen de algoritmes uit Hoofdstuk 4 toont dat $\mathcal{P}$-RMPC met $N = 5$ de laagste rekenkost, maar ook een relatief klein werkingsgebied heeft en als gevolg daarvan al snel problemen ondervindt wanneer het systeem verstoord wordt. GIMPC2 (Algoritme 4.6) leidt tot het beste resultaten en heeft bovendien een relatief lage rekencomplexiteit vergeleken met $\mathcal{P}$-RMPC met $N = 25$.

Het opstellen van controle-invariante verzamelingen voor het te regelen systeem bleek minder succesvol dan het opstellen van invariante verzamelingen. De controle-invariante verzamelingen bleken immers kleiner dan de bekomen werkingsgebieden van de algoritmes uit Hoofdstuk 4. Deze konden dus niet gebruikt worden voor het verder uitbreiden van dit werkingsgebied.

Regeling van een copolymerisatie-reactor

Sectie 8.1 beschouwt het model van een copolymerisatie-reactor. Het doel van het proces is om vanuit twee monomeren A en B een copolymeer te produceren. De toevoersnelheid van de reagentia en enkele andere chemische stoffen die de copolymerisatie-reactor beïnvloeden dient optimaal geregeld te worden teneinde een copolymeer met de gewenste eigenschappen te produceren. Figuur 8.9 geeft dit systeem schematisch weer. Het proces wordt beschreven door een model met 6 ingangen, 12 toestanden en 4 uitgangen. Er wordt een LPV model opgesteld gebaseerd op twee modellen van het systeem die geldig zijn voor twee verschillende samenstelling

van de reactorinhoud. Op deze manier worden verschillen in de dynamica ten gevolge van verstoringen wat betreft deze samenstelling opgevangen.

Het belangrijkste resultaat van deze simulatie is dat $\mathcal{P}$-RMPC in combinatie met Algoritme 5.2 resulteert in een lineair toenemend aantal beperking in functie van N, hetgeen toelaat een horizon $N = 25$ te gebruiken. Wanneer Algoritme 2.4 gebruikt wordt is een horizon van $N = 10$ al een computationele uitdaging. Hierdoor kan een erg groot werkingsgebied bekomen worden, hetgeen onmogelijk zou zijn zonder het gebruik van polyhedrale invariante verzamelingen met gereduceerde complexiteit.

Het opstellen van controle-invariante verzamelingen bleek niet haalbaar voor een systeem van deze afmetingen. Zelfs het opstellen van een controle-invariante verzameling voor het geval $r = 1$ bleek reeds computationeel niet haalbaar. Dit illustreert dat er verder onderzoek nodig is naar efficiëntere algoritmes voor grootschalige systemen.

Hoofdstuk 9: Besluiten en Toekomstig Onderzoek

Besluiten

Algemeen

In deze thesis werden verschillende wiskundige technieken onderzocht om robuuste MPC regelaars te bekomen met voordelig schalingsgedrag en met niet-conservatieve behandeling van opgelegde beperkingen.

De nadruk lag op het gebruik van polyhedrale invariante verzamelingen in plaats van ellipsoïdale verzamelingen, omdat deze laatste weinig flexibiliteit bieden en resulteren in MPC optimalisatieproblemen met hoge rekencomplexiteit. Polyhedrale invariante verzamelingen bieden maximale flexibiliteit maar hebben het nadeel dat ze leiden tot algoritmes met exponentieel schalingsgedrag, wat verhindert om grootschalige systemen te regelen of om goede regelperformantie te bekomen door een lange horizon te gebruiken.

In deze thesis werden verschillende methodes beschreven die het mogelijk maken om de flexibiliteit van polyhedrale invariante verzamelingen uit te buiten en die resulteren in robuuste MPC algoritmes met verbeterde behandeling van opgelegde beperkingen en met verbeterd schalingsgedrag. Deze verbeteringen werden bekomen op basis van resultaten op twee gebieden: 1) op het niveau van het construeren van invariante verzamelingen voor gebruik in MPC, en 2) op het niveau van het opstellen van algoritmes voor robuuste MPC. Deze twee gebieden worden apart toegelicht in de volgende secties.

Robuuste modelgebaseerde predictieve controle

In deze thesis werden verschillende algorithmische bijdragen geleverd op het gebied van robuuste MPC. Deze bijdragen moeten beschouwd worden in het licht van de conceptuele beschouwingen die in Hoofdstuk 3 besproken werden. In dit hoofdstuk werd het belang besproken van terugkoppeling in de ingangsequentie over dewelke de MPC optimalisaties worden uitgevoerd. Er werd aangetoond dat verschillende robuuste MPC algoritmes uit de literatuur incorrect zijn omwille van de afwezigheid van dit concept van terugkoppeling in de ingangsequenties. Deze aanpak, het gesloten-lus paradigma genaamd, resulteert ook in verbeterde regelperformantie en impliceert, zoals werd aangetoond in Sectie 4.3, niet noodzakelijk een verhoogd aantal optimalisatievariabelen. Hoofdstuk 5 ging nog een stap verder en toonde aan dat in het geval van gesloten-lus MPC met quasi-oneindige horizon, invariante verzamelingen met gereduceerde complexiteit de rekencomplexiteit zelfs kunnen verlagen vergeleken met open-lus robuuste MPC. Dit werd mogelijk gemaakt door het feit dat Algoritme 4.7 het mogelijk maakt om de beperkingen van het MPC optimalisatieprobleem te bepalen als een invariante verzameling van een uitgebreid autonoom systeem. Volgend kader vat deze vaststellingen samen.

Zoals wordt aangetoond in deze thesis, is het gesloten-lus MPC paradigma van primair belang voor recursieve oplosbaarheid en stabiliteit van robuuste MPC algoritmes. In tegenstelling tot wat algemeen gedacht wordt, impliceert het gebruik van gesloten-lus predicties niet noodzakelijk een verhoging van de rekencomplexiteit van de MPC optimalisatieproblemen, maar zorgt de specifieke structuur ervoor dat significante complexiteitsreducties kunnen bekomen worden. Er wordt aangetoond dat de resulterende algoritmes lineair schalingsgedrag vertonen in plaats van het exponentiële schalingsgedrag van bestaande algoritmes.

Andere algorithmische bijdragen kunnen gesitueerd worden in verschillende stadia van het MPC ontwerpproces. Eerst werd in Sectie 4.1 getoond hoe polyhedrale invariante verzamelingen de synthese van robuuste lineaire terugkoppelregelaars kunnen verbeteren. Deze kunnen later gebruikt worden als eindregelaar of als lokale regelaar in robuuste MPC algoritmes. In secties 4.2 en 4.3 werd verder aangetoond hoe twee bestaande MPC paradigma's (MPC met quasi-oneindige horizon (RMPC) en interpolatie-gebaseerde MPC (GIMPC), beide gesloten-lus paradigma's) ook hun voordeel kunnen halen uit het gebruik van polyhedrale invariante verzamelingen. GIMPC werd bovendien verder verbeterd in Sectie 4.2.5 om nog beter kunnen omgaan met opgelegde beperkingen. De auteur zou willen benadrukken dat de synthese van de RMPC- en GIMPC2-algoritmes op quasi indentieke manier gebeurt en enkel bestaat uit het construeren van een invariante verzameling en een kwadratische Lyapunov functie voor een speciaal geconstrueerd uitgebreid systeem. Beide methodes verschillen in de manier waarop dit systeem geconstrueerd wordt en hoe de aan te leggen ingangsvector afhangt van de uitgebreide toestandsvector.

De gemeenschappelijk structuur die aanwezig is in de ontwerpprocedures van RMPC en GIMPC2 suggereert rechtstreeks het bestaan van een meer algemeen MPC raamwerk, waar de klasse van kandidaat ingangsequenties geparametriseerd is door middel van een autonoom lineair systeem, waarvan de toestanden gebruikt worden als optimalisatievariabelen in het MPC optimalisatieprobleem.

Deze verschuiving van FIR-type naar IIR-type parametrisaties van de ingangsequenties is de theoretische projectie van het in de industriële praktijk gangbare gebruik om de kandidaat ingangsequenties te parametriseren als stuksgewijs constante functies, met toenemende intervallengtes naarmate men zich verder bevindt binnen de controlehorizon.

De oorsprong van deze verschuiving gaat terug tot [131] met de invoering van een eindregelaar om de ingangsequentie voorbij de horizon te parametriseren. De resultaten in [123] vergrootten verder het belang van deze lokale regelaar door hem ook een rol te laten spelen binnen de horizon. Meer recente bijdragen [29, 58, 61] tonen aan dat deze trend zich verder doorzet.

Een laatste bijdrage die besproken werd in Hoofdstuk 7 is de uitbreiding van GIMPC naar niet-lineaire regelaars, wat toelaat om te interpoleren tussen MPC regelaars en

regelaars die geïnduceerd worden door een controle-invariante verzameling. Dit laat toe om het werkingsgebied van eender welke recursief oplosbare MPC regelaar uit te breiden naar het theoretisch maximum zonder een significante meerkost op gebied van rekentijd. Het gebruik van controle-invariante verzamelingen om te kunnen garanderen dat voldaan wordt aan de opgelegde beperkingen bij regelproblemen met volgobjectief werd ook kort behandeld. Hiermee werd verder het potentieel van controle-invariante verzamelingen geïllustreerd voor een bredere klasse van regelproblemen dan wat mogelijk is met invariante verzamelingen.

Invariante verzamelingen

De centrale bijdrage van deze thesis op gebied van invariante verzamelingen is de introductie van wat men *bijna maximale* polyhedrale invariante verzamelingen kan noemen. De twee belangrijkste andere types van invariante verzamelingen binnen de klasse van polyhedrale verzamelingen (maximale invariante verzamelingen (*Maximal Admissible Sets*, MAS [52]) en invariante verzamelingen met lage complexiteit [75]) vormen twee extremen in het spectrum van afwegingen tussen een maximaal volume en een minimale complexiteit. De klasse van bijna-maximale invariante verzamelingen die in deze thesis geïntroduceerd werd, laat de gebruiker toe om deze afweging te variëren tussen deze twee extremen. Meer specifiek werd er aangetoond in deze thesis dat er typisch een verdedigbaar kleine volumereductie geobserveerd wordt in combinatie met significante complexiteitsreducties, waarbij exponentieel schalingsgedrag in bepaalde gevallen gereduceerd wordt tot lineair schalingsgedrag.

De constructie van bijna-maximale invariante verzamelingen werd aangepakt in Hoofdstuk 5 door middel van *snoeien* en *trimmen*, terwijl in Hoofdstuk 6 deze methodes werden uitgebreid naar de constructie van controle-invariante verzamelingen samen met de toevoeging van projecties met gereduceerde complexiteit. Een belangrijk aspect is dat enkel de maximaliteit van de resulterende verzamelingen opgeofferd wordt, maar dat de invariantie-eigenschap nog steeds exact geldt, waardoor de resulterende MPC algoritmes nog steeds theoretisch onderbouwd kunnen worden. Deze methode voor het opstellen van invariante verzamelingen, die gelijkaardig is aan regularisatie, is een volledig nieuwe aanpak die de gebruiker een extra ontwerpparameter biedt tijdens het ontwerp van robuuste MPC regelaars. Deze bevindingen kunnen als volgt worden samengevat.

> *De nieuwe regularisatie-achtige aanpak voor het construeren van polyhedrale invariante verzamelingen laat de gebruiker toe om een afweging te maken tussen een maximaal volume en een minimale complexiteit. De bekomen complexiteitsreducties laten toe om polyhedrale invariante verzamelingen op te stellen voor hoger dimensionale systemen dan voorheen mogelijk was. Dit laat op zijn beurt toe om robuuste MPC algoritmes te ontwerpen met significant* **grotere** *werkingsgebieden.*

Tenslotte kan geconcludeerd worden dat de verschillende methodes voor het construeren van invariante verzamelingen met gereduceerde complexiteit slechts initiële

stappen zijn in het onbekende. Ongetwijfeld zijn er nog vele andere, betere en meer elegante oplossingen bedenkbaar binnen het raamwerk dat in Hoofdstukken 5 en 6 werd geïntroduceerd. Het is pas in het laatste decennium dat er in die mate onderzoek verricht is naar invariante verzamelingen als momenteel het geval is en vele inzichten moeten nog vergaard worden. Deze observaties worden in het volgende kader samengevat.

Het raamwerk voor de constructie van (controle-)invariante verzamelingen met gereduceerde complexiteit dat geïntroduceerd werd in Hoofdstukken 5 en 6 laat ruimte voor vele vrijheidsgraden, waarvan vele slechts gedeeltelijk verkend zijn in deze thesis. Ook dienen nog vele eigenschappen van de nieuwe algoritmes uit deze thesis verder onderzocht te worden. Het inbrengen van bijkomende inzichten vanuit de computationele meetkunde zou potentieel kunnen leiden tot nieuwe doorbraken in deze twee gebieden.

Een voorbeeld hiervan is het feit dat de *combinatorische structuren* van polytopen (gedefinieerd door de zogenaamde *face lattice*) of het concept van *polaire* polytopen [146] in geen enkel van de beschreven algoritmes gebruikt zijn, terwijl het niet ondenkbeeldig is dat beschouwingen op basis van deze concepten zouden kunnen leiden tot belangrijke nieuwe inzichten in de theorie van invariante verzamelingen.

Toekomstig onderzoek

Er kunnen verschillende potentieel interessante onderwerpen voor toekomstig onderzoek onderscheiden worden, zowel op gebied van robuuste MPC als op gebied van invariante verzamelingen. Deze sectie licht enkele van de meest interessante denkrichtingen toe.

Robuuste modelgebaseerde predictieve controle

1. Zoals reeds is aangegeven in de conclusies, kan er een trend geobserveerd worden richting IIR-type parametrisaties van de ingangsequentie waarover de MPC regelaar optimaliseert. Het is ook duidelijk geworden dat twee algoritmes uit deze thesis (RMPC en GIMPC2) in dit raamwerk passen, waarbij de ingangsequenties bepaald worden door de dynamica van een lineair autonoom systeem. Een voor de hand liggend en potentieel erg interessant toekomstig onderzoeksonderwerp is daarom de ontwikkeling van een theoretisch raamwerk voor dergelijke algoritmes. Dit raamwerk zou bij voorkeur aspecten omvatten als robuustheid, uitgangsterugkoppeling, ISS stabiliteit, computationele vertragingen, enz...

2. Hoofdstuk 7 bevat reeds enkel preliminaire resultaten wat betreft regelproblemen met volgobjectief om te illustreren dat controle-invariante verzamelingen nut hebben in een dergelijke context. Belangrijke aspecten zoals robuuste stabiliteit, gegarandeerde volgperformantie, enz... werden echter nog niet behandeld. Een mogelijk toekomstig onderzoeksonderwerp is de ontwikkeling van

een stabiliteitsraamwerk voor dergelijke regelproblemen, op basis van controle-invariante verzamelingen. De combinatie van dit raamwerk met het hierboven beschreven onderzoeksonderwerp zou ook potentieel interessante resultaten kunnen opleveren.

Invariante verzamelingen

1. Een van de problemen die beschouwd wordt in deze thesis is het bepalen van de grootste invariante verzameling die binnen een opgegeven verzameling ligt. Uitbreidingen naar systemen met begrensde verstoringen worden kort aangehaald, maar worden niet in detail uitgewerkt. Een bijkomende concept dat relevant wordt in de aanwezigheid van begrensde verstoringen is dat van de kleinst mogelijk invariante verzameling. De grootte van een invariante verzameling is immers naar boven toe begrensd door de opgelegde toestandsbeperkingen, terwijl ze naar onder toe begrensd is door de grootte van de begrensde verstoringen. Wanneer echter gebruik gemaakt wordt van de polaire equivalenten van deze drie verzamelingen, worden deze relaties omgekeerd. Dit indiceert dat er mogelijk interessante eigenschappen en verbanden bestaan die uitgebuit kunnen worden om op een meer efficiënte manier maximale en minimale invariante verzamelingen te construeren. Het opstellen van deze verbanden en het formuleren van algoritmes voor het synthetiseren van invariante verzamelingen, gebruik makende van deze verbanden is een potentieel interessant onderwerp voor toekomstig onderzoek.

2. In deze thesis werden de methodes voor het construeren van invariante verzamelingen met gereduceerde complexiteit uitgebreid naar het construeren van controle-invariante verzamelingen. Er werden echter nog geen resultaten besproken wat betreft convergentie van deze algoritmes en ook het te verwachten schalingsgedrag in functie van de dimensionaliteit van het beschouwde systeem is nog slecht begrepen. Verder onderzoek op dit gebied is noodzakelijk.

3. Een derde interessante richting voor toekomstig onderzoek is het uitbreiden van de bestaande resultaten naar meer algemene klassen van systemen, zoals hybriede systemen of stuksgewijs affiene systemen. Ook de uitbreiding naar de context van *gain scheduling* is potentieel interessant. Algoritmes voor deze aanverwante problemen zijn reeds gekend, maar hun schalingsgedrag naar hoger dimensionale systemen is in het algemeen ongunstig.

Chapter 1

Introduction

"He who controls the past controls the future.
He who controls the present controls the past."

– George Orwell (1903-1950) –

Model based Predictive Control (MPC) is a control paradigm that has gained widespread acceptance in industry and has therefore received increasing amounts of attention from the academic world in the last few decades. As a result it is currently being regarded as the prime advanced process control (APC) method for a wide class of industrial processes. In this chapter the main reasons behind this steady rise are explained as well as the basic characteristics of the methodology. To form the basis for future chapters the necessary mathematical foundation is put in place and the two most important mathematical tools – convex optimization and computational geometry – are explained. The chapter finishes by clarifying the structure of this thesis by giving a chapter-by-chapter overview.

1.1 Process control

In 1788 James Watt introduced the concept of the *centrifugal governor* (Figure 1.1) to improve the reliability of the steam engines he was continuously fine tuning. The device consisted of two rotating weights connected to the main steam valve of the engine, thereby preventing runaways of the machine by gradually closing the valve in case of excessive rotational speed. This improvement, among others, resulted in the widespread adoption of the Watt steam engine across Europe and helped drive the industrial revolution at the end of the 18th and the beginning of the 19th century.

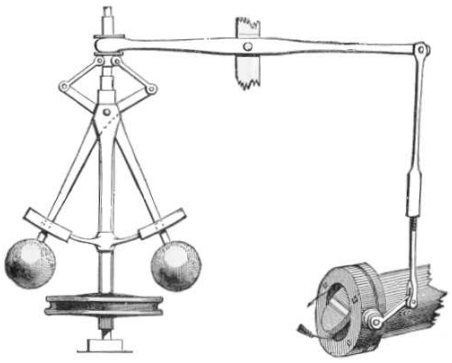


Figure 1.1: Centrifugal governor invented by James Watt in 1788 for controlling the rotary speed of a steam engine. (image taken from [127])

In this way Watt introduced the first well-known type of what is now called a *proportional controller*. Indeed, the steam flow (Manipulated Variable, MV) was adjusted proportional to the deviation of the rotary speed of the machine (Controlled Variable, CV) from its desired value and as such stabilized the machine by means of negative feedback. More generally, it can be considered a technological breakthrough that opened the path towards the widespread use of automatic control methods in industry.

This very concept of proportional negative feedback still lies at the basis of most control systems in use today, to which for example the widespread use of PID (Proportional, Integrational and Differential) controllers can testify. However, these controllers only form a small part of a larger hierarchy that makes up modern types of process control. In the next section this hierarchy will be further explained and more specifically the place that Model based Predictive Control – the topic of this thesis – has earned within this hierarchy.

1.2 The process control hierarchy

In this section we discuss the main rationale for the existence of different levels in the hierarchy of a process control system and the specific requirements that form the *raisons d'être* of Model based Predictive Control in this hierarchy.

1.2.1 Multiple-input / multiple-output control

Compared to the era of Watt's steam engine, current industrial reality has become much more complicated. Nowadays, typically many different mechanical, chemical, electric and electronic systems are interconnected and interact in complex and dynamic ways. As a result changing the position of one valve or switch typically influences a multitude of other quantities instead of just a single quantity – speed – in the case of Watt's steam engine. Or, in more mathematical terms, changing the value of a single MV typically

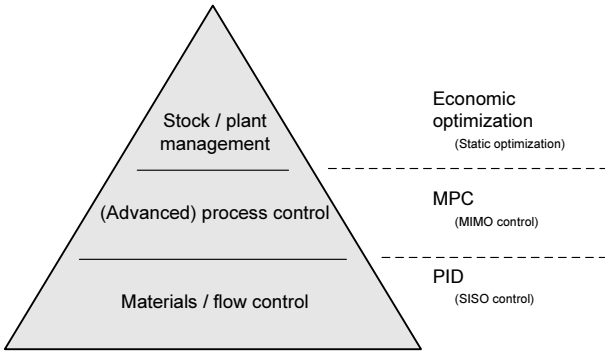


Figure 1.2: Depiction of the process control hierarchy. The different process control levels have different control scopes, use different models and have to satisfy different requirements. See also Table 1.1.

affects multiple CVs and conversely a single CV typically depends upon multiple MVs. Therefore the pairing of CVs and MVs and controlling all these pairs with independent control loops is often ineffective since in this way the interaction between the different loops is not taken into account.

While nowadays at the lowest level all valves, pumps, burners and motors are still controlled by such SISO (Single-Input / Single-Output) controllers , at a higher level (Figure 1.2) a supervisory controller is employed for adjusting the different set-points of the lower level controllers, taking into account the dynamic interactions between the different MVs and CVs.

The control paradigm used at this level hence has to be capable of efficiently tackling MIMO (Multiple-Input / Multiple-Output) control problems.

1.2.2 Constraint handling

An additional complication is the presence of constraints on certain of the plant variables. These constraints can be either hard or soft constraints and can represent inherent limitations of the controlled system (e.g., valve positions are restricted to the range 0%-100%), safety limitations (e.g., maximum tank pressure, maximum reactor temperature, ...), environmental regulations (e.g., NO_x emission restrictions), quality constraints (e.g., produced goods have to satisfy certain customer specifications) or economic constraints (e.g., maximum amount of energy to be used).

Traditionally one would choose an operating point sufficiently far from these imposed constraints in order to ensure constraint satisfaction at all times despite disturbances in the controlled system. However, the general trend of globalization of the economy, increasingly tough competition and stricter customer demands have forced companies to operate plants at their economical limits and hence closer to the economical, environmental, safety, ... constraints.

Therefore, it is important that the algorithms employed at the process level of the control hierarchy allow for efficient, non-conservative constraint handling, since this

Level	Model type	Constr.	Update freq.	Algorithm
3	static	yes	~ 1 day^{-1}	economic opt.
2	dynamic (MIMO)	yes	~ 1 min.$^{-1}$	MPC
1	dynamic (SISO)	no	~ 1 sec.$^{-1}$	PID

Table 1.1: Different requirements for the different levels of the process control hierarchy depicted in Figure 1.2.

can directly translate in financial, environmental, safety and quality benefits.

1.2.3 Pro-active plant operation

A third requirement for control at the process level is the ability to control pro-actively. In many cases external disturbances to the system can be measured before their actual effect can be noticed, e.g. a change in composition of a raw material that enters a series of reactors, whose effect on the composition of the end product can only be measured when the first product leaves the reactor. In such cases pro-actively counteracting these disturbances can give significant performance benefits compared to pure feedback control that would only start acting when the disturbance effect becomes noticeable.

Secondly, when a plant has to make frequent transitions between different operating points (e.g., in order to produce different product grades) pro-active behavior can also lead to significant performance benefits, since these transitions are typically known hours in advance.

For these reasons, the ability to act pro-actively is a third important requirement for control algorithms at the process level.

1.2.4 Computational complexity

Since typically the components that have the fastest dynamical behavior correspond to those that are controlled directly by PID controllers at the lowest level of the control hierarchy, i.e. valves, pumps, ... it is not necessary that the controller at the process control level operates at a high sample frequency. As a result the algorithms employed at this level can have a higher computational complexity compared to those at the lowest levels, which makes it computationally possible to incorporate the previous requirements.

1.2.5 Summary

In order to cope with control problems of increasing complexity, industrial process control has been built up hierarchically (see Figure 1.2, Table 1.1) and consists of three levels. At the second level the most important requirements are the ability to efficiently control MIMO systems, to handle imposed constraints non-conservatively and to be able to act pro-actively. On the other hand, due to the lower sample frequency an increased computational complexity is allowed.

Model based Predictive Control fits perfectly in these requirements, as will be clarified in the next section.

1.3 Model based predictive control

In this section a general description of Model based Predictive Control (MPC) is given for the case of Linear Time-Invariant (LTI) dynamical models. Later chapters will focus on more general settings, where the main contributions of this thesis are situated. Many good books on MPC exist, such as e.g., [10, 26, 69, 79, 117].

1.3.1 Model, constraints, control objective

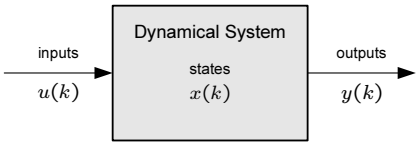


Figure 1.3: Schematic depiction of a dynamical system.

Before being able to state the basic MPC formulation, it is important to first discuss the models used to describe the dynamical behavior of the plant to be controlled and the control objective. In this chapter we consider LTI models in state space form:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k), \\ y(k) &= Cx(k), \end{aligned} \qquad k \in \mathbb{N}. \tag{1.1}$$

$u(k) \in \mathbb{R}^{n_u}$ denotes the vector of inputs at discrete time k and can be interpreted as the vector of values describing the manipulated variables of the system at time k. $y(k) \in \mathbb{R}^{n_y}$ denotes the vector of output at discrete time k and can be interpreted as the vector of controlled variables of the system at time k. $x(k) \in \mathbb{R}^{n_x}$ denotes the state vector at discrete time k and acts as a memory containing all information about the past of the system that is necessary to predict the future behavior. $A \in \mathbb{R}^{n_x \times n_x}$, $B \in \mathbb{R}^{n_x \times n_u}$, $C \in \mathbb{R}^{n_y \times n_x}$ are matrices defining the actual behavior of the system.

In this thesis we only consider state feedback MPC, which means that exact knowledge of the system state is assumed. From a practical point of view this implies that any errors introduced by a state estimator (used to estimate the states based on output measurements) are assumed to be sufficiently small and are hence neglected. Consequently the control objective is formulated in terms of the system states instead of the system outputs. The second equation of (1.1) is therefore omitted in further discussions. For a detailed discussion of this state-feedback assumption, the reader is referred to [48, 59, 60].

The aim of MPC is to implicitly construct a strategy $u(k) = \kappa_{\mathrm{MPC}}(x(k))$ for determining the inputs $u(k)$, given information about the current state of the system, in order to steer the states of the system towards their reference values despite the

influence of any external disturbances, while guaranteeing satisfaction of constraints imposed on the inputs, states and/or outputs:

$$u(k) \in \mathcal{U}, \qquad k \in \mathbb{N}, \tag{1.2a}$$
$$x(k) \in \mathcal{X}, \qquad k \in \mathbb{N}, \tag{1.2b}$$
$$y(k) \in \mathcal{Y}, \qquad k \in \mathbb{N}. \tag{1.2c}$$

Please note that, using (1.1), constraints on the outputs can always be rewritten as state constraints $Cx(k) \in \mathcal{Y}, \forall k$. In further sections we will assume that $\mathcal{U}, \mathcal{X}, \mathcal{Y}$ are convex, compact sets containing the origin in their interior. The control objective is to minimize a quadratic cost objective:

$$\sum_{k=0}^{\infty} \|x(k) - x_{\mathrm{ref}}(k)\|_Q^2 + \|u(k) - u_{\mathrm{ref}}(k)\|_R^2, \tag{1.3}$$

where $Q \in \mathbb{S}_{++}^{n_x}, R \in \mathbb{S}_{++}^{n_u}$ are state and input weighting matrices respectively. The choice of a quadratic cost objective has computational advantages, as will be discussed later, but also has the advantage of resulting in a more smooth control behavior compared to e.g. an L_1 or L_∞ objective.

1.3.2 Basic algorithm formulation

For reasons of clarity only MPC with the aim of system stabilization (i.e. $x_{\mathrm{ref}}(k) \equiv 0, u_{\mathrm{ref}}(k) \equiv 0, \forall k$) will be considered. Tracking problems ($x_{\mathrm{ref}}(k) \neq 0, u_{\mathrm{ref}}(k) \neq 0$) will be considered in Chapter 7.

Algorithm 1.1 (**Model based Predictive Control).** *Given a model* (1.1)*, subject to constraints* (1.2a)-(1.2b) *and given a control objective* (1.3)*, solve at each time instant k, given the value of the current state $x(k) \equiv x(k|k)$, the following optimization problem:*

$$\min_{\mathbf{x}_N(k), \mathbf{u}_N(k)} \quad \sum_{i=0}^{N} \|x(k+i|k)\|_Q^2 + \sum_{i=0}^{N-1} \|u(k+i|k)\|_R^2, \tag{1.4a}$$

$$\text{s.t.} \quad x(k+i|k) \in \mathcal{X}, \qquad i = 1, \ldots, N, \tag{1.4b}$$
$$u(k+i|k) \in \mathcal{U}, \qquad i = 0, \ldots, N-1, \tag{1.4c}$$
$$x(k+i+1|k) = Ax(k+i|k) + Bu(k+i|k), \quad i = 0, \ldots, N-1, \tag{1.4d}$$

with

$$\mathbf{u}_N(k) = [u(k|k); \ldots; u(k+N-1|k)], \tag{1.5}$$
$$\mathbf{x}_N(k) = [x(k+1|k); \ldots; x(k+N|k)], \tag{1.6}$$

and apply the input $u(k) \equiv u(k|k)$ to the plant. Repeat this procedure at the next time step $k+1$ based on updated state information.

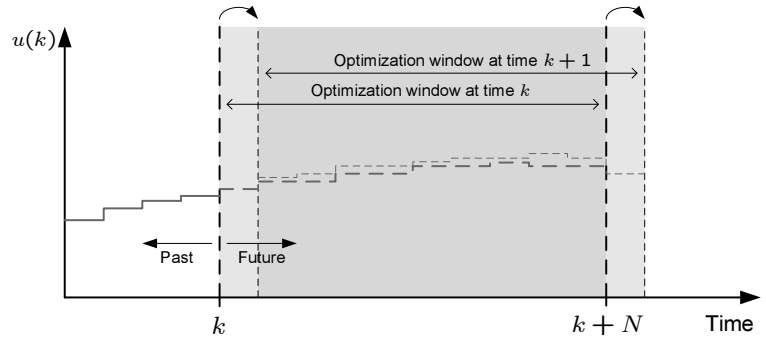


Figure 1.4: The principle of Receding Horizon Control (RHC). At every time instant k an optimal control problem of length N is solved after which only the first input vector is applied to the plant.

Model based Predictive Control hence consists of solving a finite-horizon optimal control problem at each time instant k after which only the first input vector of the optimal input sequence is applied to the system. This methodology was first proposed in the 60's [73, 109] and is depicted in Figure 1.4.

It is clear that this methodology is naturally capable of dealing with MIMO systems and explicitly takes imposed constraints into account from the outset, which classical control algorithms [22, 47, 49] typically cannot. Furthermore, due to the fact that an optimal control problem is solved over a future time window, it is straightforward to incorporate knowledge about future events into the optimization problem, leading to pro-active behavior. For these reasons MPC is considered the most suitable algorithm [111, 112] for MIMO control of many practical applications.

1.4 Mathematical tools

Before moving on to some more theoretical aspects of MPC that are of importance in later chapters, it is useful to give an introduction on two important tools that are used in this thesis: *convex optimization* and *computational geometry*.

1.4.1 Convex optimization

Since MPC is an optimization-based control paradigm it is clear that thought has to be given to how these optimization problems are formulated. Furthermore, these optimization problems are to be solved on-line which results in certain efficiency and guaranteed solvability requirements. For these reasons, one always aims to pose the on-line MPC optimization problem as a convex optimization problem [24].

1.4.1.1 Definitions

Before defining convex optimization, we define some auxiliary concepts.

Definition 1.1 (Convex set). *A set $\mathcal{S} \subseteq \mathbb{R}^n$ is convex iff for any two points $x_1, x_2 \in \mathcal{S}$ all convex combinations of these points also lie within the set $\mathcal{S}$:*

$$(1-\theta)x_1 + \theta x_2 \in \mathcal{S}, \qquad \forall \theta \in [0,1], \forall x_1, x_2 \in \mathcal{S}.$$

Definition 1.2 (Affine function). *A function $f : \mathbb{R}^n \to \mathbb{R}$ is affine iff it can be written in the following form:*

$$f(x) = a^{\mathrm{T}}x + b, \qquad a \in \mathbb{R}^n, b \in \mathbb{R}.$$

A *linear* function can be defined as the special case of an affine function where $b = 0$. In further sections the term *linear function* will be used more loosely for referring to any affine function.

Definition 1.3 (Convex function). *A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex iff its epigraph $\mathrm{epi}(f) \triangleq \{[x;c] | x \in \mathrm{dom}(f), f(x) \leq c\}$, with $\mathrm{dom}(f) \triangleq \{x \in \mathbb{R}^n | f(x)$ is well-defined$\}$ the domain of f, is a convex set. Equivalently, f is convex iff $\mathrm{dom}(f)$ is convex and*

$$f((1-\theta)x_1 + \theta x_2) \leq (1-\theta)f(x_1) + \theta f(x_2), \quad \forall \theta \in [0,1], \forall x_1, x_2 \in \mathrm{dom}(f).$$

In this thesis we consider optimization problems that can be written in the following standard form:

$$\begin{aligned} \min_{x} \quad & f_0(x), && && \text{(1.7a)} \\ \text{s.t.} \quad & f_i(x) \leq 0, && i = 1, \ldots, m_{\text{ineq}}, && \text{(1.7b)} \\ & h_i(x) = 0, && i = 1, \ldots, m_{\text{eq}}. && \text{(1.7c)} \end{aligned}$$

$x \in \mathbb{R}^n$ is called the vector of optimization variables, $f_0 : \mathbb{R}^n \to \mathbb{R}$ is called the objective function or cost function, $f_i : \mathbb{R}^n \to \mathbb{R}$ are called the inequality constraints and $h_i : \mathbb{R}^n \to \mathbb{R}$ the equality constraints.

Definition 1.4 (Feasible solution). *A vector $x^{\mathrm{f}} \in \mathbb{R}^n$ is a feasible solution to* (1.7) *iff it satisfies constraints* (1.7b)-(1.7c)*.*

Optimization problem (1.7) is called *feasible* if there exists at least one feasible solution, otherwise it is called *infeasible*.

Definition 1.5 ((Globally) optimal solution). *The globally optimal solution $x^{\mathrm{o}} \in \mathbb{R}^n$ to* (1.7) *is defined as*

$$\inf\{f_0(x) | x \text{ is a feasible solution to } (1.7)\}. \tag{1.8}$$

A feasible solution x^{f} is called *locally optimal* if there exists a value $r > 0$ such that x^{f} is the optimal solution to the optimization problem (1.7) augmented with the constraint $\|x - x^{\mathrm{f}}\| \leq r$.

We can now define when optimization problem (1.7) is called *convex* and what the implications are of that property.

Definition 1.6 (Convex optimization). *An optimization problem* (1.7) *is convex if its objective function* $f_0(x)$ *is a convex function, the inequality constraint functions* $f_{\mathrm{i}}(x), i = 1, \ldots, m_{\mathrm{ineq}}$ *are convex functions and the equality constraint functions* $h_{\mathrm{i}}(x), i = 1, \ldots, m_{\mathrm{eq}}$ *are affine functions.*

The latter two conditions imply that the set of feasible points $\mathcal{F}$, which is defined as $\mathcal{F} \triangleq \{x | x \text{ is a feasible solution to (1.7)}\}$, is a convex set.

There is an extensive amount of literature available on convex optimization, among which [24] is a good starting point, covering many different aspects, but the main properties of convex optimization problems that are relevant to this thesis can be summarized as follows:

- **Global / local optimality:** It can easily be shown [24, p. 138] that any local optimum of a convex optimization is also a global optimum. Since in this thesis optimization is mostly used as an on-line tool, where user interaction is not possible, this property is of foremost importance.
- **Computational efficiency:** Several algorithms (e.g. interior point algorithms [89]) exist that can efficiently solve convex optimization problems with guaranteed precision. For most classes of convex optimization problems worst-case bounds on the computational complexity can be obtained that are polynomial functions of the problem size $(n, m_{\mathrm{ineq}}, m_{\mathrm{eq}})$.
- **Tools for special subclasses:** Many software packages (both free and commercial) exist that are capable of solving certain classes of convex optimization problems with specific problem structures (LP, QP, SOCP, SDP). Therefore, being able to formulate convex optimization problems in the form of one of these specific subclasses can lead to substantial reductions in required computational complexity.

A few well known subclasses of convex optimization are discussed next in order of increasing generality and decreasing computational efficiency.

1.4.1.2 Linear programming

A Linear Program (LP) can be written in the following standard form

$$\begin{aligned} \min_{x} \quad & f^{\mathrm{T}} x, \\ \text{s.t.} \quad & A_{\mathrm{ineq}} x \leq b_{\mathrm{ineq}}, \\ & A_{\mathrm{eq}} x = b_{\mathrm{eq}}, \end{aligned}$$

with $f \in \mathbb{R}^n, A_{\mathrm{ineq}} \in \mathbb{R}^{m \times n}, b_{\mathrm{ineq}} \in \mathbb{R}^m, A_{\mathrm{eq}} \in \mathbb{R}^{m \times n}, b_{\mathrm{eq}} \in \mathbb{R}^m$. It is clear from the definition that LPs are convex optimization problems. We refer to [42, 78, 138] for an overview of linear programming. Linear programming is used in this thesis, among other things, to check redundancy of linear constraints with respect to other linear constraints. See Chapters 2, 5 and 6 for details. Linear equality constraints can always be eliminated by a change of variables and will therefore not be explicitly mentioned in the following subclasses. In this thesis, where not explicitly mentioned, the MATLAB toolbox Sedumi [133] is used to solve LPs.

1.4.1.3 Quadratic programming

A Quadratic Program (QP) can be written in the following standard form

$$\begin{aligned} \min_{x} \quad & \frac{1}{2} x^{\mathrm{T}} H x + f^{\mathrm{T}} x, \\ \text{s.t.} \quad & A_{\mathrm{ineq}} x \leq b_{\mathrm{ineq}}, \end{aligned}$$

with $H \in \mathbb{S}^n_{++}, f \in \mathbb{R}^n, A_{\mathrm{ineq}} \in \mathbb{R}^{m \times n}, b_{\mathrm{ineq}} \in \mathbb{R}^m$. See [24] for more information on Quadratic programming and its history. Typically on-line MPC optimization problems are cast as QPs since the quadratic control objective (1.3) cannot be represented in the more restrictive form of LPs, while more general classes of convex optimization are less favorable from a computational point of view. Note that optimization problem (1.4) can be written in the above standard form of a QP.

1.4.1.4 Second-order cone programming

A Second-Order Cone Program (SOCP) has the following standard form

$$\begin{aligned} \min_{x} \quad & f^{\mathrm{T}} x, \\ \text{s.t.} \quad & \|A_i x + b_i\|_2 \leq c_i^{\mathrm{T}} x + d_i, \qquad i = 1, \ldots, m, \end{aligned}$$

with $A_i \in \mathbb{R}^{n_i \times n}, b_i \in \mathbb{R}^{n_i}, c_i \in \mathbb{R}^n, d_i \in \mathbb{R}, n_i \in \mathbb{N}_0, i = 1, \ldots, m$. The name of this class of optimization problems refers to the fact that the inequality constraints are equivalent to demanding that the affine functions $[A_i x + b_i; c_i^{\mathrm{T}} x + d_i]$ lie in *second-order cones* in $\mathbb{R}^{n+1}$, which are defined as $\{[x; c] | x \in \mathbb{R}^n, c \in \mathbb{R}, \|x\|_2 \leq c\}$. An overview of the use of SOCPs can be found in e.g. [1, 77]. SOCPs typically arise in robust MPC where a maximum of 2-norms has to be minimized. See Chapter 3 for more details. For a computationally efficient MATLAB toolbox for solving SOCPs we refer to [85].

1.4.1.5 Semi-definite programming

A Semi-Definite Program (SDP) can be written in the following form

$$\begin{aligned} \min_{x} \quad & f^{\mathrm{T}} x, \\ \text{s.t.} \quad & F_{i,0} + F_{i,1} x_1 + \ldots + F_{i,n} x_n \preceq 0, \qquad i = 1, \ldots, m, \end{aligned}$$

with $f \in \mathbb{R}^n, F_{i,j} \in \mathbb{S}^{n_i}, n_i \in \mathbb{N}_0, i = 1, \ldots, m, j = 1, \ldots, n$. Note that the inequality $\preceq$ denotes a matrix inequality instead of a scalar inequality and hence means that the left hand-side of the above inequalities should be negative semi-definite. The above inequalities are often referred to as *Linear Matrix Inequalities (LMIs)*. Linear matrix inequalities arise in many applications of systems and control theory [23] and appear in this thesis often as off-line optimization problems to be used in the design phase of robust MPC controllers. See Chapters 3 and 4 and Appendix A for more information. In this thesis, where not explicitly mentioned, the MATLAB toolbox Sedumi [133] is used to solve SDPs.

1.4.2 Computational geometry

1.4.2.1 Constraint set manipulation

In this thesis the emphasis is put on constraint handling in MPC and as a result several techniques and algorithms are described that deal with *constraint sets*. These can either be the imposed constraint sets $\mathcal{X}$,$\mathcal{U}$ or sets derived thereof. Furthermore, since constraints in MPC typically are linear, these constraint sets can be described as *intersections of halfspaces* or as *convex hulls of points*. Finally, since we are interested in scalability issues of constraint handling in MPC, these sets will potentially be *high-dimensional*, i.e. up to 100-dimensional or more.

Due the high dimensionality of these objects it is obvious that operations on these objects (e.g., projections) can be rather complex and therefore have to be dealt with using numerical algorithms. The field of computational geometry deals with these kind of algorithms and therefore provides useful tools to deal with these constraint sets. This discipline, that lies in the intersection between mathematics and computer science, has applications in many different fields, among which computer graphics, fluid dynamics simulations, finite element modeling, robot motion planning, ... and can, for the reasons mentioned above, also be used in constrained control problems.

Therefore, in this section, we discuss several concepts and tools from computational geometry that are used in later parts of this thesis. More detailed information on computational geometry can be found in [53, 90, 107]. An overview on polytopes and their properties can be found in [146].

1.4.2.2 Set representations

First, we define the geometric objects we will deal with and their possible representations.

Definition 1.7 (Halfspace). *An $n-dimensional$ halfspace $\mathcal{H}$ is defined as*

$$\mathcal{H} = \{x \in \mathbb{R}^n | a^{\mathrm{T}} x \leq b\},$$

with $a \in \mathbb{R}^n, b \in \mathbb{R}$.

Definition 1.8 ((Convex) polyhedron). *A set $\mathcal{S} \subset \mathbb{R}^n$ is a (convex) polyhedron if it can be written as an intersection of halfspaces*

$$\mathcal{S} = \bigcap_{i=1}^{m} \mathcal{H}_i, \qquad \mathcal{H}_i = \{x \in \mathbb{R}^n | a_i^{\mathrm{T}} x \leq b_i\}, \quad i = 1, \ldots, m,$$

with $m \in \mathbb{Z}_0^+$.

Alternatively, one can write $\mathcal{S} = \{x \in \mathbb{R}^n | Ax \leq b\}$, with $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$.

Definition 1.9 ((Convex) polytope). *A set $\mathcal{S} \subset \mathbb{R}$ is a (convex) polytope if it is a polyhedron and bounded, with bounded defined as*

$$\nexists y, x_0 \in \mathbb{R}^n : \qquad x_0 + cy \in \mathcal{S}, \ \forall c \in \mathbb{R}^+.$$

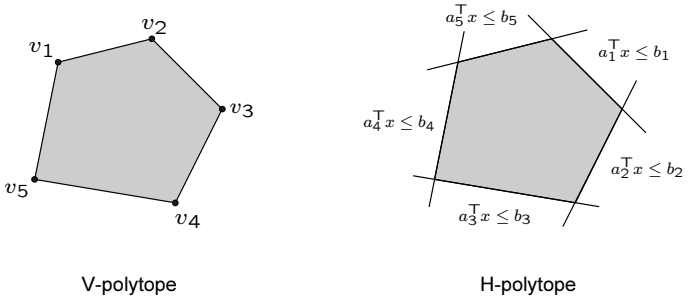


Figure 1.5: Left: Representation of a polytope by means of vertices (V-polytope). **Right:** Representation of a polytope by means of halfspaces (H-polytope).

In later sections of this thesis we will omit the term 'convex'. Furthermore, we will assume that all polytopes contain the origin in their interior, which is no restrictive assumption in MPC and allows us to write polytopes in the standard form

$$\mathcal{S} = \{x \in \mathbb{R}^n | A_{\mathcal{S}} x \leq \mathbf{1}\}, \tag{1.9}$$

with $A_{\mathcal{S}} \in \mathbb{R}^{m \times n}$. $\mathbf{1}$ denotes a vector of appropriate size containing only ones.

A polytope described as an intersection of halfplanes or equivalently as the set of solutions to a set of linear inequalities as in (1.9) is called an *H-polytope*. Alternatively, one can represent a polytope as the convex hull of a set of points in $\mathbb{R}^n$, in which case the polytope is referred to as a *V-polytope*. See Figure 1.5 for an illustration of the different representations.

Definition 1.10 (Convex hull). *The convex hull of a set of m vectors is defined as*

$$\text{Co}\{v_1, \ldots, v_m\} \triangleq \{\lambda_1 v_1 + \ldots + \lambda_m v_m | \lambda_i \geq 0, i = 1, \ldots, m, \sum_{i=1}^{m} \lambda_i = 1\}.$$

The convex hull of a set of matrices can be defined similarly and will be used in later chapters to define polytopic model uncertainty sets.

Since in constrained control problems often only component-wise constraints are imposed, the corresponding constraint sets $\mathcal{X}$,$\mathcal{U}$ are hypercubes. As a result, representation as H-polytopes is preferrable compared to V-polytopes, since the number of vertices of a hypercube equals 2^n, with n the dimensionality. An additional advantage is that H-polytopes can directly be incorporated as constraints in optimization problems.

1.4.2.3 Geometric operations

The first and most straightforward operations that can be defined are the intersection of sets and Minkowski sum and Minkowski (or Pontryagin) difference, three operations that preserve convexity and are used the most in the following chapters.

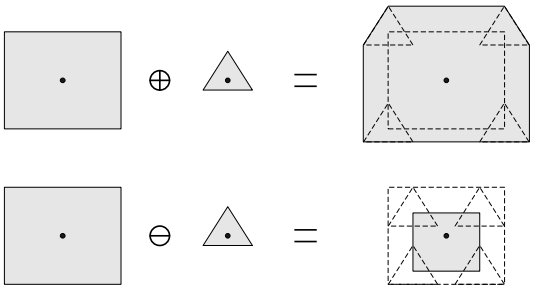


Figure 1.6: The Minkowski sum and difference of two sets. The origins of the respective sets are indicated with dots.

Definition 1.11 (Intersection: $\cap$). *The intersection of two polytopes $\mathcal{A}, \mathcal{B} \subset \mathbb{R}^n$ is defined as*

$$\mathcal{A} \cap \mathcal{B} \triangleq \{x \in \mathbb{R}^n | x \in \mathcal{A}, x \in \mathcal{B}\}.$$

If $\mathcal{A}$ and $\mathcal{B}$ are represented as H-polytopes in standard form (1.9), computing the intersection is trivial : $\mathcal{A} \cap \mathcal{B} = \{x | [A_{\mathcal{A}}; A_{\mathcal{B}}]x \leq \mathbf{1}\}$. Computing intersections involving V-polytopes is less trivial and not used in this thesis.

Definition 1.12 (Minkowski sum: $\oplus$, [146]). *The Minkowksi sum of two sets $\mathcal{A}, \mathcal{B} \subset \mathbb{R}^n$ is defined as*

$$\mathcal{A} \oplus \mathcal{B} \triangleq \{a + b | a \in \mathcal{A}, b \in \mathcal{B}\}.$$

If $\mathcal{A}$ and $\mathcal{B}$ are represented as V-polytopes $\mathcal{A} = \mathrm{Co}\{v_1, \ldots, v_{m_{\mathcal{A}}}\}$ and $\mathcal{B} = \mathrm{Co}\{w_1, \ldots, w_{m_{\mathcal{B}}}\}$, then the Minkowski sum can be computed as $\mathcal{A} \oplus \mathcal{B} = \mathrm{Co}\{v_i + w_j | i = 1, \ldots, m_{\mathcal{A}}, j = 1, \ldots, m_{\mathcal{B}}\}$.

If $\mathcal{A}$ and $\mathcal{B}$ are represented as H-polytopes in standard form (1.9), then the Minkowski sum can be computed as $\mathcal{A} \oplus \mathcal{B} = \{x \in \mathbb{R}^n | [A_{\mathcal{A}}; A_{\mathcal{B}}]x \leq \mathbf{1} + [c; d]\}$, with $c \in \mathbb{R}^{m_{\mathcal{A}}}, d \in \mathbb{R}^{m_{\mathcal{B}}}$ computed as

$$\begin{aligned} c_i &= \max_x \; A_{\mathcal{A}}[i,:]x \quad \text{s.t.} \quad A_{\mathcal{B}}x \leq \mathbf{1}, && i = 1, \ldots, m_{\mathcal{A}}, \\ d_i &= \max_x \; A_{\mathcal{B}}[i,:]x \quad \text{s.t.} \quad A_{\mathcal{A}}x \leq \mathbf{1}, && i = 1, \ldots, m_{\mathcal{B}}. \end{aligned}$$

Definition 1.13 (Minkowski (or Pontryagin) difference: $\ominus$, [146]). *The Minkowksi difference of two sets $\mathcal{A}, \mathcal{B} \subset \mathbb{R}^n$ is defined as*

$$\mathcal{A} \ominus \mathcal{B} \triangleq \{x | \{x\} \oplus \mathcal{B} \subseteq \mathcal{A}\} \equiv \{a | \forall b \in \mathcal{B} : a + b \in \mathcal{A}\}.$$

If $\mathcal{A}$ is represented as an H-polytope, the Minkowski difference can be computed similarly as the Minkowski sum. In the other case, the computation is less trivial. Only the case when $\mathcal{A}$ is an H-polytope is needed in this thesis.

Note that, as opposed to the scalar sum and difference, the relationship $(\mathcal{A} \ominus \mathcal{B}) \oplus \mathcal{B} = \mathcal{A}$ is not true in general, whereas the following relationship $(\mathcal{A} \oplus \mathcal{B}) \ominus \mathcal{B} = \mathcal{A}$ *is* true in

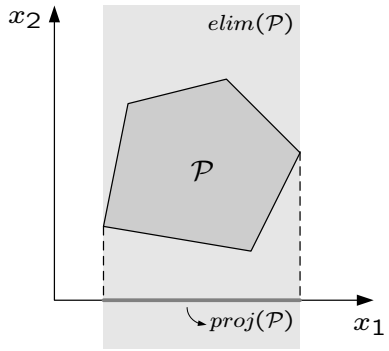


Figure 1.7: Illustration of the projection and elimination of a two dimensional polytope along the second dimension.

general, so care has to be taken when thinking about these operations intuitively. See Figure 1.6 for an illustration of the Minkowski sum and difference.

Two other geometric tools used in this thesis are related to projecting sets to lower-dimensional spaces. They are further elaborated on in later chapters and are hence only mentioned briefly here. For the sake of simplicity of notation we only consider one-dimensional projection and elimination of n-dimensional polytopes along the n-th dimension.

Definition 1.14 (Projection). *The one-dimensional projection of a polytope $\mathcal{P} \in \mathbb{R}^n$ along the n-th dimension is defined as*

$$\mathrm{proj}(\mathcal{P}) \triangleq \{x \in \mathbb{R}^{n-1} | \exists c \in \mathbb{R} : [x; c] \in \mathcal{P}\}.$$

Definition 1.15 (Elimination). *The one-dimensional elimination of a polytope $\mathcal{P} \in \mathbb{R}^n$ along the n-th dimension is defined as*

$$\mathrm{elim}(\mathcal{P}) \triangleq \{x \in \mathbb{R}^n | \exists c \in \mathbb{R} : x + ce_n \in \mathcal{P}\},$$

where e_n is the n-th unit vector of $\mathbb{R}^n$.

Both operations are clearly related and one can see that $\mathrm{proj}(\mathrm{elim}(\mathcal{P})) \equiv \mathrm{proj}(\mathcal{P})$ and $\mathrm{elim}(\mathcal{P}) \equiv \mathrm{proj}(\mathcal{P}) \times \mathbb{R}$. Figure 1.7 further illustrates both operations.

In the case that $\mathcal{P}$ is represented as a V-polytope these operations are trivial, but in the case that $\mathcal{P}$ is represented as a H-polytope – which will mostly be the case in this thesis – these operations are computationally non-trivial and need to be performed using Fourier-Motzkin elimination. This algorithm will be discussed and modified in Chapter 6.

1.5 Stability framework

1.5.1 Stability of MPC vs. linear control laws

Classical linear system theory extensively deals with stability issues and performance of linear systems. Therefore linear system theory and more specifically, analysis of poles and zeros of closed-loop transfer functions is the tool of choice to analyze properties of linear controllers.

However, due to the fact that imposed constraints are explicitly taken into account in MPC controllers, the resulting control laws will not be linear and hence these tools cannot be used. Therefore a new stability framework (see e.g. [82, 131]) emerged in the previous decade, based on Lyapunov theory.

In the case of MPC algorithms for stabilization of disturbance-free systems, one typically aims at proving *asymptotic stability* for a non-trivial (or loosely speaking a 'larger-than-ϵ') region of initial states around the origin. This is done in two steps. The first step aims at characterizing the set of initial states for which the resulting closed loop trajectories are *well-defined*. For this purpose one uses the concept of *recursive feasibility*:

Definition 1.16 (Recursive feasibility). *An MPC optimization problem is said to be recursively feasible, if feasibility at time k implies feasibility at time $k+1$.*

If an MPC algorithm is shown to be recursively feasible, then it is guaranteed by induction that if it is feasible at $k = 0$, it will be feasible $\forall k \in \mathbb{N}$ and hence the resulting trajectory is well-defined. In a second step one aims at proving asymptotic stability for all these initial states that lead to a feasible MPC optimization problem, which we will refer to as *semi-global asymptotic stability*:

Definition 1.17 (Semi-global asymptotic stability). *The closed-loop system formed by a dynamical system* (1.1) *and a controller is asymptotically stable for a given set of initial states $\mathcal{X}_0$ iff*

$$\lim_{k\to\infty} \|x(k)\| = 0, \qquad \forall x(0) \in \mathcal{X}_0. \tag{1.10}$$

Semi-global asymptotical stability can be proven by means of the following lemma.

Lemma 1.1 (Lyapunov stability). *An autonomous system $x(k+1) = g(x(k)), k \in \mathbb{N}_0$ is asymptotically stable for all initial values $x(0) \in \mathcal{X}_0$ if there exists a convex function $V : \mathcal{X}_0 \to \mathbb{R}$ (Lyapunov function) that satisfies the following conditions:*

- $V(g(x)) < V(x), \forall x \in \mathcal{X}_0 \backslash 0$,
- $V(g(0)) = V(0)$.

The following section describes how semi-global asymptotic stability can be obtained.

1.5.2 Quasi-infinite horizon MPC

In order to be able to guarantee asymptotic stability of the closed-loop system, Algorithm 1.1 has to be slightly modified. The following algorithm, that in essence

was already presented in [135] but is more widely known in the form presented in [131], leads to asymptotic stability under conditions that are stated below. Since the modifications essentially are aimed at taking into account the constraints and control objective beyond the horizon, it is often referred to as *Quasi-Infinite Horizon MPC.*

Algorithm 1.2 (Quasi-Infinite Horizon MPC). *Given a model* (1.1)*, subject to constraints* (1.2a)-(1.2b) *and given a control objective* (1.3)*, solve at each time instant* k*, given the value of the current state* $x(k) \equiv x(k|k)$*, the following optimization problem:*

$$\min_{\mathbf{x}(k),\mathbf{u}(k)} \quad \sum_{i=0}^{N-1} \|x(k+i|k)\|_Q^2 + \sum_{i=0}^{N-1} \|u(k+i|k)\|_R^2 + \|x(k+N|k)\|_{Q_N}^2, \tag{1.11a}$$

$$\text{s.t.} \quad x(k+i|k) \in \mathcal{X}, \qquad i = 1, \ldots, N-1, \tag{1.11b}$$

$$x(k+N|k) \in \mathcal{X}_N, \tag{1.11c}$$

$$u(k+i|k) \in \mathcal{U}, \qquad i = 0, \ldots, N-1, \tag{1.11d}$$

$$x(k+i+1|k) = Ax(k+i|k) + Bu(k+i|k), \quad i = 0, \ldots, N-1, \tag{1.11e}$$

and apply the input $u(k) \equiv u(k|k)$ *to the plant. Repeat this procedure at the next time step* $k+1$ *based on updated state information.*

$Q_N \in \mathbb{S}_{++}^{n_x}$ and $\mathcal{X}_N \subseteq \mathbb{R}^{n_x}$ are called the terminal cost matrix and the terminal constraint set respectively. The latter is included in order to obtain recursive feasibility, while the former is included in the optimization problem in order to obtain asymptotic stability of the closed-loop system. Many conceptually similar algorithms exist [17, 32, 74, 75, 80]. Most algorithms differ in the model class for which the controller is designed or the way the terminal cost and terminal constraint set are actually calculated.

The following lemma, discussed in much detail in [82], states under which conditions Algorithm 1.2 is recursively feasible and leads to a semi-globally asymptotically stable closed-loop system. We also give the proof since it provides insights in the role of the terminal cost, terminal constraint set and the terminal controller, that are also of importance in the following chapters.

Lemma 1.2 (Semi-global asymptotic stability). *The closed-loop system formed by system* (1.1) *and the MPC controller defined in Algorithm 1.2 leads to recursive feasibility of the controller and asymptotic stability of the closed-loop system for all states* $x(0)$ *that lead to a feasible optimization problem* (1.11)*, if a controller* $u(k) = \kappa_N(x(k))$ *(terminal controller) exists such that the following conditions are satisfied:*

1. $\mathcal{X}_N \subseteq \mathcal{X}$, (1.12a)
2. $\kappa_N(x) \in \mathcal{U}, \qquad \forall x \in \mathcal{X}_N,$ (1.12b)
3. $Ax + B\kappa_N(x) \in \mathcal{X}_N, \qquad \forall x \in \mathcal{X}_N,$ (1.12c)
4. $\|x\|_{Q_N}^2 - \|(Ax + B\kappa_N(x))\|_{Q_N}^2 \geq \|x\|_Q^2 + \|\kappa_N(x)\|_R^2, \quad \forall x \in \mathcal{X}_N.$ (1.12d)

Proof: Consider a feasible solution $\mathbf{x}^{\mathrm{o}}(k), \mathbf{u}^{\mathrm{o}}(k)$ to optimization problem (1.11) at time k:

$$\begin{aligned}\mathbf{x}^{\mathrm{o}}(k) &= [x^{\mathrm{o}}(k+1|k); \ldots; x^{\mathrm{o}}(k+N|k)],\\ \mathbf{u}^{\mathrm{o}}(k) &= [u^{\mathrm{o}}(k|k); \ldots; u^{\mathrm{o}}(k+N-1|k)].\end{aligned}$$

It is now possible to construct a candidate feasible solution $\mathbf{x}^{\mathrm{f}}(k+1), \mathbf{u}^{\mathrm{f}}(k+1)$ for the optimization problem at time $k+1$ as follows:

$$\begin{aligned}\mathbf{x}^{\mathrm{f}}(k+1) &= [x^{\mathrm{o}}(k+2|k); \ldots; x^{\mathrm{o}}(k+N|k); x^{\mathrm{f}}(k+N+1|k+1)],\\ \mathbf{u}^{\mathrm{f}}(k+1) &= [u^{\mathrm{o}}(k+1|k); \ldots; u^{\mathrm{o}}(k+N-1|k); u^{\mathrm{f}}(k+N|k+1)],\end{aligned}$$

with

$$\begin{aligned}x^{\mathrm{f}}(k+N+1|k+1) &= Ax^{\mathrm{o}}(k+N|k) + B\kappa_N(x^{\mathrm{o}}(k+N|k)),\\ u^{\mathrm{f}}(k+N|k+1) &= \kappa_N(x^{\mathrm{o}}(k+N|k)).\end{aligned}$$

If the real plant behavior is identical to the model used in the MPC controller this candidate solution satisfies constraints (1.11e) at time $k+1$. Conditions (1.12a)-(1.12c) respectively guarantee that constraint (1.11b) for $i = N-1$, constraint (1.11d) for $i = N-1$ and constraint (1.11c) are satisfied, which proves that $\mathbf{x}^{\mathrm{f}}(k+1), \mathbf{u}^{\mathrm{f}}(k+1)$ are feasible solutions to (1.11) at time $k+1$. This proves recursive feasibility, which, by induction on k, shows that (1.11) is feasible for $k \in \mathbb{N}$.

In order to prove that the resulting closed-loop system is asymptotically stable for all feasible initial states $x(0) \in \mathcal{X}_0$, we show that $J^{\mathrm{o}}(x(k))$, the optimal objective function value of (1.11) for current state $x(k)$ is a valid Lyapunov function. Therefore, it is sufficient to show that $J^{\mathrm{f}}(x(k+1)) < J^{\mathrm{o}}(x(k))$, with $J^{\mathrm{f}}(x(k+1))$ the objective function value corresponding to the feasible values $\mathbf{x}^{\mathrm{f}}(k+1), \mathbf{u}^{\mathrm{f}}(k+1)$. Straightforward algebraic manipulation yields

$$\begin{aligned}J^{\mathrm{o}}(x(k)) - J^{\mathrm{f}}(x(k+1)) = \|x(k|k)\|_Q^2 + \|u(k|k)\|_R^2 - \|x^{\mathrm{o}}(k+N|k)\|_Q^2\\ + \|x^{\mathrm{o}}(k+N|k)\|_{Q_N}^2 - \|\kappa_N(x^{\mathrm{o}}(k+N|k))\|_R^2\\ - \|Ax^{\mathrm{o}}(k+N|k) + B\kappa_N(x^{\mathrm{o}}(k+N|k))\|_{Q_N}^2.\end{aligned}$$

Due to condition (1.12d) one can see that indeed $J^{\mathrm{f}}(x(k+1)) < J^{\mathrm{o}}(x(k))$ and therefore, since $J^{\mathrm{o}}(x(k+1)) \leq J^{\mathrm{f}}(x(k+1))$, also that $J^{\mathrm{o}}(x(k+1)) < J^{\mathrm{o}}(x(k)), \forall x(k) \neq 0$. It can also be shown that $J^{\mathrm{o}}(x)$ is a convex function and that it hence satisfies all conditions of Lemma 1.1. This proves semi-global asymptotic stability. □

Note that controllability of the given system is no explicit condition in the above lemma. However, controllability is subsumed by condition (1.12c), because a set $\mathcal{X}_N$ satisfying this condition can only be found if κ_N stabilizes the given system, which in turn is only possible of the system is controllable.

Furthermore, it is also important to note that the terminal control law is not explicitly used in the on-line algorithm, but only is a theoretical tool to prove stability and recursive feasibility.

Chapter 3 will extend this stability theory to robust MPC, whereas later chapters will introduce stability notions for other control settings like tracking. In all cases the same two-step procedure is to be followed: a) recursive feasibility, b) stability.

1.6 Set Invariance

As mentioned in the previous section, the inclusion of a terminal constraint set in the MPC optimization problem is aimed at obtaining recursive feasibility. In summary, If the terminal constraint set satisfies conditions (1.12a)-(1.12c) then recursive feasibility of Algorithm 1.2 is guaranteed. However, it is not immediately clear how to construct such sets. The theory of *invariant sets* deals with exactly this problem and plays an important role in this thesis.

Definition 1.18 (**Positive invariance).** *A set $\mathcal{S} \in \mathbb{R}^{n_x}$ is positive invariant with respect to the autonomous system*

$$x(k+1) = g(x(k)), \qquad k \in \mathbb{N}, \tag{1.13}$$

with $x(k) \in \mathbb{R}^{n_x}, \forall k$ and $g : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ iff

$$g(x) \in \mathcal{S}, \qquad \forall x \in \mathcal{S}. \tag{1.14}$$

Hence, a positive invariant set with respect to some dynamics, if the dynamics are guaranteed to keep the state inside the set once it has entered. The term 'positive' refers to the fact that only forward predictions are considered and will be omitted in future sections for reasons of brevity. We can now see that condition (1.12c) is equivalent to requiring that $\mathcal{X}_N$ is invariant with respect to the closed-loop terminal dynamics $x(k+1) = Ax(k) + B\kappa_N(x(k))$.

Definition 1.19 (**Feasibility).** *Set $\mathcal{S}$ is feasible with respect to constraints* (1.2b) *iff* $\mathcal{S} \subseteq \mathcal{X}$.

Conditions (1.12a)-(1.12b) are equivalent to demanding that $\mathcal{X}_N$ is feasible with respect to $\mathcal{X}' \triangleq \mathcal{X} \cap \{x \in \mathbb{R}^{n_x} | \kappa_N(x) \in \mathcal{U}\}$.

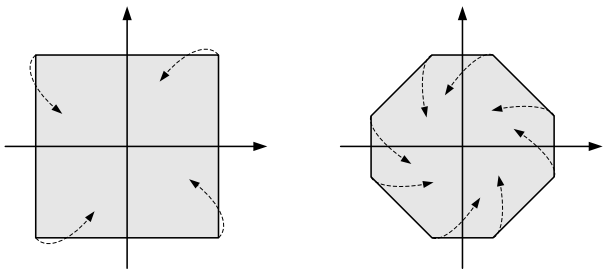


Figure 1.8: Left: Polyhedral non-invariant set. **Right:** Polyhedral invariant set. Trajectories starting from the vertices of the sets are depicted as dashed lines.

In order to maximize the feasible region, it is obvious that one would like to find the largest possible feasible invariant set. This set is called the *Maximal Admissible Set* or MAS [52]. Different algorithms exist depending on the model class for which an invariant set has to be constructed and the type of set (polyhedral, ellipsoidal, parallellotopic, ...) one wants to obtain. This will be discussed in more detail in Chapter 2.

The ability to construct feasible invariant sets for a given system and given constraints is the determining factor whether a recursively feasible MPC algorithm can be constructed. As will become clear in later chapters, invariant sets can be used in MPC algorithms in several different ways, but the invariance condition (1.14) always plays a key role in proving recursive feasibility.

Furthermore, the representation of the set (ellipsoidal, V-polytope, H-polytope) determines the computational efficiency of the resulting MPC algorithm and the size of the set determines the feasible region of the MPC algorithm. Chapter 5 will show how to make a trade-off between the complexity (of representation) and the volume of an invariant set.

1.7 Design goals

This section is aimed at clarifying the different goals when designing an MPC controller or when constructing an MPC algorithm and how these are influenced by the different elements present in the quasi-infinite horizon MPC scheme introduced in the previous section. It will become clear that some of these goals are conflicting, which will illustrate the need for improved algorithms and as such forms the main rationale for this thesis.

1.7.1 Stability

One of the most important aims is obviously to obtain a controller that stabilizes the system according to some stability measure. In the case of Algorithm 1.2 one obtains semi-global asymptotic stability. However, one should note that conditions (1.12) are *sufficient* conditions and therefore it is also possible that a stable controller is obtained if these conditions are not satisfied. Also in the case of Algorithm 1.1 one can obtain stable behavior under certain conditions systems if the prediction horizon N is chosen sufficiently large, which is typically what is being done in practice.

However, more or less since the appearance of the overview article by Mayne *et al.* [82], there is general agreement that newly proposed MPC algorithms should incorporate measures that guarantee recursive feasibility and stability in a sense relevant to the control problem they aim to solve, i.e stabilization, tracking or disturbance rejection.

1.7.2 Feasible region

An issue related to stability is the *feasible region* $\mathcal{X}_0$ that is defined as follows:

$$\mathcal{X}_0 \triangleq \{x(0) \in \mathbb{R}^{n_x} | \text{ the MPC optimization problem is feasible } \forall k\}. \tag{1.15}$$

For MPC algorithms that are guaranteed to be recursively feasible, this set is equal to the set of initial states for which the optimization problem is initially feasible (i.e. at time $k = 0$). Typically this is also the region for which stability is guaranteed. As a result another important aim is to obtain a feasible region that is as large as possible. However, e.g. incorporating stability measures, like introducing a terminal constraint, typically significantly reduces the size of the feasible region. On the other hand, larger values of N generally result in a larger feasible region.

1.7.3 Local optimality

MPC controllers are typically used when imposed constraints play an important role in the control problem and as such typically influence the obtainable control performance significantly compared to the unconstrained case. However, when operating in regions further away from the imposed constraints, e.g. close to the origin, it is desirable that the behavior of the MPC controller closely resembles that of an optimal unconstrained controller, e.g. an LQR controller. This property is referred to as *local optimality*.

Two ways of improving local optimality of an MPC controller are increasing the prediction horizon N or, if N is kept small, improving optimality of the terminal control law κ_N. However, typically more optimal terminal control laws (e.g. the LQR-optimal) lead to relatively small terminal constraint sets, whereas large terminal constraint sets typically correspond to suboptimal terminal controllers. Therefore increasing N seems to be the only compromise-free possibility for obtaining local optimality.

1.7.4 Computational complexity

As already explained MPC consists of on-line solving an optimization problem at every time instant k. As a consequence the computational complexity of solving this optimization problem cannot exceed a certain threshold since the calculations have to be finished within one sample interval. This is one of the main reasons why convex optimization problems are preferable in MPC, since many solvers for such optimization problems have known worst-case bounds on their computational complexity.

Computational complexity is mainly determined by the number of optimization variables, the number of constraints and the class of optimization problems. The number of optimization variables typically increases proportional to N, which obviously limits the length of the horizon that one can choose. Therefore, it is obvious that there is a clear trade-off between computational complexity and the two previous design goals: feasibility and local optimality.

The number of constraints also increases proportional to N but is also partly determined by the complexity of the description of the terminal constraint set $\mathcal{X}_N$, e.g. the number of inequality constraints, if $\mathcal{X}_N$ is a polytopic set.

Finally, the optimization class greatly influences the computational complexity, because the exponents of the polynomial scaling of the computational complexity as a function of the problem size are class-dependent. Therefore it is preferable to limit the on-line optimization to either LP or QP. SOCP and SDP are well-suited for off-line use during the design phase, but less appropriate for on-line application.

1.7.5 Robustness

Finally, another important aim of MPC controller design is its robustness with respect to differences between the prediction model and the plant model and robustness with respect to external disturbances that act upon the system. As will be shown in the next chapter it is possible to incorporate measures to guarantee robustness, but this goes at a significant cost of the computational complexity.

1.7.6 Conclusion

The five main design goals for MPC controllers – stability, feasibility, local optimality, low complexity and robustness – can be obtained by changing different tuning parameters in the MPC controller design. However, the different goals are often not obtainable simultaneously, which is not necessarily a property of the problem description, but a property of the classical MPC algorithms. Different solutions will be presented in the next chapters, in order to alleviate some of these compromises.

1.8 General outline of this thesis

The main goal of this thesis is to develop new algorithms that satisfy all the design goals explained in the previous section, or at least satisfy these design goals better than existing algorithms. This section therefore gives an outline of this thesis and will highlight how the aforementioned design goals are met.

First of all we restrict ourselves to **robust MPC algorithms**, that have guaranteed **stability** for a certain *class* of systems rather than a single system. To this end Section 1.5 introduced the necessary concepts that are relevant in the construction of MPC algorithms with recursive feasibility and guaranteed stability for the nominal (i.e., non-robust) case. Later in this thesis these concepts are then extended to the robust case.

In the robust case the issue of recursive feasibility (and hence also asymptotic stability) becomes somewhat more complicated and additional measures have to be taken. A distinction between *open-loop* and *closed-loop* predictions can be made. In the nominal case both approaches result in identical behavior and only lead to numerical differences, but in the robust case the differences between both methods become crucial and have important implications on recursive feasibility. **Chapter 3** will give an assessment of the differences between both approaches and will show that the use of *closed-loop* predictions is essential for obtaining robust MPC algorithms that satisfy the design goal of guaranteed stability.

Secondly, the aim is to construct MPC algorithms that maintain the property of **local optimality**. To this end, we restrict ourselves to robust MPC algorithms that allow the incorporation of a local control law, such as the quasi-infinite horizon MPC algorithm discussed in Section 1.5 or general interpolation based robust MPC. Furthermore, the requirement of local optimality eliminates MPC algorithms based on L_1 and L_∞ norms, since these can result in non-optimal local control behavior. This eliminates algorithms based on linear programming (LP), which has an implication on the computational efficiency of the obtained algorithms.

In the majority of cases, non-LP based robust MPC algorithms make use of semi-definite programming (SDP) because typically ellipsoidal invariant sets are used to ensure recursive feasibility. The computational complexity of such optimization problems is often prohibitively high for practical applications and therefore the aim here is to obtain robust MPC algorithms that make use of quadratic programming (QP). To this end we aim to use **polyhedral invariant sets** instead of ellipsoidal ones. Therefore, **Chapter 2** will introduce the basic principles of invariant sets and some existing algorithms for the construction of these sets. Some important properties, such as the structure of the resulting sets will also be discussed, since these insights will be useful in later chapters, where further complexity reductions will be obtained based on these insights. **Chapter 4** will then introduce several robust MPC algorithms that make use of polyhedral invariant sets. In this way the design goal of obtaining algorithms with a **favorable computational complexity** is met, at least for low-dimensional systems.

It is important to point out that the algorithms introduced in **Sections 4.3 and 4.2.5** allow the entire constraint structure of the on-line optimization to be computed by constructing a single polyhedral invariant set for an appropriately constructed augmented system. In this way, improving the algorithms for the construction of such invariant sets will have a more profound impact on the performance and efficiency of these MPC algorithms, which can be seen as both an advantage and a disadvantage.

The algorithms discussed in Chapter 2 have some disadvantages with respect to their scaling behavior. For high-dimensional systems, these algorithms can result in exponentially increasing computational requirements and can return polyhedral invariant sets with a prohibitively large number of inequality constraints, which in turn can lead to computationally inefficient MPC algorithms. Therefore, **Chapter 5** will introduce algorithms for the construction of **reduced-complexity polyhedral invariant sets**. This will enable the construction of robust MPC algorithms with **lower computational complexity** that are hence applicable to higher-dimensional systems. More specifically, it can be proven that under certain conditions the scaling behavior of the algorithm introduced in Section 4.3 will become linear as a function of the horizon length instead of exponential.

This improved scaling behavior will enable the use of significantly longer prediction horizons. In this way the resulting algorithms will have bf enlarged feasible regions compared to the feasible regions of algorithms using full-complexity polyhedral invariant sets or ellipsoidal invariant sets.

In order to further enlarge the size of the operating region of the obtainded MPC controllers, without compromising other design goals such as low computational complexity, **Chapters 6 and 7** will investigate reduced complexity control-invariant sets and their use in robust MPC algorithms. Chapter 7 will show that control-invariant sets allow the **feasible region** to be **extended towards the theoretical maximum**.

Chapter 8 finally shows that the obtained algorithms also give favorable results when applied to models of a few real-life systems.

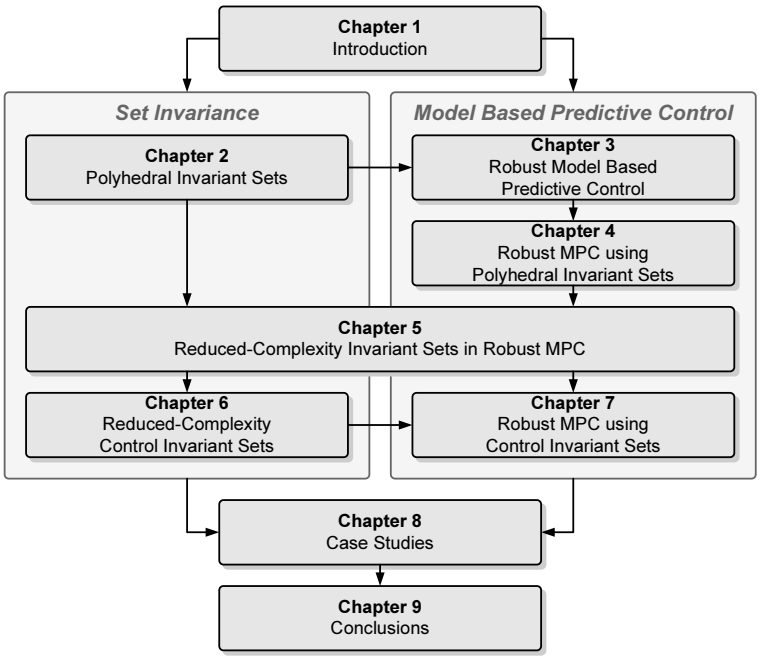


Figure 1.9: Overview and connection between the different chapters in this thesis. Arrows suggest possible reading trajectories.

1.9 Chapter-by-chapter overview

Figure 1.9 gives an overview of the different chapters in this thesis and how they relate to each other. We will now give an overview of the different contributions of each chapter:

Chapter 2: This chapter concerns the construction of invariant sets for linear systems with polytopic uncertainty description. These sets can be used to formulate several different MPC algorithms. Combining several elements already in literature an algorithm is formulated that is able to efficiently construct polytopic invariant sets for such systems. These sets have the advantage that they lead to more efficient MPC optimization problems and that the feasible region is typically increased significantly. The main contribution of the chapter, however, is the analysis of the structure of polytopic invariant sets, which paves the way for the results presented in Chapter 5.

Chapter 3: This chapter explains how Algorithm 1.2 can be extended in order to explicitly incorporate measures for guaranteeing robustness with respect to model uncertainty or disturbances. Special attention is given to the possible ways for computing within-horizon predictions despite the uncertainties and to how this influences recursive feasibility. The chapter describes and corrects two algorithms presented in recent publications that falsely claim to be recursively feasible. Counterexamples are

also provided.

Chapter 4: In this chapter several MPC algorithms are presented that make use of the polytopic invariant sets discussed in the previous chapter. A first set consists of the interpolation-based algorithms, while the second type of algorithm presented is a robust and improved version of the quasi-infinite horizon MPC algorithm presented in Chapter 1. Both types of algorithms are suited for different kinds of problems.

Chapter 5: The results presented in this chapter are an extension of those presented in Chapter 2. The polytopic invariant sets discussed there have an impractically large number of constraints under certain conditions which limits the use of the algorithms presented in Chapter 4 for larger-dimensional systems. This chapter analyzes the cause of this problem and proposes a new algorithm for constructing reduced-complexity versions of such sets. In this way the use of polytopic sets in MPC algorithms becomes more advantageous.

Chapter 6: This chapter forms a further extension of the results from Chapter 5. The construction of control invariant sets is discussed. This types of sets are also potentially useful in MPC algorithms. For the same reasons as in the previous chapters, reduced-complexity versions of these sets are investigated. As a side-result the construction of reduced-complexity projections of polytopic sets is also discussed.

Chapter 7: In this chapter the use of controlled invariant sets in MPC algorithms is explored. It is shown that control invariant sets allow the extension of the feasible region to the theoretical limit. Another advantage of the use of controlled invariant sets is that tracking problems can also be tackled.

Chapter 8: Whereas in the previous chapters the new methods are illustrated using simple numerical examples, this chapter demonstrates the new algorithms using more real-life systems.

1.10 Specific contributions of this thesis

This thesis makes contributions in two main areas: set invariance theory and robust model based predictive control. As indicated in Figure 1.9 the contributions in these two areas are not discussed chronologically in this thesis, but here we will treat both aspects separately.

Set invariance

A first contribution in this area is given in Chapter 2, where the structure of polyhedral invariant sets for LPV systems is discussed. It is observed that redundant constraints induce sparsity in the tree structure. Furthermore, the depth of the tree is linked to the Joint Spectral Radius (JSR) of the given autonomous system. These two insights are crucial for the development of new algorithms in later chapters. References are

[98, 101].

The second and most important contribution of this thesis in the area of set invariance is the introduction in Chapter 5 of new algorithms for the construction of reduced-complexity polyhedral invariant sets, i.e. invariant sets that are described by a reduced number of linear inequality constraints. Two different methods are introduced:

- **Pruning.** This method reduces the number of constraints by means of tightening certain constraints with a small factor. The algorithm detects when such constraint tightening can make other constraints redundant and calculates the exact tightening factor to accomplish this. For efficiency reasons, this constraint tightening is done during the construction of the invariant sets.

- **Trimming.** This method is based on a new theorem that when modifying the system matrices in a certain way, any invariant set for the modified system will also be invariant for the original system. These additional degrees of freedom can then be exploited in order to reduce the number of constraints of the resulting polyhedral invariant sets.

Both algorithms can be interpreted in terms of the tree structure discussed in Chapter 2. The aim of pruning is the reduction of the number of parallel branches in the tree structure, while the aim of trimming is reducing the depth of the tree.

Both methods can also be linked to the JSR of the system. In the case of pruning, the JSR determines the amount of constraint tightening that can be performed without loosing convergence of the algorithm, while in the case of trimming, the JSR provides a heuristic for optimally choosing the parameters involved in modifying the system matrices.

In both methods a trade-off can be made between maximal volume and minimal complexity. This regularization-like approach to the construction of invariant sets is novel and not found elsewhere in literature.

Finally, it is proven in Section 5.3 that pruning, when applied to the algorithm of Section 4.3 can lead to lead to linear (rather than exponential) scaling behavior as a function of the horizon length. References are [94, 106].

A third contribution in the area of set invariance is the introduction in Chapter 6 of algorithms for the construction of reduced-complexity control invariant sets. These sets are similar to invariant sets but also allow inputs to be present in the given systems. As a result, algorithms for constructing such sets typically involve additional steps where projections of intermediate polytopic sets have to be calculated.

First of all the methods of pruning and trimming are generalized towards the setting of computing control invariant sets. Secondly, new algorithms are proposed for computing approximate reduced-complexity projections of polytopic sets. The obtained algorithms allow a trade-off to be made between maximal volume and minimal complexity. At the time of writing, these results are not yet published elsewhere.

Model based predictive control

Several contributions have also been made in the area of the synthesis of robust MPC algorithms. Some of these contributions are based on improvements in the area of set invariance, as discussed above, others are not directly linked to such advances.

A first contribution in this area is the identification of theoretical misconceptions regarding some existing robust MPC algorithms. Sections 3.4 and 3.5 highlight and correct errors present in the algorithms discussed in [142] and [31] respectively. These corrections have been published in [105, 143] and [94] respectively.

A second contribution is the extension of several robust MPC algorithms towards the use of polyhedral invariant sets instead of ellipsoidal invariant sets. Three different algorithms are extended to this setting:

- Section 4.1 extends the results from [68] and describes how one can use polyhedral invariant sets for obtaining robust linear feedback controllers with improved optimality and less conservative constraint handling. These results can either be used on-line in a receding horizon approach, or can be used for computing locally optimal controllers for use in other MPC algorithms. These results are published in [93].

- Section 4.2 shows how robust interpolation based MPC can be implemented using polyhedral invariant sets. These results are published in [98], which received the Student Best Paper Award of the 2005 American Control Conference.

- Section 4.3 shows how one can use polyhedral invariant sets for constructing robust MPC controllers with quasi-infinite horizon. These results are published in [100].

A third contribution is the improvement of constraint handling in interpolation based MPC algorithms on an algorithmic level. The new algorithm is able to take the interaction between the different linear control laws into account and uses this information to reduce the conservativeness of constraint handling. These results are published in [118, 120].

A fourth contribution consists of the extension of the algorithms discussed in Chapter 4 towards the use of reduced-complexity polyhedral invariant sets. This allows these algorithms to be used for higher-order models and allows the use of longer prediction horizons (if applicable). More specifically, it is shown that under certain conditions when using the quasi-infinite horizon MPC algorithm discussed in Section 4.3 in conjunction with reduced-complexity polyhedral invariant sets, one obtains linear scaling behavior as a function of the horizon length.

A fifth contribution is the extension of the concept of general interpolation to interpolation between non-linear control laws, like e.g. different MPC controllers, as described in Section 7.2. At the time of writing, these results are not yet published

elsewhere.

A sixth and final contribution is a method to enlarge the operating region of any given recursively feasible MPC controller to the maximal control admissible set for the given system. This is discussed in Section 7.3. At the time of writing, these results are not yet published elsewhere.

Chapter 2

Polyhedral Invariant Sets

"When in doubt, predict that the present trend will continue."

– Merkin's Maxim –

This chapter extends the concept of invariant sets towards the robust case. Whereas previous chapters illustrated the use of invariant sets, this chapter focusses on how to construct such sets for autonomous LPV systems. A literature overview is given, after which an efficient algorithm is presented for constructing the Maximal Admissible Set (MAS) using linear programming. Extensions to this algorithm are discussed and some numerical examples are given. In the following chapters these invariant sets are then used in robust MPC algorithms and further extended.

2.1 Set invariance

The basic concepts of set invariance were already discussed in Chapter 1, so we will only briefly repeat the main definitions and properties for the specific case of autonomous LPV systems subject to bounded disturbances and subject to linear constraints.

2.1.1 Definitions

In this chapter we consider autonomous LPV systems subject to bounded disturbances

$$x(k+1) = \Phi(k)x(k) + w(k), \qquad k \in \mathbb{N}, \tag{2.1}$$

with $\Phi(k) \in \mathbb{R}^{n_x \times n_x}$ belonging to an uncertainty polytope $\Omega' \subset \mathbb{R}^{n_x \times n_x}$ defined as

$$\Phi(k) \in \Omega' \equiv \mathrm{Co}\{\Phi_1, \ldots, \Phi_r\}, \qquad k \in \mathbb{N}, \tag{2.2}$$

or equivalently

$$\Phi(k) \in \left\{ \sum_{j=1}^{r} \lambda_j(k)\Phi_j \;\middle|\; \lambda_1(k) \geq 0, \ldots, \lambda_r(k) \geq 0, \sum_{j=1}^{r} \lambda_j(k) = 1 \right\}, \quad k \in \mathbb{N}. \tag{2.3}$$

The disturbance input $w(k)$ is bounded by a V-polytope

$$w(k) \in \mathcal{W} \triangleq \mathrm{Co}\{w_1, \ldots, w_l\}, \qquad k \in \mathbb{N}, \tag{2.4}$$

and the state $x(k)$ is constrained to (1.2b). The reason for this specific uncertainty representation will be explained in more detail in Chapter 3. We now redefine invariance for this class of systems.

Definition 2.1 (**Robustly positive invariant set**, [14])**.** *A set $\mathcal{S} \in \mathbb{R}^{n_x}$ is robustly positive invariant with respect to the system* (2.1) *iff*

$$\Phi x + w \in \mathcal{S}, \qquad \forall x \in \mathcal{S}, \forall \Phi \in \Omega', \forall w \in \mathcal{W}. \tag{2.5}$$

In future sections, when appropriate, the term $\langle \Omega', \mathcal{W} \rangle$-invariant will be used. If the set $\mathcal{S}$ is convex, then the following condition is equivalent to (2.7)

$$\Phi_i x + w_j \in \mathcal{S}, \qquad \forall x \in \mathcal{S}, i = 1, \ldots, r, j = 1, \ldots, l. \tag{2.6}$$

A related concept is that of λ-contractive positive invariant sets and can be defined as follows:

Definition 2.2 (**λ-contractive robustly positive invariant set**, [13])**.** *A set $\mathcal{S} \in \mathbb{R}^{n_x}$ is a λ-contractive (with $\lambda \in (0, 1]$) robustly positive invariant set with respect to system* (2.1) *iff*

$$\Phi x + w \in \lambda\mathcal{S}, \qquad \forall x \in \mathcal{S}, \forall \Phi \in \Omega', \forall w \in \mathcal{W}, \tag{2.7}$$

with the scalar multiplication of sets defined as $\lambda\mathcal{S} \triangleq \{\lambda x | x \in \mathcal{S}\}$ as in [63, 146]. In future sections, when appropriate, the term $\langle \Omega', \mathcal{W}, \lambda \rangle$-invariant will be used.

The definition of a *feasible set* is already given in Chapter 1 and is hence not repeated. A related definition is that of *output admissible sets*. Since all output constraints considered in this thesis can be mapped to equivalent state constraints, we will use the term *admissible set* in this thesis:

Definition 2.3 (**Admissible set).** *A set $\mathcal{S}$ is admissible with respect to* (2.1),(1.2b) *iff $x(0) \in \mathcal{S}$ implies that all future states $x(k), k \in \mathbb{N}$ satisfy constraint* (1.2b)*:*

$$x(0) \in \mathcal{S} \quad \Rightarrow \quad x(i) \in \mathcal{X}, \qquad \forall \Phi(k) \in \Omega',\ \forall w(k) \in \mathcal{W}, \quad k = 1, \ldots, i-1, \quad i \in \mathbb{N}_0. \tag{2.8}$$

Please note that an admissible set is not necessarily positive invariant. Admissibility is related to the size of a set (because $\mathcal{X}$ contains the origin in its interior, any sufficiently small neighborhood around the origin is an admissible set) whereas positive invariance is related to the shape of the set. Conversely, one can see that all feasible positive invariant sets are admissible.

Definition 2.4 (**Maximal Admissible Set (MAS, [52])**). *The maximal admissible set* $\overline{\mathcal{S}}$ *for system* (2.1) *and constraints* (1.2b) *is defined as the set of* all *initial states for which the corresponding trajectories under the autonomous dynamics guarantee constraint satisfaction:*

$$\overline{\mathcal{S}} \triangleq \{x(0) | x(i) \in \mathcal{X}, \forall \Phi(k) \in \Omega', \forall w(k) \in \mathcal{W}, k = 0, \ldots, i-1, i \in \mathbb{N}_0\}. \quad (2.9)$$

2.1.2 Properties

The properties given here will be used either explicitly or implicitly in the following sections and are stated here as background material for better understanding of the following sections.

Property 2.1. *If* $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{n_x}$ *are invariant sets with respect to* (2.1) *then the following sets are also positive invariant with respect to* (2.1)*:*

a) $\mathcal{A} \cap \mathcal{B}$,
b) $\mathcal{A} \cup \mathcal{B}$,
c) $\mathrm{Co}\{\mathcal{A}, \mathcal{B}\}$.

Please note that part a) states the opposite of [63, Remark 2.3]. However, one can clearly see that if $x(k) \in \mathcal{A}$ and $x(k) \in \mathcal{B}$, positive invariance guarantees that $x(k+1) \in \mathcal{A}, x(k+1) \in \mathcal{B}$ and hence $x(k+1) \in \mathcal{A} \cap \mathcal{B}$, which proves part a) of the above property.

The following fundamental property, the first reference to which can be found in [52], validates the use of the MAS as a terminal constraint set in MPC algorithms:

Property 2.2. *The MAS for a given system is the largest positive invariant set for that system, if it exists.*

Due to the model class and type of constraints under consideration in this thesis, the following property also holds:

Property 2.3. *The MAS for* (2.1),(1.2b) *is a convex set, if it exists.*

These two properties form the main reasons why the MAS is the preferred type of positive invariant set to be used as a terminal constraint in MPC.

2.2 State of the art

In this section we give a brief overview of the state of the art concerning the construction of the MAS or invariant inner approximations of the MAS for LPV systems. We start off with the LTI case, since this technique lies at the basis of what is described in the latter part of this chapter. In the remainder of this thesis we focus on the case of polytopic input and state constraint sets containing the origin in their interior:

$$x(k) \in \mathcal{X} \triangleq \{x \in \mathbb{R}^{n_x} | A_x x \leq \mathbf{1}\}, \qquad k \in \mathbb{N}, \quad (2.10)$$

$$u(k) \in \mathcal{U} \triangleq \{u \in \mathbb{R}^{n_u} | A_u u \leq \mathbf{1}\}, \qquad k \in \mathbb{N}, \tag{2.11}$$

which is not a major restriction in the MPC setting. In future sections m_x, m_u are used to denote the number of rows in A_x and A_u respectively.

2.2.1 Maximal admissible set for LTI systems

Although the concept of positive invariant sets was already studied in the 80's (e.g. [11, 12, 56]), the link with admissible sets and more specifically maximal admissible sets was made in [52]. The paper presents both theoretical and algorithmic results concerning construction of the MAS for deterministic discrete-time non-linear systems subject to a generic class of inequality constraints. In this section we will describe the results for the LTI case, whereas the following sections discuss results for the case with model uncertainty.

In this section we consider autonomous LTI systems, i.e. systems of the form (2.1) with $r = 1$ and $\Omega = \{\Phi\}$, subject to constraints (1.2b). The results of [52] make use of sets $\mathcal{O}_k$ defined as

$$\mathcal{O}_k = \{x | x \in \mathcal{X}, \Phi x \in \mathcal{X}, \ldots, \Phi^k x \in \mathcal{X}\}. \tag{2.12}$$

Two important results of [52] that are of relevance in this thesis can be summarized as follows:

- The MAS for an LTI system subject to a constraint set $\mathcal{X}$ is equal to $\mathcal{O}_\infty$.
- If the eigenvalues of Φ satisfy $|\lambda_i| \leq 1$, there exists $k^\star \in \mathbb{Z}^+$ such that $\mathcal{O}_\infty = \mathcal{O}_{k^\star}$.

No explicit values for $k^\star$ are given, but the following algorithm iteratively circumvents this problem:

Algorithm 2.1 (MAS for LTI systems). *Given an autonomous LTI system* $x(k+1) = \Phi x(k)$ *subject to constraints* (2.10)*, set* $i := 1$ *and perform the following steps:*

1. *if* $\mathcal{O}_{i+1} = \mathcal{O}_i$ *then go to step 3,*
2. *set* $i := i + 1$ *and go to step 1*
3. *return* $\mathcal{O}_\infty \equiv \mathcal{O}_i$ *and* $k^\star \equiv i$.

The variable $k^\star$ is called the *admissibility index* and indicates how many time steps ahead one has to enforce the constraints in order to ensure positive invariance of the resulting set. The practicality of Algorithm 2.1 depends on how efficient the condition in step 1 can be verified. The concepts of *constraint significance* and *constraint redundancy*, which were not used explicitly in [52] but are introduced here already with further chapters in mind, are helpful in this respect:

Definition 2.5 (Constraint significance). *The significance* $sig_{\mathcal{S}}(a^{\mathrm{T}})$ *of a constraint* $a^{\mathrm{T}} x \leq 1$ *with respect to a set* $\mathcal{S}$ *containing the origin, is defined as*

$$sig_{\mathcal{S}}(a^{\mathrm{T}}) \triangleq \max_{x \in \mathcal{S}} a^{\mathrm{T}} x. \tag{2.13}$$

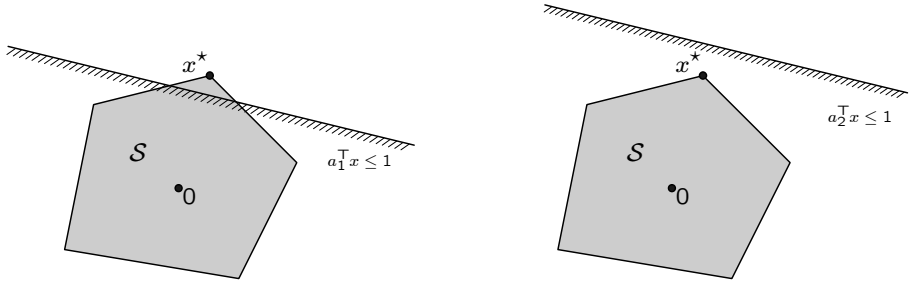


Figure 2.1: Illustration of the significance of a constraint. Constraint $a_1^{\mathrm{T}} x \leq 1$ has a significance larger than 1, while constraint $a_2^{\mathrm{T}} x \leq 1$ has a significance smaller than 1. The optimal x-vector resulting from (2.13) is indicated as $x^\star$.

The redundance of a constraint is defined as the reciprocal of the significance:

Definition 2.6 (Constraint redundance). *The redundance $r_{\mathcal{S}}(a^{\mathrm{T}})$ of a constraint $a^{\mathrm{T}} x \leq 1$ with respect to a set $\mathcal{S}$ containing the origin, is defined as $red_{\mathcal{S}}(a^{\mathrm{T}}) \triangleq \frac{1}{sig_{\mathcal{S}}(a^{\mathrm{T}})}$.*

In order to make a distinction between vectors that represent points in space or vectors that represent constraint coefficients, we use the same convention as used in [146] and denote points in space by column vectors (e.g., a) and denote constraint coefficient vectors as row vectors (e.g., b^{T}).

The interpretation of the significance of a constraint is straightforward. If the significance is strictly larger than 1, then $\mathcal{S}$ would decrease in size if it would be intersected with $a^{\mathrm{T}} x \leq 1$. A significance strictly smaller than 1 implies that the constraint lies outside the set $\mathcal{S}$. This is illustrated in Figure 2.1. In this way step 1 of Algorithm 2.1 can be performed by calculating the significance of the constraints of $\mathcal{O}_{i+1}$ with respect to $\mathcal{O}_i$. If all significance values are ≤ 1, then the condition $\mathcal{O}_{i+1} = \mathcal{O}_i$ is satisfied and the algorithm can terminate. In Algorithm 2.1 all intermediate sets are polytopes and hence the optimization (2.13) reduces to an LP. This indicates that Algorithm 2.1 can be implemented and executed efficiently.

Extensions towards the inclusion of bounded disturbances are possible and are based upon the same principles as those used in for the construction of LPV systems shown in Figures 2.5 and 2.6. We refer to [51, 63, 67] for more details.

2.2.2 Ellipsoidal invariant sets for LPV systems

The construction of invariant sets for LPV systems is somewhat more complex, since for such systems the current state does not uniquely define the future trajectory that will be followed. All possible trajectories have to be guaranteed to be feasible, which seems to imply a significant increase in the number of constraint necessary to describe invariant sets for this class of systems. Therefore, in MPC algorithms for LPV systems, the use of ellipsoidal invariant sets prevailed until recently [3, 31, 92, 142]. The main advantages are the fact that the complexity of description of an ellipsoidal set uniquely

depends on its dimensionality and the fact that such sets can be constructed by solving a single convex optimization problem. One can distinguish between two different cases, depending on whether or not a feedback control law for controlling the open-loop system is already known.

In the first case, no control law is known in advance and hence needs to be calculated together with a corresponding invariant set. This problem was solved in [68] and is summarized in Appendix A.

In the second case, one would like to construct an invariant ellipsoid for a given closed loop system (2.1)-(2.2) subject to constraints (2.10). In this case the following simpler method can be used [23]:

Algorithm 2.2 (**Ellipsoidal invariant set for LPV systems).** *Given a system* (2.1)-(2.2) *subject to constraints* (2.10)*, solve the following optimization problem:*

$$\max_{Z \in \mathbb{S}^{n_x}_{++}} \quad \det Z^{-1}, \tag{2.14a}$$

$$\text{s.t.} \quad \begin{bmatrix} Z & * \\ \Phi_i Z & Z \end{bmatrix} \succ 0, \qquad i = 1, \dots r, \tag{2.14b}$$

$$\begin{bmatrix} Z & * \\ A_x(j,:)Z & 1 \end{bmatrix} \succ 0, \qquad j = 1, \dots, m_x. \tag{2.14c}$$

Construct an ellipsoidal invariant set $\mathcal{E} \triangleq \{x | x Z^{-1} x \leq 1\}$.

In the above optimization asterisks represent expressions that make the matrices symmetric. This avoids redundant notations, since Linear Matrix Inequalities (LMIs, [23, 24]) always need to consist of symmetric matrix expressions.

The above optimization problem is a so called *determinant maximization* problem, which can be shown to be convex and hence can be readily solved [24]. In case the main axes of the invariant ellipsoid can be expected to have lengths of the same order of magnitude, the objective of (2.14) can be well approximated by $\min_{Z \in \mathbb{S}^{n_x}_{++}} Tr(Z)$, which allows the use of standard SDP optimization algorithms.

Although the construction of ellipsoidal invariant sets is straightforward, there are several disadvantages associated with the use of ellipsoidal invariant sets

- **computational complexity**: due to the fact that an ellipsoidal set is defined by a quadratic inequality, MPC algorithms that make use of such sets automatically have to make use of SOCP or SDP optimization. However, for computational efficiency reasons, MPC algorithms are preferably formulated as LP or QP optimization problems.
- **restricted shape**: Ellipsoidal invariant sets are rather restricted in shape, due to the limited number of degrees of freedom associated with the fixed complexity of representation. Therefore, these sets are sometimes relatively conservative inner approximations of the real MAS of the system. This problem increases as the dimensionality of the system increases.
- **inherent symmetry**: Ellipsoidal invariant sets are by definition centered around the origin, since they are constructed as level sets of quadratic Lyapunov

functions. As a result they are point-symmetric with respect to the origin and can therefore only cope with asymmetric constraints in a conservative way.

These disadvantages can be eliminated by using polyhedral invariant sets. This problem is tackled in the next section.

2.2.3 Maximal admissible set for LPV systems

The main advantage of ellipsoidal invariant sets is their computability by means of solving a single convex optimization problem. The disadvantage is the fact that ellipsoidal invariant sets are only inner approximations of the MAS. However, the MAS typically cannot be constructed by solving a single optimization problem, but has to be constructed iteratively.

All current algorithms for constructing the MAS make iterative use of the *one-step controllability set*[1] $\text{Pre}_{\langle\Omega,\mathcal{U},\mathcal{W}\rangle}(\mathcal{S})$:

$$\text{Pre}_{\langle\Omega,\mathcal{U},\mathcal{W}\rangle}(\mathcal{S}) \triangleq \{x \in \mathbb{R}^{n_x} | \exists u \in \mathcal{U} : Ax + Bu + w \in \mathcal{S}, \forall [A\ B] \in \Omega, \forall w \in \mathcal{W}\}.$$

In a more general setting also a contraction constraints can be imposed, similar to the one imposed in Definition 2.2:

Definition 2.7 (λ-contractive one-step controllability set, [13,63]**).** *The λ-contractive controllability set ($\lambda \in (0, 1]$) of $\mathcal{S}$ with respect to uncertain dynamics Ω, input constraint set $\mathcal{U}$ and disturbance set $\mathcal{W}$ is defined as*

$$\begin{aligned} &\textit{Pre}_{\langle\Omega,\mathcal{U},\mathcal{W},\lambda\rangle}(\mathcal{S}) \triangleq \\ &\qquad \{x \in \mathbb{R}^{n_x} | \exists u \in \mathcal{U} : Ax + Bu + w \in \lambda\mathcal{S}, \forall [A\ B] \in \Omega, \forall w \in \mathcal{W}\}. \end{aligned} \quad (2.15)$$

In further sections this set will be referred to as the *pre-set*. In case no inputs are present this set is denoted as $\text{Pre}_{\langle\Omega,\mathcal{W},\lambda\rangle}(\mathcal{S})$. This shortened notation can be distinguished from the notation mentioned above, by checking the dimensions of Ω for the presence of inputs. In case no contractivity is demanded (i.e. $\lambda = 1$) and in case no disturbances are present, the shorthand notation $\text{Pre}_{\langle\Omega\rangle}(\mathcal{S})$ will be used.

Based on this tool, a necessary and sufficient condition for invariance can be formulated:

Lemma 2.1 (Geometric condition for positive invariance, [63]**).** *Given a system* (2.1), (2.2) *and a set $\mathcal{S} \in \mathbb{R}^{n_x}$, then the set $\mathcal{S}$ is $\langle\Omega', \mathcal{W}, \lambda\rangle$-invariant iff*

$$\mathcal{S} \subseteq \textit{Pre}_{\langle\Omega',\mathcal{W},\lambda\rangle}(\mathcal{S}). \quad (2.16)$$

Proof: (Sufficient) If $x(k) \in \mathcal{S}$, it follows from (2.16) that $x(k) \in \text{Pre}_{\langle\Omega',\mathcal{W},\lambda\rangle}(\mathcal{S})$. Due to Definition (2.7) it is then guaranteed that $\Phi x(k) + w \in \lambda\mathcal{S}, \forall\Phi \in \Omega', \forall w \in \mathcal{W}$, which proves that $\mathcal{S}$ is $\langle\Omega', \mathcal{W}, \lambda\rangle$-invariant.
(Necessary) If $\mathcal{S} \nsubseteq \text{Pre}_{\langle\Omega',\mathcal{W},\lambda\rangle}(\mathcal{S})$, then $\exists x(k) \in (\mathcal{S} \setminus \text{Pre}_{\langle\Omega',\mathcal{W},\lambda\rangle}(\mathcal{S}))$. Due to Definition (2.7) it is then guaranteed that $\exists\Phi \in \Omega', w \in \mathcal{W} : x(k) + w \notin \lambda\mathcal{S}$,

[1] By some authors the one-step controllability set is also referred to as the *one-step set* [63], the *preimage set* [16] or the *pre-set*.

which shows that in that case $\mathcal{S}$ cannot be $\langle \Omega', \mathcal{W}, \lambda \rangle$-invariant. Therefore (2.16) is a necessary condition for $\langle \Omega', \mathcal{W}, \lambda \rangle$-invariance. □

Based on the above condition a straightforward iterative strategy can be formulated for constructing the MAS:

Algorithm 2.3 ($\langle \Omega', \mathcal{W}, \lambda \rangle$-invariant set construction, [13]**).** *Given a system* (2.1),(2.2) *subject to constraints* (2.10) *and variables* $\lambda, \lambda' \in \mathbb{R}^+$ *such that* $\lambda \in (0, 1], \lambda' \in (0, 1], \lambda' \leq \lambda$*, perform the following steps:*

1. *Initialize* $\mathcal{O}_0 := \mathcal{X}, i := 0$.
2. *Execute iteratively until* $\mathcal{O}_i \subseteq Pre_{\langle \Omega', \mathcal{W}, \lambda \rangle}(\mathcal{O}_i)$*:*
   (a) *Set* $i := i + 1$.
   (b) *Calculate* $\mathcal{O}_i := Pre_{\langle \Omega', \mathcal{W}, \lambda' \rangle}(\mathcal{O}_{i-1}) \cap \mathcal{O}_{i-1}$.

Return the set $\overline{\mathcal{S}} \triangleq \mathcal{O}_i$ *and admissibility index* $k^\star \triangleq i$.

Theorem 2.1 (Maximal $\langle \Omega', \mathcal{W}, \lambda \rangle$-invariant set, [13]**).** *If Algorithm 2.3 terminates, the resulting set* $\overline{\mathcal{S}}$ *is* $\langle \Omega', \mathcal{W}, \lambda \rangle$*-invariant. Furthermore, if* $\lambda = \lambda'$*, the resulting set is the maximal* $\langle \Omega', \mathcal{W}, \lambda \rangle$*-invariant set.*

Proof: The termination condition implies that $\mathcal{O}_{k^\star} \subseteq \text{Pre}_{\langle \Omega', \mathcal{W}, \lambda \rangle}(\mathcal{O}_{k^\star})$. Due to condition 2.16 it is directly guaranteed that the set $\overline{S}$ is $\langle \Omega', \mathcal{W}, \lambda \rangle$-invariant. If $\lambda = \lambda'$, the resulting set can also be proven to be the maximal $\langle \Omega', \mathcal{W}, \lambda \rangle$-invariant set, meaning that every feasible $\langle \Omega', \mathcal{W}, \lambda \rangle$-invariant set is a subset of $\overline{\mathcal{S}}$. Assume there exists a $\langle \Omega', \mathcal{W}, \lambda \rangle$-invariant set $\mathcal{S}' \nsubseteq \overline{S}$. This implies that there exists a state $x \in (\mathcal{S}' \setminus \mathcal{O}_{k^\star})$. This in turn implies $\exists \Phi \in \Omega', w \in \mathcal{W} : \Phi x + w \notin \lambda' \mathcal{O}_{k^\star - 1} = \lambda \mathcal{O}_{k^\star - 1}$, while at the same time, for the same Φ, w it holds that $\Phi x + w \in \lambda \mathcal{S}'$. This means that $\Phi x + w \in (\lambda \mathcal{S}' \setminus \lambda \mathcal{O}_{k^\star - 1})$, which means $\exists x' \in (\mathcal{S}' \setminus \mathcal{O}_{k^\star - 1})$. By induction one can therefore see that $\exists x \in (\mathcal{S}' \setminus \mathcal{O}_0)$ which means that $\mathcal{S}' \nsubseteq \mathcal{X}$ and hence is not feasible. This proves that $\overline{\mathcal{S}}$ is the maximal $\langle \Omega', \mathcal{W}, \lambda \rangle$-invariant set. □

Furthermore, one can show that if $\lambda' \to \lambda$, the resulting set $\overline{\mathcal{S}}$ also approximates the real maximal $\langle \Omega', \mathcal{W}, \lambda \rangle$-invariant set with increasing accuracy.

The above algorithm can be used for the construction of (approximations to) the MAS for a wide variety of model classes. The only requirement is the modification of the definition of the pre-set according to the given model class.

In the following section a reformulation of the above algorithm is given, specifically stating the operations on the different constraints describing the intermediate sets. This will provide insight into the constraint structure of the obtained sets, which will allow us to reduce the complexity of the obtained sets in Chapter 5.

2.3 Efficient computation of the MAS for LPV systems

First we formulate a basic algorithm for the disturbance-free case without contraction constraints. Later we formulate both additions as extensions of this basic algorithm.

2.3.1 Efficient algorithm formulation

First of all it should be noted that the pre-set $\text{Pre}_{\langle\Omega\rangle}(\mathcal{S})$, with $\mathcal{S} = \{x | A_{\mathcal{S}} x \leq \mathbf{1}\}$, can easily be calculated, by virtue of the following lemma.

Lemma 2.2 (**Pre-set calculation**, [13])**.** *Given an autonomous system* (2.1), (2.2) *with* $\mathcal{W} \equiv \{0\}$ *and a polyhedral set* $\mathcal{S} = \{x | A_{\mathcal{S}} x \leq \mathbf{1}\}$*, then the pre-set* $Pre_{\langle\Omega\rangle}(\mathcal{S})$ *is given by*

$$Pre_{\langle\Omega\rangle}(\mathcal{S}) = \{x | A_{\mathcal{S}}\Phi_1 x \leq \mathbf{1}, \ldots, A_{\mathcal{S}}\Phi_r x \leq \mathbf{1}\}. \tag{2.17}$$

Proof: By making convex combinations $\left(\sum_{i=1}^{r} \lambda_i A_{\mathcal{S}}\Phi_i\right) x \leq \mathbf{1}$, of the different constraints $A_{\mathcal{S}}\Phi_i x \leq \mathbf{1}, i = 1, \ldots, r$, one can see that satisfaction of $A_{\mathcal{S}}\Phi_i x \leq \mathbf{1}, i = 1, \ldots, r$ implies that $A_{\mathcal{S}}\Phi x \leq \mathbf{1}, \forall \Phi \in \Omega'$, which, combined with the definition of the pre-set, proves the lemma. □

On one hand, this lemma allows the straightforward iterative calculation of the sets $\mathcal{O}_i$ in Algorithm 2.3, but also shows the exponential growth of the number of constraints describing these sets for increasing values of i.

However, one can see that the constraints describing two successive sets $\mathcal{O}_i$ and $\mathcal{O}_{i-1}$ partly overlap, since all constraints describing $\mathcal{O}_i$ are by construction present in the description of $\mathcal{O}_{i-1}$. Hence, when calculating the pre-set of $\mathcal{O}_i$, only the non-overlapping part of the constraints of $\mathcal{O}_i$ has to be considered explicitly.

A second important aspect is the elimination of redundant constraints. As already pointed out earlier, if $r > 1$ the number of constraints describing the sets $\mathcal{O}_i$ increases exponentially as a function of i. However, especially for lower-dimensional systems, typically a large fraction of these constraints will be redundant and hence can be omitted. Therefore, significant efficiency gains can be obtained when only retaining those constraints of $\text{Pre}_{\langle\Omega'\rangle}(\lambda'\mathcal{O}_{i-1})$ that are non-redundant with respect to $\mathcal{O}_{i-1}$.

Finally, one should note that by iteratively adding constraints, constraints constructed in earlier iterations can become redundant, even if they were initially non-redundant. Therefore, efficient implementations should also regularly check the redundancy of constraints constructing in earlier iterations, a process we will refer to as *garbage collection*.

Given these considerations, Algorithm 2.3 can now be efficiently reformulated for the case $\mathcal{W} = \{0\}, \lambda = \lambda' = 1$:

Algorithm 2.4 (**Efficient MAS-computation for LPV systems**, [98])**.** *Given an LPV system* (2.1),(2.2) *subject to constraints* (2.10)*.*

1. *Initialize* $A_{\mathcal{S}} := A_x,\ i := 1$.
2. *Perform the following steps iteratively until* $i >$ *rows*$(A_{\mathcal{S}})$*:*
   (a) *Set* $a^{\mathrm{T}} := A_{\mathcal{S}}(i, :)$.
   (b) *Check the redundance of the constraints* $a^{\mathrm{T}}\Phi_i x \leq 1,\ i = 1, \ldots, r$ *with respect to* $\mathcal{S} \triangleq \{x | A_{\mathcal{S}} x \leq \mathbf{1}\}$*. For each* $i = 1, \ldots, r$*, if* $sig_{\mathcal{S}}(a^{\mathrm{T}}\Phi_i) > 1$*, then add the constraint* $a^{\mathrm{T}}\Phi_i x \leq 1$ *to* $A_{\mathcal{S}}$ *by setting* $A_{\mathcal{S}} := [A_{\mathcal{S}}; a^{\mathrm{T}}\Phi_i]$.

(c) If necessary[2]*, perform* garbage collection*, i.e. check for every row of $A_{\mathcal{S}}$ whether the corresponding constraint is redundant with respect to the set defined by the other rows of $A_{\mathcal{S}}$ and if so, remove that row from $A_{\mathcal{S}}$.*

(d) Set $i := i + 1$.

One can easily verify that Algorithm 2.4 is identical to Algorithm 2.3 with $\mathcal{W} = \{0\}, \lambda = \lambda' = 1$, apart from the following aspects:

- Algorithm 2.4 makes no explicit distinction between the different sets $\mathcal{O}_i, i \geq 0$. This is possible due to the fact that every set $\mathcal{O}_i, i \geq 0$ is described by the constraints of $\mathcal{O}_{i-1}$ supplemented with the constraints of $\text{Pre}_{\langle\Omega'\rangle}(\mathcal{O}_{i-1})$.
- The resulting invariant sets are geometrically identical for the two algorithms, but differ in their algebraic representation, since Algorithm 2.4 eliminates all redundant constraints.

Since Algorithms 2.3 and 2.4 are identical from a conceptual point of view, Theorem 2.1 also applies to Algorithm 2.4. Since Algorithm 2.4 always uses $\lambda = \lambda' = 1$, the resulting set is therefore guaranteed to be the MAS for system (2.1), (2.2) subject to constraints (2.10). However, it is not guaranteed that the algorithm terminates after a finite number of iterations. The following Theorem shows under what circumstances Algorithm 2.4 terminates in a finite number of iterations.

Theorem 2.2. *Consider the disturbance-free system* (2.1), (2.2) *subject to constraints* (2.10)*. If the following condition is satisfied*

$$\exists c \in \mathbb{R}^+, \alpha \in (0,1): \quad \|\Phi(k)\Phi(k-1)\dots\Phi(0)\| \leq c\alpha^k, \quad \forall \Phi(0),\dots,\Phi(k) \in \Omega', k \in \mathbb{N}, \quad (2.18)$$

then Algorithm 2.3 (with $\lambda = \lambda' = 1$) and Algorithm 2.4 terminate in a finite number of iterations.

Proof: First of all it should be noted that the sets $\mathcal{O}_i, i \in \mathbb{N}$ constructed in Algorithm 2.3 can be written as

$$\mathcal{O}_i = \bigcap_{j=0}^{i} \mathcal{X}_j, \qquad i \in \mathbb{N}, \quad (2.19)$$

with $\mathcal{X}_0 = \mathcal{X}$ and $\mathcal{X}_{i+1} = \text{Pre}_{\langle\Omega'\rangle}(\mathcal{X}_i), i \in \mathbb{N}$. This can be proven by induction on i. Expression (2.19) is trivially satisfied for $i = 0$. Furthermore, if it is guaranteed for i it is also guaranteed for $i+1$, since $\text{Pre}_{\langle\Omega'\rangle}(\mathcal{O}_i) = \text{Pre}_{\langle\Omega'\rangle}(\bigcap_{j=0}^{i} \mathcal{X}_i) = \bigcap_{j=0}^{i} \text{Pre}_{\langle\Omega'\rangle}(\mathcal{X}_i) = \bigcap_{j=1}^{i+1} \mathcal{X}_i$ and hence $\mathcal{O}_{i+1} = (\bigcap_{j=1}^{i+1} \mathcal{X}_i) \cap (\bigcap_{j=0}^{i} \mathcal{X}_i) = \bigcap_{j=0}^{i+1} \mathcal{X}_i$.

Due to the recursion $\mathcal{X}_{i+1} = \text{Pre}_{\langle\Omega'\rangle}(\mathcal{X}_i)$, the sets $\mathcal{X}_i, i \in \mathbb{N}$ can be written as $\mathcal{X}_i = \{x | A_{\mathcal{X}_i} x \leq \mathbf{1}\}$ with $A_{\mathcal{X}_0} = A_{\text{x}}$ and $A_{\mathcal{X}_{i+1}} = [A_{\mathcal{X}_i}\Phi_0; \dots; A_{\mathcal{X}_i}\Phi_r]$. This allows

[2]The necessity of garbage collection can only be evaluated a posteriori. However, a rule of thumb that experimentally has been proven to be adequate in a wide range of circumstances, is to perform garbage collection with every 50%-increase in rows$(A_{\mathcal{S}})$.

us to construct an upper bound to $\overline{l}(i) \triangleq \max_{j,x\in\mathcal{X}} \|A_{\mathcal{X}_i}(j,:)x\|$ as a function of the iteration number i:

$$\begin{aligned}\overline{l}(i) &\triangleq \max_{j,x\in\mathcal{X}} \|A_{\mathcal{X}_i}(j,:)x\|,\\ &\leq \max_j \|A_{\mathrm{x}}(j,:)\| \max_{\Phi(0\ldots i)\in\Omega'} \|\Phi(i)\Phi(i-1)\ldots\Phi(0)\| \max_{x\in\mathcal{X}} \|x\|,\\ &\leq c\alpha^i \max_j \|A_{\mathrm{x}}(j,:)\| \max_{x\in\mathcal{X}} \|x\|. \end{aligned} \tag{2.20}$$

One can see that if $\overline{l}(i) \leq 1$, which is satisfied if $abc\alpha^i \leq 1$, with $a = \max_j \|A_{\mathrm{x}}(j,:)\|$ and $b = \max_{x\in\mathcal{X}} \|x\|$, it is guaranteed that $\mathcal{X} \subseteq \mathcal{X}_i$. Because $\mathcal{O}_{i-1} \subseteq \mathcal{X}$ and $\mathcal{O}_i = \mathcal{O}_{i-1} \cap \mathcal{X}_i$ it then follows that $\mathcal{O}_{i-1} = \mathcal{O}_i$. Consequently, since $\mathrm{Pre}_{\langle\Omega'\rangle}(\mathcal{O}_{i-1}) \cap \mathcal{O}_{i-1}$ it can be concluded that the termination condition $\mathcal{O}_{i-1} \subseteq \mathrm{Pre}_{\langle\Omega'\rangle}(\mathcal{O}_{i-1})$ used in Algorithm 2.3 is satisfied for these values of i. This shows that Algorithm 2.3 terminates in at most $k^\star_{\max}$ iterations, with $k^\star_{\max}$ defined as

$$k^\star_{\max} = \left\lfloor -\frac{\ln a + \ln b + \ln c}{\ln \alpha} \right\rfloor. \tag{2.21}$$

Since for every iteration executed in Algorithm 2.3, Algorithm 2.4 only has to execute a finite number of iterations (i.e., equal to the number of constraints in the set $\mathcal{X}_i$), also the latter algorithm is proven to terminate in a finite number of iterations. □

This result shows that the MAS of systems satisfying condition (2.18) is finitely determined, meaning it can be described by a finite number of linear inequalities, and hence is a polytope.

Condition (2.18) is less strict than the one proven in [98, 101], which required the given LPV system to be quadratically stable. Quadratic stability of the given system ensures satisfaction of condition (2.18), but the opposite is not guaranteed. It can be shown that (2.18) is satisfied for all systems where the *Joint Spectral Radius* (JSR, [137]) of the matrices $\Phi_1, \ldots, \Phi_r$ is strictly smaller than 1. We refer to Appendix C for more details.

Theorem 2.3. *If for a given uncertainty polytope Ω' the JSR $\hat{\rho}(\Omega') = \lambda < 1$, then condition* (2.18) *is satisfied $\forall\alpha \in (\lambda, 1)$.*

Proof: If $\hat{\rho}(\Omega') = \lambda < \alpha$ then, due to the definition of $\limsup$, there exists a positive integer $n \in \mathbb{Z}^+$ such that $\max_{A(i)\in\Omega', i=1,\ldots,k} \|A(1)\cdot\ldots\cdot A(k)\|^{\frac{1}{k}} < \alpha$, $\forall k \geq n$, or equivalently that $\max_{A(0\ldots k)\in\Omega'} \|A(0)\cdot\ldots\cdot A(k)\| < \alpha^{k+1}$, $\forall k \geq n$. If we now choose a positive scalar $c' \in \mathbb{R}^+$ such that

$$c' > \max\left(1, \max_{\substack{k=0,\ldots,n-1\\ A(i)\in\Omega', i=0,\ldots,k}} \left(\frac{\|A(0)\cdot\ldots\cdot A(k)\|}{\alpha^{k+1}}\right)\right),$$

then it is clear that $\max_{A(i)\in\Omega', i=0,\ldots,k} \|A(0)\cdot\ldots\cdot A(k)\| < c'\alpha^{k+1}$, $\forall k \in \mathbb{N}$. Condition (2.18) is hence satisfied $\forall c \geq c'\alpha$. □

Since $\hat{\rho}(\Omega') < 1$ is equivalent with asymptotic stability of the autonomous LPV system [18] the above theorem extends the applicability of Algorithm 2.4 from quadratically stable systems (as proven in [98]) to all asymptotically stable autonomous LPV systems of the form considered in this thesis. Similar statements can be made if $\max_j \|\Phi_j\| = \lambda < 1$, but the value of λ thus obtained will be equal or larger than that obtained using the JSR [18] and therefore the upper bound $k^\star_{\max}$ will typically be much more conservative. However, the main disadvantage of using the JSR, is that its computation is an NP-hard problem. Even computing arbitrarily close approximations to the JSR is an operation with a higher than polynomial worst-case computational cost if there is no problem structure (e.g., matrix symmetry, non-negativity of matrix elements, ...) that can be exploited [18, 137]. In general it can be shown [20, 71] that $\hat{\rho} \leq 1$ is *algorithmically* undecidable. This seems to imply that using the JSR might not always be useful in this context, but in many cases sufficiently close approximations can be computed [18] in an acceptable amount of time in order to either confirm or rule out that $\hat{\rho}(\cdot) < 1$.

It should be noted that Theorem 2.2 cannot be applied if the constraints $\mathcal{X}$ is not bounded, since in that case b will be infinite and no useful bound will be obtained. However, the following corollary might be useful in that case and can be verified easily:

Corollary 2.1. *Consider the disturbance-free system* (2.1), (2.2) *subject to constraints* (2.10)*. If condition* (2.18) *is satisfied and there exists $j \in \mathbb{Z}^+$ such that $\mathcal{O}_j$ is bounded, then an upper bound to the number of iterations of Algorithm 2.3 is given by*

$$k^\star_{\max} = j + \left\lfloor -\frac{\ln a + \ln b + \ln c}{\ln \alpha} \right\rfloor, \tag{2.22}$$

where a and b are calculated using set $\mathcal{O}_j$ instead of $\mathcal{X}$.

Proof: The above upper bound can easily be verified by applying Theorem 2.2 to the case when Algorithm 2.3 is applied to system (2.1),(2.2) subject to constraint set $\mathcal{O}_j$ instead of $\mathcal{X}$. □

This corollary will prove insightful in Section 5.1, where polyhedral invariant sets for systems with a specific structure are studied.

2.3.2 Structure of the MAS

Considering the results of the previous section, it is now easy to see that the MAS $\overline{\mathcal{S}}$ for a given LPV system has a specific structure since it can be expressed as an intersection of different sets $\overline{\mathcal{S}} = \bigcap_{i=0}^{k^\star} \mathcal{X}_k$ and due to the recursive relationship $\mathcal{X}_{i+1} = \text{Pre}_{\langle\Omega'\rangle}(\mathcal{X}_i)$ between these sets.

The different sets $\mathcal{X}_i$ induce a hierarchical ordering of the constraints describing $\overline{\mathcal{S}}$ into different levels, depending on the value of i. The recursion relationship between successive sets $\mathcal{X}_i$ and $\mathcal{X}_{i+1}$ induces links between the constraints on level i and those on level $i+1$. As a result the structure of the MAS of an LPV system can be depicted as a tree of constraints. The following properties can be verified easily and are illustrated in the next section:

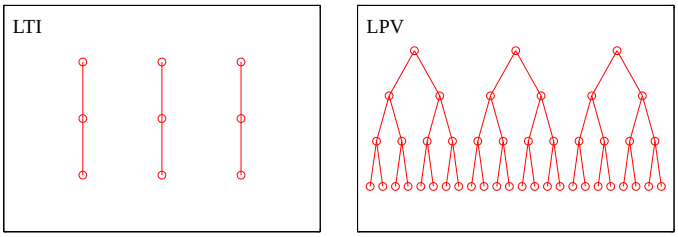


Figure 2.2: Left: Tree structure of the MAS of an LTI system with $m_y = 3, k^\star = 3$. **Right:** Tree structure of the MAS of an LPV system with $m_y = 3, r = 2, k^\star = 3$. All constraints are assumed to be non-redundant.

- The tree structure has a depth of at most $k^\star$ layers.
- Every constraint at level i will have at most r *children* at level $i+1$. Due to redundance of constraints the number of children of a given constraint can be strictly smaller than r.
- Every constraint $a^{\mathrm{T}}x \leq 1$ at an arbitrary level i can be expressed as $a^{\mathrm{T}} = b^{\mathrm{T}}\Phi_j$ with $j \in \{1, \ldots, r\}$ and $b^{\mathrm{T}}x \leq 1$ a constraint at level $i-1$. This can be verified by analyzing the recursion between $\mathcal{X}_i$ and $\mathcal{X}_{i-1}$ and Lemma 2.2.
- The tree structure consists of at most m_x constraints. This is due to the fact that all constraints at an arbitrary level i are constructed using exactly one constraint at level $i-1$ and hence all constraints can be 'traced back' to exactly one constraint at level 0. Due to the initialization $\mathcal{X}_0 = \mathcal{X}$ there are at most m_x constraints at level 0.
- Every constraint in the tree structure has a *parent*, except the constraints at level 0. This is equivalent with saying that if a constraint $a^{\mathrm{T}}x \leq 1$ at an arbitrary level i is redundant with respect to the invariant set, all its children $a^{\mathrm{T}}\Phi_i x \leq 1, i = 1, \ldots, r$ will also be redundant. Indeed, if one constraint $a^{\mathrm{T}}\Phi_{i^\star}x \leq 1, i^\star \in \{1, \ldots, r\}$ is non-redundant, while $a^{\mathrm{T}}x \leq 1$ is redundant, one can find a state vector $x^\star \in \overline{\mathcal{S}}$ such that $a^{\mathrm{T}}\Phi_{i^\star}x^\star = 1$. The vector $\Phi_{i^\star}x$ hence lies exactly on the constraint induced by a^{T} which is redundant. $\Phi_{i^\star}x$ therefore has to lie outside of $\overline{\mathcal{S}}$ which contradicts the invariance property of $\overline{\mathcal{S}}$. Consequently, all constraints $a^{\mathrm{T}}\Phi_i x \leq 1, i = 1, \ldots, r$ have to be redundant.

A schematic depiction of a possible tree structure as described above, is shown in Figure 2.2 and is compared to the structure present in a polyhedral set for an LTI system (i.e., $r = 1$). One can see that in the worst-case the number of constraints describing the set can increase exponentially as a function of $k^\star$, while this increase is only linear in the LTI case. Given values for $m_x, r, k^\star$, the following expression gives the worst-case number of constraints

$$\text{rows}(A_{\overline{\mathcal{S}}}) = m_x \sum_{i=0}^{k^\star} r^i = m_x \frac{1 - r^{k^\star+1}}{1 - r}. \tag{2.23}$$

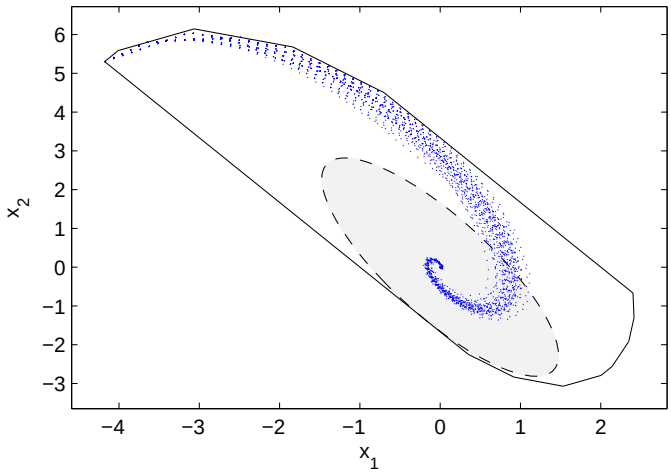


Figure 2.3: The maximal admissible set computed using Algorithm 2.4 (solid) and an ellipsoidal invariant set computed using Algorithm 2.2 (dashed and shaded) for system (2.24), controlled by controller $u(k) = [-0.5 \ -0.3]x(k)$ and subject to constraints (2.25). 50 trajectories starting from the leftmost vertex of the MAS are depicted in dotted lines.

In most cases the real number of constraints will typically be significantly lower, an example of which is given in the next section.

2.3.3 Example

In this section a numerical example is given in order to illustrate the efficacy of Algorithm 2.4.

We consider a system with $n_x = 2, n_u = 1, r = 2$, described by matrices

$$A_1 = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix}, \qquad A_2 = \begin{bmatrix} 1 & 0.2 \\ 0 & 1 \end{bmatrix}, \tag{2.24a}$$

$$B_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \qquad B_2 = \begin{bmatrix} 0 \\ 1.5 \end{bmatrix}, \tag{2.24b}$$

subject to constraints (for $k = 0, \ldots, \infty$)

$$-1 \leq u(k) \leq 0.5, \tag{2.25a}$$

$$\begin{bmatrix} -10 \\ -10 \end{bmatrix} \leq x(k) \leq \begin{bmatrix} 10 \\ 10 \end{bmatrix}. \tag{2.25b}$$

A linear feedback controller $u(k) = -Kx(k)$ is chosen as $K = [0.5 \ 0.3]$. This results in an autonomous LPV system with $\Omega' = \mathrm{Co}\{A_1 - B_1K, A_2 - B_2K\}$ subject to state constraints defined by $A_x = [-2K; K; 0.1I; -0, 1I]$. The closed-loop system is asymptotically stable and satisfies condition (2.18).

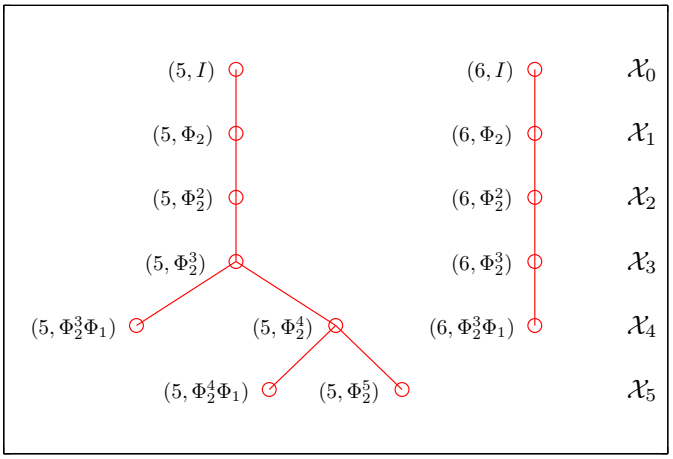


Figure 2.4: Tree structure of the polyhedral invariant set depicted in Figure 2.3. Shorthand notation (i, C) is used to denote constraints $A_x(i,:)Cx \leq 1$. At the right hand-side the sets $\mathcal{X}_i$ are indicated to which the constraints of that level of the tree belong.

Figure 2.3 shows the MAS obtained using Algorithm 2.4 and compares it to an ellipsoidal invariant set computed using Algorithm 2.2. The polyhedral invariant set clearly is significantly larger than the ellipsoidal set and is able to take the asymmetric constraints into account. The ellipsoidal invariant set is not able to efficiently deal with the asymmetry in the imposed constraints since by construction it is centered around the origin. The plotted trajectories show that both sets are indeed invariant with respect to the uncertain dynamics.

Figure 2.4 depicts the tree structure corresponding to the polyhedral invariant set depicted in Figure 2.3. The structure consists of two trees, corresponding to imposed constraints 5 and 6. Since in this example the 4 first constraints correspond to the state constraints, one can see that these constraints are all redundant and only the input constraints determine the size and shape of the MAS in this case.

One can also see that most constraints only have 1 *child*, whereas the theoretical maximum is $r = 2$. These observations indicate that a massive reduction of the number of constraints is obtained by removing the redundant ones. The theoretical maximum number of constraints in this case is $m_x(r^{(k^\star+1)} - 1) = 6(2^6 - 1) = 378$. Only 11 of those constraints are found to be non-redundant. It should be noted that during the execution of Algorithm 2.4 the number of rows in matrix $A_{\mathcal{S}}$ never exceeded 15. Total computation time on a 1.6GHz Pentium-M CPU was 7.4 seconds. MATLAB 6.5 and the standard toolboxes were used.

More detailed examples and scalability analyses will be shown in Chapters 5 and 8.

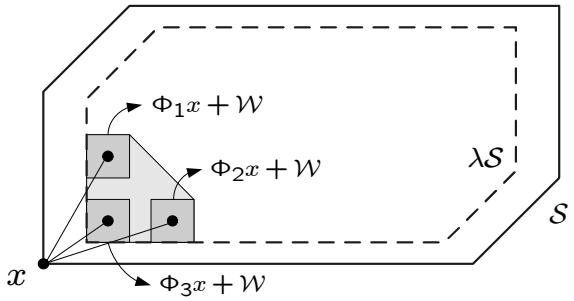


Figure 2.5: Schematic representation of condition (2.29) for a $\langle \Omega', \mathcal{W}, \lambda \rangle$-invariant set $\mathcal{S}$ with $n_x = 2, r = 3, l = 4$ and $\lambda < 1$.

2.4 Extensions

In this section we discuss two basic extensions to the basic formulation of Algorithm 2.4, namely the extensions towards $\mathcal{W} \neq \{0\}$ and $\lambda < 1$. The basic structure of the algorithm does not change, but the way the pre-set is calculated has to be updated accordingly. Also the conditions under which convergence of the algorithm is guaranteed, will have to be updated.

2.4.1 Contraction constraints

Often it is desired to construct invariant sets with a given imposed contraction rate, i.e. $\langle \Omega', \{0\}, \lambda \rangle$-invariant sets, with $\lambda \in (0, 1)$. This can then e.g. be used to obtain constrained controllers with a certified rate of convergence towards the origin. Given a set $\mathcal{S} = \{x | A_{\mathcal{S}} x \leq \mathbf{1}\}$, the pre-set can now be calculated as

$$\text{Pre}_{\langle \Omega, \{0\}, \lambda \rangle}(\mathcal{S}) = \{x | A_{\mathcal{S}} \Phi_1 x \leq \lambda \mathbf{1}, \ldots, A_{\mathcal{S}} \Phi_r x \leq \lambda \mathbf{1}\}, \tag{2.26}$$

or if the standard formulation with right hand-side $\mathbf{1}$ should be maintained:

$$\text{Pre}_{\langle \Omega, \{0\}, \lambda \rangle}(\mathcal{S}) = \{x | \lambda^{-1} A_{\mathcal{S}} \Phi_1 x \leq \mathbf{1}, \ldots, \lambda^{-1} A_{\mathcal{S}} \Phi_r x \leq \mathbf{1}\}. \tag{2.27}$$

This way it can be seen that computing an $\langle \Omega', \{0\}, \lambda \rangle$-invariant set actually corresponds to computing an $\langle \Omega'', \{0\}, 1 \rangle$-invariant set with Ω'' defined as $\Omega'' \triangleq \text{Co}\{\lambda^{-1}\Phi_1, \ldots, \lambda^{-1}\Phi_r\}$. Condition (2.18) now becomes

$$\begin{aligned} &\exists c \in \mathbb{R}^+, \alpha \in (0, \lambda) : \\ &\|\Phi(k)\Phi(k-1)\ldots\Phi(0)\| \leq c\alpha^k, \quad \forall \Phi(0), \ldots, \Phi(k) \in \Omega', k \in \mathbb{N}, x \in \mathcal{X}, \end{aligned} \tag{2.28}$$

which is guaranteed to be satisfied if $\max_j \|\Phi_j\| < \lambda$ or more generally speaking if $\hat{\rho}(\Omega') < \lambda$. Again, the latter condition is less conservative, but is more difficult to check, since computing the joint spectral radius is an NP-hard problem.

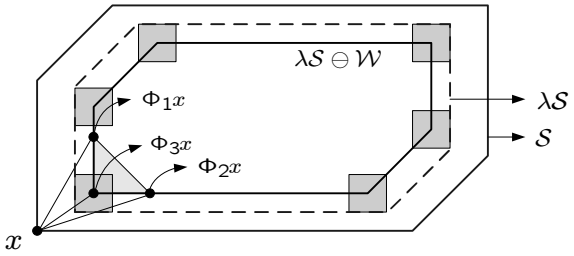


Figure 2.6: Schematic representation of condition (2.30) for a $\langle \Omega', \mathcal{W}, \lambda \rangle$-invariant set $\mathcal{S}$ with $n_x = 2, r = 3, l = 4$ and $\lambda < 1$.

2.4.2 Bounded disturbances

Computing $\langle \Omega', \mathcal{W}, \lambda \rangle$-invariant sets, with $\lambda \in (0, 1)$, can be done by making similar adjustments to the calculation of the pre-set. The pre-set of $\mathcal{S}$ consists of all states x for which the following condition is satisfied:

$$\Phi x + w \in \lambda\mathcal{S}, \qquad \forall \Phi \in \Omega', \forall w \in \mathcal{W}, \tag{2.29}$$

which, according to Definition (1.13) and due to the fact that Ω' is a polytope, is equivalent with

$$\Phi_i x \in (\lambda\mathcal{S} \ominus \mathcal{W}), \qquad \forall i = 1, \ldots, r. \tag{2.30}$$

Figure 2.5 schematically depicts condition (2.29) for a $\langle \Omega', \mathcal{W}, \lambda \rangle$-invariant set $\mathcal{S}$ with $n_x = 2, r = 3, l = 4$ and $\lambda < 1$. Figure 2.6 shows condition (2.30) for the same situation.

Using the remarks in Chapter 1 regarding the computation of the Minkowski difference of a H-polytope and a V-polytope, the pre-set can now be calculated as

$$\text{Pre}_{\langle \Omega, \{0\}, \lambda \rangle}(\mathcal{S}) = \left\{ x \,\middle|\, \frac{1}{\lambda} A_{\mathcal{S}} \Phi_i x \leq \mathbf{1} - \max_{j=1,\ldots,l} A_{\mathcal{S}} w_j,\ i = 1, \ldots, r \right\}, \tag{2.31}$$

where the maximum of $A_{\mathcal{S}} w_j$ is calculated for each component separately. Convergence of the accordingly modified Algorithm 2.4 is now guaranteed if

$$\frac{\max_j \|\Phi_j\|}{1 - \max_j \|A_{\mathcal{S}}(j,:)\| \max_j \|w_j\|} < \lambda. \tag{2.32}$$

This expression can be very conservative and is not independent of linear state transformations $x' = Mx, M \in \mathbb{R}^{n_x \times n_x}$. However, a similar expression using the JSR, which does not suffer from these drawbacks, is not possible due to the bounded disturbances.

2.5 Conclusions

This section discussed the concept of set invariance, and more precisely the construction of polyhedral invariant sets for LPV systems. Relevant definitions and properties

are given, as well as an overview of the state-of-the-art regarding algorithms for constructing invariant sets.

The main contribution of this chapter is twofold. First of al a more detailed algorithm for the construction of the MAS for LPV systems is given, illustrating the ability to achieve significant computational improvements by only retaining redundant constraints during the computations. The new algorithm also gives insight in the structure present in the constraints describing the MAS. This structure will be exploited in Chapter 5 in order to obtain reduced-complexity polyhedral invariant sets.

Secondly, some attention is given to convergence properties of the described algorithm using the Joint Spectral Radius. The obtained insights in the convergence behavior will also prove useful in Chapter 5 in order to assess the convergence behavior of the algorithms presented there.

The ability to construct polyhedral invariant sets will be used extensively in Chapter 3 in order to improve several existing robust MPC algorithms.

Chapter 3

Robust Model Based Predictive Control

"In these matters the only certainty is that nothing is certain."

– Caius Plinius Secundus (23-79) –

This chapter gives an overview on how to extend the stabilizing MPC framework of the previous chapter towards the robust control setting. A more general model class than the one used in the previous chapter is described, allowing the inclusion of uncertain dynamics and bounded disturbance input in the model description. This chapter describes the modifications to the quasi-infinite horizon MPC algorithm needed for maintaining recursive feasibility and stability in this more general setting. Finally, two existing robust MPC algorithms are described that are incorrectly claimed to be recursively feasible and asymptotically stabilizing in literature. Counterexamples to these claims are provided and corrections are proposed, illustrating the need for careful consideration when designing robust MPC algorithms.

3.1 Introduction

The main aim of this chapter is to relax the assumption of the previously presented quasi-infinite horizon MPC algorithm and its accompanying stability framework that there is no mismatch between the predictions made by the MPC algorithm and the real plant behavior. In real applications it is unavoidable to experience such mismatches for two important reasons.

First of all the prediction model used in the MPC controller – i.e. constraint (1.11e) – typically only approximately describes the plant's real dynamic behavior. There are

multiple causes that contribute to this fact. In most situations prediction models used in MPC are *black box* models, which means they are based on input/output measurements of the real plant. These measurements are noisy and limited in number, which in turn limits the accuracy of the resulting model. Even if *white box* models are used, it is often difficult to exactly determine all involved physical constants. Furthermore, due to wear, maintenance, ... the behavior of a real plant can be expected to change as a function of time which also compromises the accuracy of the prediction model.

Secondly, apart form limited accuracy of the prediction model, often only a limited number of inputs of a system can be manipulated by the controller. In reality a multitude of inputs exist, called *disturbance inputs*, that cannot be manipulated by the controller, because they are governed by phenomena that lie beyond the scope of the plant. These phenomena include dynamics occurring in connected systems that are controlled by other controllers, natural phenomena like rain, outside temperature, ... or other uncontrollable effects like fluctuations in the composition of certain reagents.

Two different approaches to robustness can be identified in the MPC literature. One aspect is the inherent robustness of *nominal* MPC algorithms, i.e. algorithms that were not designed for robustness from the outset. Examples of this approach are [43, 81], but constrained MPC, the main focus of this thesis, is not considered.

A second robustness aspect that has received attention in MPC literature is based on the observation that, although the quantification of the exact contribution of these phenomena at every time instant is impossible due to their uncertain nature, it often *is* possible to quantify the worst-case magnitude of these phenomena, which allows them to be taken into account during the controller design. This is typically done by including this uncertainty information in the model upon which the controller is based, which is referred to as robust control (see e.g., [145]). This is also the approach taken in this thesis.

3.2 Model class

3.2.1 Uncertain dynamics

Before being able to start the robust controller design process, some information is needed about the model uncertainty. Early results [2, 44, 50, 144] considered impulse response models with bounded uncertainty on the coefficients. However, for reasons of computational efficiency, flexibility and compactness of representation, recent MPC algorithms typically makes use of state space models. The two most well-known uncertainty classes that are able to represent uncertain dynamics in the state-space framework are those of *norm-bounded uncertainty* (or *structured uncertainty*, [91]) and *polytopic uncertainty* [68]. For use in MPC it is computationally convenient to have explicit expressions for the extremal values of the uncertain dynamics, so one typically uses linear state space models with polytopic uncertainty, where the uncertainty region is described as a V-polytope ([146], see Figure 1.5):

$$x(k+1) = A(k)x(k) + B(k)u(k), \qquad k \in \mathbb{N}, \tag{3.1}$$

with

$$[A(k)\ B(k)] \in \Omega \equiv \mathrm{Co}\{[A_1\ B_1], \ldots, [A_r\ B_r]\}, \qquad k \in \mathbb{N}, \tag{3.2}$$

or equivalently

$$[A(k)\ B(k)] \in \left\{ \sum_{j=1}^{r} \lambda_j(k)[A_j\ B_j] \,\middle|\, \lambda_1(k) \geq 0, \ldots, \lambda_r(k) \geq 0, \sum_{j=1}^{r} \lambda_j(k) = 1 \right\}, \quad k \in \mathbb{N}. \tag{3.3}$$

In further sections we will use the shorthand notation $\lambda(k) = [\lambda_1(k); \ldots; \lambda_r(k)]$. Depending on the allowed variation in time of $\lambda(k)$ and the availability of information regarding this variation a distinction can be made between the following model classes:

Linear Time-Invariant (LTI):	$\lambda(k)$ is constant (i.e. independent of time k) and known a priori.
Linear Time-Varying (LTV):	$\lambda(k)$ is time-dependent and known a priori.
Linear Parameter-Varying (LPV):	$\lambda(k)$ is time-dependent but unknown a priori.

Note that in MPC literature the term LTV is sometimes used to refer to LPV systems. Most existing MPC algorithms are designed for the latter model class and hence also in this thesis only LPV systems are considered.

Also note that from a control point of view LPV systems are more general than LTV or LTI systems in the sense that beyond the bounding plytope Ω no further information is known about $\lambda(k)$ and hence the controller has to be designed for the worst-case variations. A controller that is designed for an LPV system with uncertainty region Ω will therefore also stabilize any LTV system with the same uncertainty region or any LTI system with dynamics inside Ω.

Strictly speaking, the set Ω only represents an *uncertainty* in the system matrices in the LPV case, while in the other cases the set Ω simply represents bounds on the (known) value(s) of $[A(k)\ B(k)]$. However, in this thesis we will always build controllers for the LPV case and therefore, even if the real system is LTI or LTV, the real values of $[A(k)\ B(k)]$ are not known to the controller. Therefore, with some slight abuse of terminology, the set Ω is always referred to as the *uncertainty polytope* in the rest of this thesis.

Finally, it should also be noted that we do not consider bounds on the speed of variation of the variables $\lambda(k)$, which is typically the case in *gain scheduling* [76, 128, 134] and which is often implied when using the term LPV. The robust controllers discusses in this thesis allow for arbitrarily fast changes of $\lambda(k)$ (within Ω) and therefore should also work if the real values of $\lambda(k)$ have a limited rate of change. Admittedly, taking into account such bounds on the rate of change, if they

are known, could improve control performance significantly. An examples of such an approach can be found in e.g., [134].

3.2.2 Disturbance inputs

A second source for mismatches between the predicted and the actual plant behavior, is the existence of inputs that cannot be manipulated by the controller and are typically impossible to measure. In the MPC framework these disturbances are typically modeled as bounded state disturbances denoted as $w(k)$:

$$x(k+1) = A(k)x(k) + B(k)u(k) + w(k), \qquad k \in \mathbb{N}, \tag{3.4}$$

with $[A(k)\ B(k)]$ satisfying (3.2) and $w(k)$ satisfying (2.4). $\mathcal{W}$ is typically represented as a V-polytope, since for computational reasons it is favorable to have an explicit expression for the extremal values of $w(k)$. However, an H-polytope representation for $\mathcal{W}$ would only mildly complicate the way disturbances can be taken into account. Therefore, in further chapters, MPC algorithms that take disturbances into account in general can be extended to this setting with only relatively small implementational modifications.

To make the distinction between system (3.1)-(3.2) and system (2.4),(3.2),(3.4) in future sections the former will be referred to as the *disturbance-free system*, while the latter will be referred to as the *disturbed system*. When not explicitly mentioned, the disturbance-free system (3.1)-(3.2) is being referred to. System (1.1) will be referred to as the *nominal system*.

3.3 Constraint handling in robust MPC

The aim of this section is not to give an exhaustive overview of the state-of-the-art of robust MPC. An overview of related algorithms and necessary background will be given when new algorithms will be introduced in further sections. The aim of this section is rather to give a more fundamental idea of how robustness can be incorporated into the MPC design and how it influences recursive feasibility. Two important methodologies are explained, after which a critical assessment is given about the advantages of each of both methods. The main idea behind both methods is the optimization of the *worst case* (over all possible values of the uncertainty) control cost [2, 27]. Special attention will be given to feasibility issues, since this is an important area in which both methods differ.

In order to improve clarity we only discuss disturbance-free LPV systems in further sections of this chapter, although some of the algorithms described here were initially proposed for LTI systems subject to bounded disturbances. However, conceptually there are no significant differences between both settings with regards to recursive feasibility. Therefore the conclusions of this chapter apply to robustness with respect to polytopic model uncertainty, polytopically bounded disturbances and the combination of both. The next two sections describe results published in [144] and [129], which still lie at the basis of many recently published robust MPC algorithms, either directly or indirectly.

3.3.1 Open-loop min-max MPC

One of the first well-known stability results for robust MPC incorporating worst-case predictions is the paper by Zheng and Morari published in 1993 [144]. A modification to the standard finite-horizon MPC Algorithm 1.1 is proposed that allows uncertainty in the future output/state evolutions to be taken into account for a class of uncertain FIR-models. However, the absence of output/state constraints and the rather restrictive class of models considered there, did not necessitate the incorporation of specific stability measures as in Algorithm 1.2. Therefore the results only have limited general applicability. In what follows we briefly describe the method of [144] applied to model class (3.1) and objective function (1.3).

3.3.1.1 Open-loop predictions

In Algorithm 1.1, at every time instant an optimal input sequence $\mathbf{u}^{\mathrm{o}}(k)$ is optimized by solving a finite-horizon optimal control problem. Within the optimization problem the corresponding within-horizon state sequence $\mathbf{x}^{\mathrm{o}}(k)$ is computed by means of model equations (1.1) and both input and state constraints are applied. However, when using model (3.1), this state sequence cannot be computed deterministically. The method described in [144] tackles this by computing set-valued state predictions $\mathcal{X}_{\mathbf{u}_N}(k+i|k)$ corresponding to input sequence $\mathbf{u}_N$ (see (1.5)):

$$\mathcal{X}_{\mathbf{u}_N(k)}(k+i+1|k) \triangleq \{Ax+Bu(k+i|k)|[A\ B] \in \Omega, x \in \mathcal{X}_{\mathbf{u}_N(k)}(k+i|k)\}, \quad i=0,\ldots,N-1, \quad (3.5)$$

with $\mathcal{X}_{\mathbf{u}_N(k)}(k|k) \equiv \{x(k|k)\}$. However, due to the fact that Ω is a V-polytope and due to linearity of the prediction equations the prediction sets can also be represented as V-polytopes:

$$\mathcal{X}_{\mathbf{u}_N(k)}(k+i|k) \triangleq \mathrm{Co}\{x_{j_0,\ldots,j_{i-1}}(k+i|k)|j_m=1,\ldots,r, m=0,\ldots,i-1\}, \quad i=1,\ldots,N, \quad (3.6a)$$

with

$$x_{j_0}(k+1|k) = A_{j_0}x(k|k)+B_{j_0}u(k|k), \qquad j_0=1,\ldots,r, \quad (3.6b)$$

$$x_{j_0,\ldots,j_{i-1},j_i}(k+i+1|k) = A_{j_i}x_{j_0,\ldots,j_{i-1}}(k+i|k)+B_{j_i}u(k+i|k), \quad \begin{cases} i=1,\ldots,N-1, \\ j_m=1,\ldots,r, \\ m=0,\ldots,i. \end{cases} \quad (3.6c)$$

This prediction methodology is depicted in Figure 3.1. It is clear that for any $[A(k+i)\ B(k+i)] \in \Omega, i=0,\ldots,j-1, j=1,\ldots,N$ it is guaranteed that $x(k+j|k) \in \mathcal{X}_{\mathbf{u}_N(k)}(k+j|k)$ and that hence the sets $\mathcal{X}_{\mathbf{u}_N(k)}(k+j|k)$ capture all possible state evolutions within the prediction horizon. Due to the fact that a deterministic sequence of inputs is considered, ignoring the fact that at time $k+1$ a different input can be applied depending on the measured value of $x(k+1|k)$, the sets $\mathcal{X}_{\mathbf{u}_N(k)}(k+1|k)$ defined in (3.6) are called *open-loop predictions*. Other examples of this approach can be found in e.g., [31, 33, 84, 110, 142, 143].

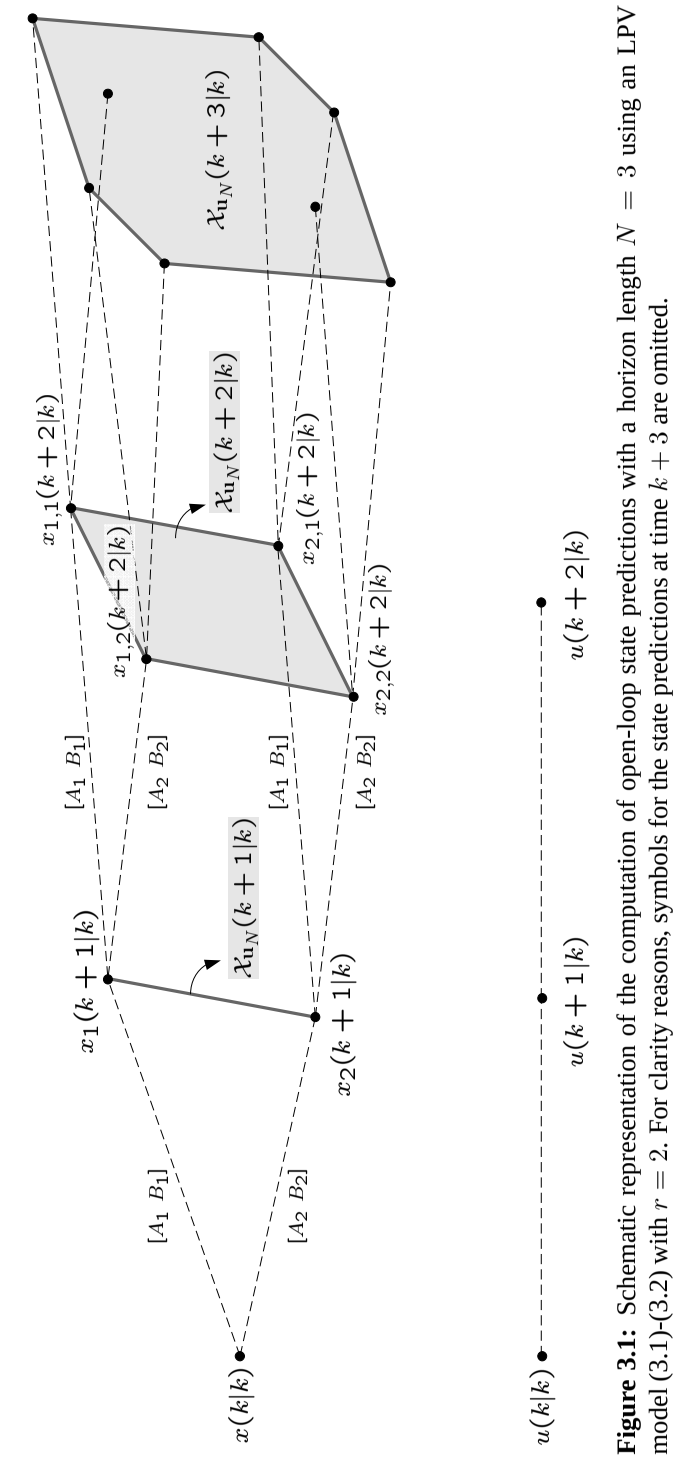


Figure 3.1: Schematic representation of the computation of open-loop state predictions with a horizon length $N = 3$ using an LPV model (3.1)-(3.2) with $r = 2$. For clarity reasons, symbols for the state predictions at time $k + 3$ are omitted.

3.3.1.2 Algorithm formulation

We can now formulate the *robust open-loop min-max MPC algorithm* by imposing state constraints to all vertices of the state prediction sets $\mathcal{X}_{\mathbf{u}_N(k)}(k+i|k), i = 1, \ldots, N$ and by minimizing the worst-case cost by means of a min-max optimization problem:

Algorithm 3.1 (Robust open-loop min-max MPC). *Given a system described by* (3.1)-(3.2)*, subject to constraints* (1.2a)-(1.2b) *and given a control objective* (1.3)*, solve at each time instant k, given the value of the current state $x(k) \equiv x(k|k)$, the following optimization problem:*

$$\min_{\mathbf{u}(k)} \max_{\substack{j_m=1,\ldots,r \\ m=0,\ldots,N-1}} \left(\sum_{i=0}^{N} \|x_{j_0,\ldots,j_{i-1}}(k+i|k)\|_Q^2 + \sum_{i=1}^{N} \|u(k+i|k)\|_R^2 \right), \quad (3.7a)$$

$$\text{s.t.} \quad x_{j_0,\ldots,j_{i-1}}(k+i|k) \in \mathcal{X}, \quad \begin{cases} j_m = 1, \ldots, r, \\ m = 0, \ldots, i-1, \\ i = 1, \ldots, N, \end{cases} \quad (3.7b)$$

$$u(k+i|k) \in \mathcal{U}, \qquad i = 0, \ldots, N-1, \quad (3.7c)$$

and subject to (3.6b)-(3.6c)*. Apply the input $u(k) \equiv u(k|k)$ to the plant. Repeat this procedure at the next time step $k+1$ based on updated state information.*

Due to convexity of the state constraint set $\mathcal{X}$ and the fact that the state prediction sets $\mathcal{X}(k+i|k)$ are polytopic, it is clear that imposing the state constraints to the vertices of $\mathcal{X}_{\mathbf{u}_N(k)}(k+i|k)$, guarantees that $\mathcal{X}_{\mathbf{u}_N(k)}(k+i|k) \subseteq \mathcal{X}$.

It should be noted that the original algorithm presented in [144] used FIR-models with uncertain coefficients and an L_∞- and L_1-based cost expressed in terms of $y(k)$ and $\Delta u(k) \triangleq u(k) - u(k-1)$. However, the idea of making open-loop worst-case predictions as explained above is conceptually identical.

If $\mathcal{X}, \mathcal{X}_N$ and $\mathcal{U}$ are polyhedral the above optimization problem can be convexified by reformulation as an SOCP. Due to the choice of a different cost objective and a more restrictive model class, the original algorithm can be solved by means of an LP whose dimensions increase polynomially in terms of the control problem size, whereas the above algorithm would require an exponentially increasing number of optimization variables.

3.3.1.3 Recursive feasibility and stability

Due to the fact that only input constraints are considered in [144], recursive feasibility is not an issue, since one can always choose the trivial input sequence $u(k+i|k) \equiv 0, i = 0, \ldots, N-1$ or $\Delta u(k+i|k) \equiv 0, i = 0, \ldots, N-1$. Therefore, no terminal constraint is considered. Furthermore, due to the choice of FIR-models, which are stable by construction and have finite settling-behavior, no specific measures need to be included for asymptotic stability.

However, when extended to a more general setting, like Algorithm 3.1, where state constraints are also included, recursive feasibility *does* become an issue, as is also

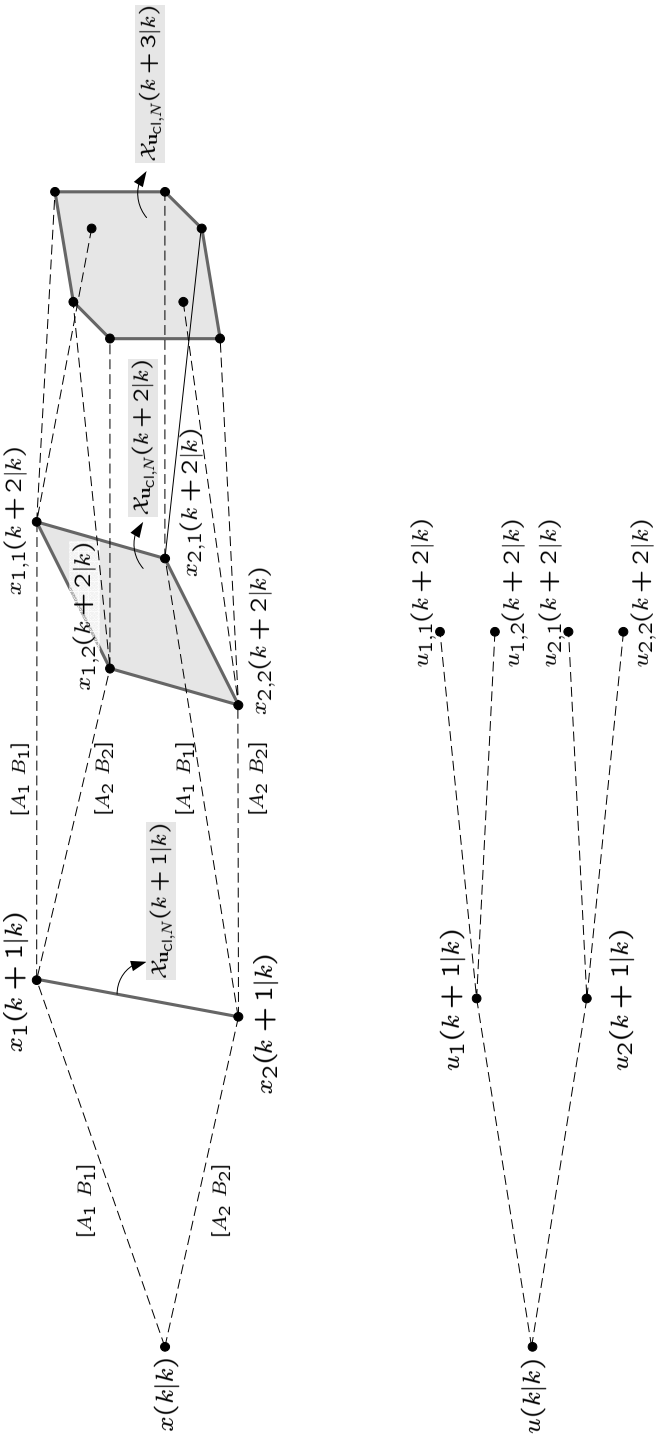


Figure 3.2: Schematic representation of the computation of closed-loop state predictions with a horizon length $N = 3$ using an LPV model (3.1)-(3.2) with $r = 2$. For clarity reasons, symbols for the state predictions at time $k + 3$ are omitted.

pointed out in [144, Remark 2]. Also, when applied to the more general model class (3.1)-(3.2), stability is not guaranteed without including an appropriate terminal cost.

3.3.2 Closed-loop min-max MPC

In order to cope with the recursive feasibility issues found in the min-max MPC using open-loop predictions, [129] introduced the notion of *within-horizon feedback* leading to *closed-loop* predictions. This paradigm explicitly makes of the fact that at time $k+1$ additional information will be available in the form of new state measurements that can then be used to adjust the control strategy.

3.3.2.1 Closed-loop predictions

Instead of optimizing a deterministic sequence of control actions (inputs), the closed-loop MPC paradigm optimizes over a strategy of control actions that is ordered in a tree structure similar to the state predictions. While the open-loop paradigm employs a single input $u(k+i|k)$ at every time instant within the horizon, the closed-loop paradigm employs different inputs $u_{j_0,\ldots,j_{i-1}}$ for every state prediction $x_{j_0,\ldots,j_{i-1}}$ within the horizon. In this way a different control strategy is proposed for every possible realization of the dynamic uncertainty $[A(k+i)\ B(k+i)], i=0,\ldots,N-1$. As such the inputs over which the optimization takes place are dependent of the actual state evolution within the horizon, which corresponds to the notion of feedback. This concept is depicted schematically in Figure 3.2.

Mathematically, the new input sequence parametrization is denoted as $\mathbf{u}_{\mathrm{cl},N}(k) = [\mathbf{u}_{\mathrm{cl}}(k|k);\ldots;\mathbf{u}_{\mathrm{cl}}(k+N-1|k)]$, with

$$\mathbf{u}_{\mathrm{cl}}(k+i|k) \triangleq \begin{bmatrix} u_{1,\ldots,1}(k+i|k) \\ u_{1,\ldots,2}(k+i|k) \\ \vdots \\ \underbrace{u_{r,\ldots,r}}_{i}(k+i|k) \end{bmatrix}, \qquad i=0,\ldots,N-1. \tag{3.8}$$

$\mathbf{u}_{\mathrm{cl},N}(k)$ is referred to as a *closed-loop input sequence*. The corresponding *closed-loop state prediction sets* now become:

$$\mathcal{X}_{\mathbf{u}_{\mathrm{cl},N}(k)}(k+i|k) \triangleq \mathrm{Co}\{x_{j_0,\ldots,j_{i-1}}(k+i|k) | j_m = 1,\ldots,r, m=0,\ldots,i-1\}, \quad i=1,\ldots,N, \tag{3.9a}$$

with

$$x_{j_0}(k+1|k) = A_{j_0}x(k|k) + B_{j_0}u(k|k), \qquad j_0=1,\ldots,r, \tag{3.9b}$$

$$x_{j_0,\ldots,j_{i-1},j_i}(k+i+1|k) = A_{j_i}x_{j_0,\ldots,j_{i-1}}(k+i|k) + B_{j_i}u_{j_0,\ldots,j_{i-1}}(k+i|k), \quad \begin{cases} i=1,\ldots,N-1, \\ j_m=1,\ldots,r, \\ m=0,\ldots,i. \end{cases} \tag{3.9c}$$

Apart from being a solution to ensure recursive feasibility, the additional degrees of freedom resulting from the within-horizon feedback also enables the controller to reduce the spread in the state predictions. As a result the feasible region can be significantly larger compared to open-loop min-max MPC. The same principle is used in e.g., [68, 104, 129, 142].

3.3.2.2 Algorithm formulation

The algorithm described in [129] was initially formulated for LTI systems subject to bounded disturbances and was formulated for a generic class of cost objectives. Here we give the equivalent formulation for disturbance-free LPV systems (3.1)-(3.2) and quadratic control objectives (1.3):

Algorithm 3.2 (**Robust closed-loop min-max MPC).** *Given a system described by* (3.1)-(3.2)*, subject to constraints* (1.2a)-(1.2b) *and given a control objective* (1.3)*, solve at each time instant k, given the value of the current state $x(k) \equiv x(k|k)$, the following optimization problem:*

$$\min_{\mathbf{u}_{\mathrm{cl},N}(k)} \max_{j_{0\dots N-1}=1,\dots,r} \left(\sum_{i=0}^{N-1} \|x_{j_0,\dots,j_{i-1}}(k+i|k)\|_Q^2 + \|x_{j_0,\dots,j_{N-1}}(k+N|k)\|_{Q_N}^2 + \sum_{i=1}^{N} \|u_{j_0,\dots,j_i}(k+i|k)\|_R^2 \right), \quad (3.10a)$$

$$\text{s.t.} \quad x_{j_0,\dots,j_{i-1}}(k+i|k) \in \mathcal{X}, \qquad \begin{cases} j_m = 1,\dots,r, \\ m = 0,\dots,i-1, \\ i = 1,\dots,N-1, \end{cases} \quad (3.10b)$$

$$x_{j_0,\dots,j_{N-1}}(k+N|k) \in \mathcal{X}_N, \qquad \begin{cases} j_m = 1,\dots,r, \\ m = 0,\dots,N-1, \end{cases} \quad (3.10c)$$

$$u_{j_0,\dots,j_i}(k+i|k) \in \mathcal{U}, \qquad \begin{cases} j_m = 1,\dots,r, \\ m = 0,\dots,i, \\ i = 0,\dots,N-1, \end{cases} \quad (3.10d)$$

and subject to (3.9b)-(3.9c)*. Apply the input $u(k) \equiv u(k|k)$ to the plant. Repeat this procedure at the next time step $k+1$ based on updated state information.*

It is clear that compared to Algorithm 3.1, the above algorithm is computationally even more demanding. However, if $\mathcal{X}, \mathcal{X}_N$ and $\mathcal{U}$ are polyhedral it still can be converted to an SOCP and therefore for small values of N it is still practically implementable.

3.3.2.3 Recursive feasibility and stability

One can show that recursive feasibility and asymptotic stability of Algorithm 3.2 is guaranteed if $\mathcal{X}_N$ is convex and there exists a feedback control law $u(k) = -Kx(k)$

such that the following modified stability conditions hold:

$$\begin{aligned}
&1.\quad \mathcal{X}_N \subseteq \mathcal{X}, && && \text{(3.11a)}\\
&2.\quad -Kx \in \mathcal{U}, && \forall x \in \mathcal{X}_N, && \text{(3.11b)}\\
&3.\quad (A-BK)x \in \mathcal{X}_N, && \forall x \in \mathcal{X}_N, \forall [A\ B] \in \Omega, && \text{(3.11c)}\\
&4.\quad Q_N - (A-BK)^{\mathrm{T}} Q_N (A-BK) \succeq Q + K^{\mathrm{T}} R K, && \forall [A\ B] \in \Omega. && \text{(3.11d)}
\end{aligned}$$

Condition (3.11c) means that $\mathcal{X}_N$ should be chosen such that it is robustly positive invariant with respect to the closed loop system formed by the disturbance-free LPV system and the terminal controller. More details on how to construct such sets will be given in Chapter 2.

Condition (3.11d) implies that the terminal cost Q_N should be an upper bound to the worst-case cost beyond the horizon if the terminal controller would be applied. Q_N can be calculated by solving an SDP imposing (3.11d) for all vertices of Ω. More details can be found in Chapter 4, where similar optimization problems are discussed.

Lemma 3.1 (Robust semi-global asymptotic stability). *The closed-loop system formed by system* (3.1)-(3.2) *and the MPC controller defined in Algorithm 3.2 leads to recursive feasibility of the controller and asymptotic stability of the closed-loop system for all states $x(0)$ that lead to a feasible optimization problem* (3.7)*, if $\mathcal{X}_N$ is convex and a controller $u(k) = -Kx(k)$ exists such that conditions* (3.11) *are satisfied.*

Proof. We only give the proof of recursive feasibility. Assume an optimal solution exists at time k:

$$u^{\mathrm{o}}(k|k), u^{\mathrm{o}}_{j_0}(k+1|k), \ldots, u^{\mathrm{o}}_{j_0,\ldots,j_{N-2}}(k+N-1|k), \quad \begin{cases} j_m = 1,\ldots,r, \\ m = 0,\ldots,N-2, \end{cases} \tag{3.12}$$

$$x^{\mathrm{o}}_{j_0}(k+1|k), \ldots, x^{\mathrm{o}}_{j_0,\ldots,j_{N-1}}(k+N|k), \quad \begin{cases} j_m = 1,\ldots,r, \\ m = 0,\ldots,N-1, \end{cases} \tag{3.13}$$

and a new state measurement $x(k+1) \equiv x(k|k)$ is obtained. Since it is assumed that the real system is described by (3.1)-(3.2), it is possible to write

$$x(k+1) = \sum_{j=1}^{r} \lambda_j(k) x^{\mathrm{o}}_j(k+1|k), \qquad \text{with} \qquad \lambda_j(k) \geq 0, \sum_{j=1}^{r} \lambda_j(k) = 1. \tag{3.14}$$

Using these uncertainty coefficients $\lambda_j = \lambda_j(k)$ it is possible to construct a feasible input sequence at time $k+1$:

$$u^{\mathrm{f}}_{j_1,\ldots,j_{i-1}}(k+i|k+1) = \sum_{j_0=1}^{r} \lambda_{j_0} u^{\mathrm{o}}_{j_0,j_1,\ldots,j_{i-1}}(k+i|k), \quad \begin{cases} j_m = 1,\ldots,r, \\ m = 1,\ldots,i-1, \\ i = 1,\ldots,N-1, \end{cases} \tag{3.15}$$

$$u^{\mathrm{f}}_{j_1,\ldots,j_{N-1}}(k+N|k+1) = -K \sum_{j_0=1}^{r} \lambda_{j_0} x^{\mathrm{o}}_{j_0,j_1,\ldots,j_{N-1}}(k+N|k),$$

$$\begin{cases} j_m = 1, \dots, r, \\ m = 1, \dots, N-1, \end{cases} \tag{3.16}$$

with a corresponding feasible state sequence

$$x^{\mathrm{f}}_{j_1,\dots,j_{i-1}}(k+i|k+1) = \sum_{j_0=1}^{r} \lambda_{j_0} x^{\mathrm{o}}_{j_0,j_1,\dots,j_{i-1}}(k+i|k), \quad \begin{cases} j_m = 1, \dots, r, \\ m = 1, \dots, i-1, \\ i = 2, \dots, N, \end{cases} \tag{3.17}$$

$$\begin{aligned} x^{\mathrm{f}}_{j_1,\dots,j_N}(k+N+1|k+1) &= (A_{j_N} - B_{j_N}K) \sum_{j_0=1}^{r} \lambda_{j_0} x^{\mathrm{o}}_{j_0,j_1,\dots,j_{N-1}}(k+N|k), \\ &= (A_{j_N} - B_{j_N}K) x^{\mathrm{f}}_{j_1,\dots,j_{N-1}}(k+N|k+1), \\ &\qquad \begin{cases} j_m = 1, \dots, r, \\ m = 1, \dots, N. \end{cases} \end{aligned} \tag{3.18}$$

Due to convexity of $\mathcal{U}$ and $\mathcal{X}$ and condition (3.11a) it is guaranteed that the inputs and states constructed in (3.15) and (3.17) respectively satisfy constraints (3.10d) and (3.10b) in the optimization problem at time $k+1$. Due to convexity of $\mathcal{X}_N$ and condition (3.11b) it is also guaranteed that inputs (3.16) satisfy (3.10d) at time $k+1$. Finally, due to condition (3.11c) and convexity of $\mathcal{X}_N$ the states (3.18) satisfy (3.10c). Finally, by construction the above proposed inputs and states also satisfy conditions (3.9), which shows that the proposed input and state sequence is indeed feasible for the optimization problem at time $k+1$, which proves recursive feasibility.

Once the feasible solution at time $k+1$ is constructed, the proof of robust asymptotic stability is similar to the proof of Lemma 1.2 and is hence omitted. □

3.3.3 Assessment

As shown by Lemma 3.1, Algorithm 3.2 guarantees recursive feasibility, whereas Algorithm 3.1 in its present form does not guarantee recursive feasibility in the presence of state constraints. The main reason for this is not the absence of a terminal contraint in Algorithm 3.1, but the fact that no feedback is considered within the horizon. Even if a terminal constraint would be added to Algorithm 3.1, recursive feasibility would still not be guaranteed.

Insight into this issue can be gathered by looking at equation (3.16), where it becomes clear that the MPC algorithm has enough degrees of freedom to choose a different input vector for each terminal state obtained at time k. As a result the corresponding new terminal states (3.18) are kept within the terminal constraint by the closed-loop dynamics $A_{j_N} - B_{j_N}K$ of the terminal controller and the LPV system.

When doing open-loop predictions only one control action is available, which is in general not sufficient to keep all the terminal states within the terminal constraint. It is hence due to the use of an open-loop input sequence that recursive feasibility typically cannot be guaranteed in open-loop min-max MPC algorithms. An exception to this observation is the combination of an open-loop input sequence with a time-varying horizon length, as is described in [143].

On the other hand, open-loop min-max MPC exhibits a lower computational cost because a significantly smaller number of optimization variables is involved. Furthermore, the open-loop input sequence parametrization is notationally somewhat more transparent, which *seems* to facilitate analysis of the algorithm.

Mainly due to the computational advantage and despite the fact that open-loop input sequences typically do not lead to recursively feasible algorithms, some recently published algorithms still aim to obtain recursive feasibility while employing open-loop input sequence parameterizations. The next two sections show that this is not always done successfully.

3.4 Corrections to [Wan *et al.*, 2003]

In this section we treat the robust MPC algorithm introduced in [142]. Despite the use of an open-loop input parametrization, the algorithm is claimed to be recursively feasible. The contribution of this section is the detection of an error in the recursive feasibility of this algorithm. By means of a counterexample the algorithm is shown not to be recursively feasible in general. A correction to the algorithm is proposed and illustrated by means of the same numerical example.

3.4.1 Introduction

The problem considered in [142] is that of controlling a disturbance-free LPV system (3.1)-(3.2) subject to input constraints $|u(k)| \leq u_{\max}, k \in \mathbb{N}$. A nominal model $[\hat{A}\ \hat{B}] \in \Omega$, representing the most likely model of the true system is also assumed to be known. The aim is to construct a model predictive controller to robustly, asymptotically, stabilize the system with (1.3) and $x_{\mathrm{ref}}(k) \equiv 0, u_{\mathrm{ref}}(k) \equiv 0$ as a control objective.

The algorithm proposed in [142] addresses this issue by using a finite horizon length, within which an open-loop input sequence is employed, combined with a time-varying terminal constraint set that is imposed on the terminal states of the state prediction tree. The stability proof of the algorithm is based on the assertion that, due to the fact that the terminal constraint set is invariant with respect to a robustly stabilizing terminal controller, each terminal state is driven further inside the terminal constraint set. However, due to the choice of a deterministic input sequence, proving stability requires the construction of a single input vector that simultaneously drives all terminal states further inside the terminal constraint set. For unstable systems this cannot be guaranteed based upon the invariance property, that only guarantees that such an input vector exists for each terminal state individually. For this reason the algorithm proposed in [142] cannot be proven to be asymptotically stabilizing, neither can it be guaranteed to be feasible if it is initially feasible.

In this section we present a new algorithm with a time-varying terminal constraint set that optimizes over a closed-loop input sequence rather than an open-loop input sequence, similar to Algorithm 3.2. Due to this modification, the invariance property of the terminal constraint becomes a sufficient condition for stability of the controller.

This section	[105]	[142]	
n_x, n_u	n, m	n, m	state and input dimensionality
Q, R	Q, R	$\mathscr{Q}, \mathscr{R}$	state and input weights
$\mathcal{X}_N, \mathcal{F}_N(\cdot)$	$\mathscr{X}_{\mathrm{f}}, \mathscr{F}(\cdot)$	$\mathscr{X}_{\mathrm{f}}, \mathscr{F}(\cdot)$	terminal constraint and cost
$\kappa_N(\cdot)$	$\kappa_{\mathrm{f}}(\cdot)$	$\kappa_{\mathrm{f}}(\cdot)$	terminal controller
Z_i	Q_i	Q_i	terminal constraint parameter
γ_i, X_i, Y_i	γ_i, X_i, Y_i	γ_i, X_i, Y_i	terminal cost, controller param.
r	L	L	number of vertices describing Ω
ρ_i	r_i	r_i	radius of ball inscribed in i-th terminal constraint set
$\mathbf{u}_N(k)$	$U(k)$	$U(k)$	open-loop input sequence
$\mathbf{u}_{\mathrm{cl},N}(k)$	$\mathbf{u}_N(k)$	/	closed-loop input sequence
$\mathcal{X}_{\mathbf{u}_N}(k+i\|k)$	$\mathscr{X}(k+i\|k)$	$\mathscr{X}(k+i\|k)$	state prediction sets
$u_{j_0,\ldots,j_{p-1}}(\cdot)$	$u_{j_{p-1},\ldots,j_0}(\cdot)$	/	closed-loop input vector
$\lambda_{j_i}(k+i)$	$c_{i+1,j_{i+1}}$	$c_{i+1,j_{i+1}}$	model uncertainty coefficients at time $k+i$

Table 3.1: Notational differences and similarities between this section, [105] and [142].

3.4.2 Time-varying terminal constraint set

In this section a brief description is given of the off-line part of the corrected algorithm, which is identical to that of the original algorithm. For the sake of brevity in this section we refer to equations of the original paper [142] with the notation $(\cdot)^\star$. Similar notations will be used to refer to theorems, corollaries and algorithms. For the sake of uniformity with other parts of this thesis, some notations used in this section deviate from the notations used [105, 142]. Table 3.1 gives an overview of differing notations.

We make use of the classical ingredients *a) a terminal controller* $\kappa_N(\cdot) : \mathbb{R}^{n_x} \to \mathbb{R}^{n_u}$*, b) a terminal cost* $\mathcal{F}_N(\cdot) : \mathbb{R}^{n_x} \to \mathbb{R}$ *and c) a terminal constraint* $\mathcal{X}_N \subset \mathbb{R}^{n_x}$. The latter two elements can be interpreted as an upper bound to the (in this case nominal) control cost of the terminal controller, when applied at the end of the control horizon of the MPC controller. The terminal constraint represents a feasible invariant set associated with the terminal controller, given the aforementioned input constraints. See [82, 142] for details. Theorem $1^\star$ allows the construction of a triplet $(\mathcal{X}_N, \kappa_N(\cdot), \mathcal{F}_N(\cdot))$ parameterized by variables γ, X, Y, Z as $\mathcal{X}_N = \{x \in \mathbb{R}^n | x^{\mathrm{T}} Z^{-1} x \leq 1\}$, $\kappa_N(x) = YZ^{-1}x$ and $\mathcal{F}_N(x) = x^{\mathrm{T}}\gamma Z^{-1}x$, with (γ, X, Y, Z) satisfying $(5)^\star, (6)^\star, (9)^\star$ and $(11)^\star$ for a given $\rho > 0$, denoting the radius of a hyperball inscribed in $\mathcal{X}_N$. $\mathcal{X}_N$ is an invariant ellipsoid with respect to $\kappa_N(\cdot)$, meaning that

$$x \in \mathcal{X}_N \Rightarrow A(k)x + B(k)\kappa_N(x) \in \mathcal{X}_N, \qquad \forall [A(k)\ B(k)] \in \Omega. \tag{3.19}$$

Corollary $1^\star$ then allows the construction of a continuum $(\mathcal{X}_N(\theta), \kappa_N(\theta, \cdot), \mathcal{F}_N(\theta, \cdot))$ based on two sets of parameters $(\gamma_1, X_1, Y_1, Z_1)$ and $(\gamma_0, X_0, Y_0, Z_0)$, obtained with Theorem $1^\star$ for two values ρ_0 and ρ_1 with $\rho_1 > \rho_0$, by considering the convex

combination

$$(\gamma(\theta), X(\theta), Y(\theta), Z(\theta)) = \theta(\gamma_1, X_1, Y_1, Z_1) + (1-\theta)(\gamma_0, X_0, Y_0, Z_0), \quad (3.20)$$

with $\theta \in [0,1]$ and constructing the corresponding $(\mathcal{X}_N(\theta), \kappa_N(\theta,\cdot), \mathcal{F}_N(\theta,\cdot))$ as in Theorem $1^\star$. See also Appendix A for more details. Algorithm $1^\star$ makes use of Theorem $1^\star$ and Corollaries $1^\star$ and $2^\star$ to construct $(\gamma_0, X_0, Y_0, Z_0)$ and $(\gamma_1, X_1, Y_1, Z_1)$ and the corresponding continuum of terminal constraint sets in a practical way.

3.4.3 Recursive feasibility

The proof of Theorem $2^\star$ asserts that, given optimal solutions $\mathbf{u}^{\mathrm{o}}(k), \mathcal{X}^{\mathrm{o}}(k+i|k), \theta^{\mathrm{o}}(k)$ at time k, one can find a feasible input sequence

$$\mathbf{u}_N^{\mathrm{f}}(k+1) = \left[u^{\mathrm{o}}(k+1|k);\ \ldots\ ; u^{\mathrm{o}}(k+N-1|k);\ u^{\mathrm{f}}(k+N|k+1)\right], \quad (3.21)$$

for the optimization problem at time $k+1$, such that the corresponding state prediction set $\mathcal{X}_{\mathbf{u}_N^{\mathrm{f}}(k+1)}(k+N+1|k+1)$ satisfies $\mathcal{X}_{\mathbf{u}_N^{\mathrm{f}}(k+1)}(k+N+1|k+1) \subset \mathcal{X}_N(\theta^{\mathrm{o}}(k))$, by applying the terminal controller $\kappa_N(\theta^{\mathrm{o}}(k),\cdot)$ to the terminal state $x(k+N|k)$.

However, this terminal state is not uniquely determined due to the unknown coefficients $\lambda_{j_i}(k+i), j_i = 1,\ldots,r, i = 0,\ldots,N-1$ (cfr. expressions between $(3)^\star$ and $(4)^\star$). It is therefore unclear what value to choose for $u^{\mathrm{f}}(k+N|k+1)$. Neither can it be guaranteed that there exists any value for $u^{\mathrm{f}}(k+N|k+1)$, such that $\mathcal{X}_{\mathbf{u}_N^{\mathrm{f}}(k+1)}(k+N+1|k+1) \subset \mathcal{X}_N(\theta^{\mathrm{o}}(k))$. This would require that

$$Ax + Bu^{\mathrm{f}}(k+N|k+1) \in \mathcal{X}_N(\theta^{\mathrm{o}}(k)), \quad \forall [A\ B] \in \Omega, \forall x \in \mathcal{X}_N(\theta^{\mathrm{o}}(k)), \quad (3.22)$$

while set invariance of $\mathcal{X}_N(\theta^{\mathrm{o}}(k))$ only guarantees that

$$Ax + B\kappa_N(\theta^{\mathrm{o}}(k), x) \in \mathcal{X}_N(\theta^{\mathrm{o}}(k)), \quad \forall [A\ B] \in \Omega, \forall x \in \mathcal{X}_N(\theta^{\mathrm{o}}(k)), \quad (3.23)$$

which means that a different input vector is used for each $x \in \mathcal{X}_N(\theta^{\mathrm{o}}(k))$, which conflicts with condition (3.22). Therefore, it is not always possible to find an appropriate value for $u^{\mathrm{f}}(k+N|k+1)$ and hence $\theta^{\mathrm{o}}(k)$ is not necessarily non-decreasing, neither is recursive feasibility guaranteed.

However, in the case that $\mathcal{X}_N(\theta^{\mathrm{o}}(k))$ is invariant with respect to the open-loop model $x_{k+1} = A(k)x_k$ with $A(k) \in \mathrm{Co}\{A_1,\ldots,A_r\}$, one can choose $u^{\mathrm{f}}(k+N|k+1) = 0$. This suggests an explanation why the example in [142] does not result in unstable closed-loop behavior. This also suggests the possible successful application of the original algorithm to certain classes of stable systems.

3.4.4 Feedback MPC formulation

In this section we present the on-line part of the new algorithm and the different modes of operation that are employed. The focus is on the introduction of feedback within the horizon as in Algorithm 3.2, in order to be able to guarantee recursive feasibility.

The modified algorithm uses a closed-loop input sequence $\mathbf{u}_{\mathrm{cl},N}$ instead of an open-loop input sequence $\mathbf{u}_N$. For notational simplicity we define r sub-sequences $\mathbf{u}_{\mathrm{cl},j}(k+p|k)$ of $\mathbf{u}_{\mathrm{cl}}(k+p|k)$:

$$\mathbf{u}_{\mathrm{cl},j}(k+p|k) \triangleq \begin{bmatrix} u_{j,1,\ldots,1}(k+p|k) \\ u_{j,1,\ldots,2}(k+p|k) \\ \vdots \\ u_{\underbrace{j,r,\ldots,r}_{p}}(k+p|k) \end{bmatrix}, \qquad p=0,\ldots,N-1, \tag{3.24}$$

such that $\mathbf{u}_{\mathrm{cl}}(k+p|k) = [\mathbf{u}_{\mathrm{cl},1}(k+p|k);\ldots;\mathbf{u}_{\mathrm{cl},r}(k+p|k)]$. Please note that a slightly different notation as in [105] is used in order to be consistent with the notation of Algorithm 3.2. This tree of inputs implicitly defines a control policy for all possible combinations of $[A(k+p)\ B(k+p)] \in \Omega, p=0,\ldots,N-1$, since these can be described as convex combinations of the nodes of the polytope Ω^{N-1}. A corresponding state prediction tree $x_{j_0,\ldots,j_{p-1}}(k+p|k)$ with $j_{0\ldots p-1}=1,\ldots,r$ and $p=1,\ldots,N$ is constructed as in (3.9c). Similar notations $\mathbf{x}_j(k+p|k)$ and $\mathbf{x}(k+p|k)$ with $p=0,\ldots,N$ and $\mathbf{x}_N(k)$ will be used in the remainder of this section.

In an initial phase (*mode 1*), an optimization is performed over the inputs $\mathbf{u}_{\mathrm{cl},N}(k)$ in order to minimize the size of the terminal constraint set, characterized by the parameter θ:

$$\min_{\mathbf{u}_{\mathrm{cl},N}(k),\theta} \ \theta, \tag{3.25a}$$

subject to

$$|u_{j_0,\ldots,j_{p-1}}(k+p|k)| < u_{\max}, \qquad \begin{cases} j_m=1,\ldots,r, \\ m=0,\ldots,p-1, \\ p=0,\ldots,N-1, \end{cases} \tag{3.25b}$$

$$x_{j_0,\ldots,j_{N-1}} \in \mathcal{X}_N(\theta), \qquad \begin{cases} j_m=1,\ldots,r, \\ m=0,\ldots,N-1, \end{cases} \tag{3.25c}$$

$$0 \leq \theta \leq 1. \tag{3.25d}$$

The optimal value of θ at time k is denoted as $\theta^{\circ}(k)$.

A second phase (*mode 2*) is initiated when at the previous time step $k-1$ the smallest terminal constraint $\theta^{\circ}(k-1)=0$ is obtained. In this phase the horizon length is reduced with 1 at each time step ($N:=N-1$) and an MPC problem with a nominal cost objective is solved. For the sake of clarity and without loss of generality, we assume that the nominal system model is given by $[\hat{A}\ \hat{B}]=[A_1\ B_1]$. The nominal state and input predictions now become $\hat{x}(k+p|k)=x_{1,\ldots,1}(k+p|k)$ and $\hat{u}(k+p|k)=u_{1,\ldots,1}(k+p|k)$. The following optimization problem is obtained:

$$\min_{\hat{\mathbf{u}}_N(k)} \sum_{i=0}^{N-1} \|\hat{u}(k+i|k)\|_R + \sum_{i=0}^{N-1} \|\hat{x}(k+i|k)\|_Q + \|\hat{x}(k+N|k)\|_{\gamma_0 Z_0^{-1}}, \tag{3.26}$$

where $\hat{\mathbf{u}}_N(k) \triangleq [\hat{u}(k|k); \ldots; \hat{u}(k+N-1|k)]$, subject to (3.25b) and (3.25c) with $\theta = 0$.

If at the previous time step $k-1$ a mode 2 optimization problem was solved with $N = 1$, a third and final phase (*mode 3*) is initiated. In this mode, the current state $x(k)$ is guaranteed to be positioned within the smallest terminal constraint $\mathcal{X}_N(0)$ and therefore the corresponding terminal control law $u(k) = \kappa_N(0, x(k)) = Y_0 Z_0^{-1} x(k)$ is applied to further drive the system to the origin.

These 3 modes of operation can be summarized in the following new algorithm:

Algorithm 3.3. *Initialize $N := N_0$ and $mode := 1$. Given a system described as* (3.1)-(3.2)*, input constraints $u_{\max}$, state and input weighting matrices $Q \in \mathbb{S}^{n_x}_{++}$ and $R \in \mathbb{S}^{n_u}_{++}$ and two sets $(\gamma_0, X_0, Y_0, Z_0)$ and $(\gamma_1, X_1, Y_1, Z_1)$ calculated using Algorithm* $1^\star$ *and given the current state $x(k)$, perform at each time step k the following steps:*

- *If $mode = 1$, solve optimization problem* (3.25) *and apply $u(k|k)$ to the system. If $\theta^{\mathrm{o}}(k) = 0$ set $mode := 2$. Wait until time step $k+1$.*
- *If $mode = 2$, set $N := N - 1$, solve optimization problem* (3.26) *subject to* (3.25b) *and* (3.25c) *with $\theta = 0$ and apply $u(k|k)$ to the system. If $N = 1$, set $mode := 3$. Wait until time step $k+1$.*
- *If $mode = 3$, apply $u(k) = Y_0 Z_0^{-1} x(k)$ to the system. Wait until time step $k+1$.*

3.4.5 Feasibility and asymptotic stability

The following theorem is proven as a corrected version to Theorem $2^\star$.

Theorem 3.1. *Given a system described as* (3.1)-(3.2)*, input constraints $u_{\max}$, state and input weighting matrices Q and R and an initial horizon length N_0. If optimization problem* (3.25) *is feasible at time $k = 0$ for the initial state $x(0)$ and $N = N_0$, then Algorithm 3.3 is also feasible for $k > 0$ and robustly, asymptotically stabilizes* (3.1)-(3.2)*.*

Proof: We prove that under the given assumptions modes 1 and 2 are feasible and terminate in a finite number of time steps. Therefore robust asymptotic stability is proven if mode 3 is robustly asymptotically stable.

First, we prove by induction that mode 1 is feasible and terminates in a finite amount of time. By assumption a feasible mode 1 solution $\mathbf{u}^{\mathrm{o}}_{\mathrm{cl},N}(k-1), \mathbf{x}^{\mathrm{o}}_N(k-1)$ and $\theta^{\mathrm{o}}(k-1)$ exists for each $k > 0$. We will now construct a feasible mode 1 solution $\mathbf{u}^{\mathrm{f}}_{\mathrm{cl},N}(k), \mathbf{x}^{\mathrm{f}}_N(k)$ and $\theta^{\mathrm{f}}(k)$ with $\theta^{\mathrm{f}}(k) < \theta^{\mathrm{o}}(k-1)$. Since $[A(k)\ B(k)] \in \Omega$, it is possible to find values $\lambda_1(k), \ldots, \lambda_r(k)$, with $\lambda_i(k) \geq 0$ and $\sum_{i=1}^r \lambda_i(k) = 1$, such that $x(k|k) = \sum_{i=1}^r \lambda_i(k) x^{\mathrm{o}}_i(k|k-1)$. Using these values $\lambda_{1\ldots r}(k)$ it is possible to construct a new set of feasible inputs

$$\mathbf{u}^{\mathrm{f}}_{\mathrm{cl},N}(k+p|k) = \sum_{i=1}^{r} \lambda_i(k) \mathbf{u}^{\mathrm{o}}_{\mathrm{cl},i}(k+p|k-1), \qquad p = 0, \ldots, N-2, \tag{3.27a}$$

and

$$u^{\mathrm{f}}_{j_0,\ldots,j_{N-2}}(k+N-1|k) = \kappa_N(\theta^{\mathrm{o}}(k-1), x^{\mathrm{f}}_{j_0,\ldots,j_{N-2}}(k+N-1|k)), \tag{3.27b}$$

with $j_m = 1, \ldots, r,\ m = 0, \ldots, N-2$. A set of feasible states can be constructed likewise:

$$\mathbf{x}^{\mathrm{f}}(k+p|k) = \sum_{i=1}^{r} \lambda_i(k) \mathbf{x}_i^{\mathrm{o}}(k+p|k-1), \qquad p = 0, \ldots, N-1, \tag{3.28}$$

and $x^{\mathrm{f}}_{j_0,\ldots,j_{N-1}}(k+N|k)$ defined using (3.9c). It is clear that through construction these sets of inputs and states satisfy (3.9c). Due to the convexity of the terminal constraint set, it is also clear that by construction all the states from $\mathbf{x}^{\mathrm{f}}(k+N-1|k)$ also lie within the terminal constraint set $\mathcal{X}_N(\theta^{\mathrm{o}}(k-1))$. Due to the invariance of this terminal constraint set, the strict inequality in (5)* and the construction of $\mathbf{u}^{\mathrm{f}}_{\mathrm{cl},N}(k+N-1|k)$, all terminal states from $\mathbf{x}^{\mathrm{f}}(k+N|k)$ lie in the strict interior of $\mathcal{X}_N(\theta^{\mathrm{o}}(k-1))$. Therefore $\inf_k(\theta^{\mathrm{o}}(k-1) - \theta^{\mathrm{o}}(k)) > 0$, which proves the fact that mode 1 terminates in a finite amount of time.

In mode 2 the horizon length is decreased by 1 in each time step, so it is trivial to prove that the condition $N = 1$ will be satisfied in a finite number (i.e. $N_0 - 1$) of time steps. By constructing a feasible mode 2 solution $\mathbf{u}^{\mathrm{f}}_{N-1}(k), \mathbf{x}^{\mathrm{f}}_{N-1}(k)$ using (3.27a) and (3.28) feasibility is also trivially guaranteed.

By means of a similar argument one can easily see that if $N = 1$ at time $k-1$, the current state $x(k)$ will lie in the terminal constraint set $\mathcal{X}_N(0)$, which legitimates the use of the terminal controller $\kappa_N(0, \cdot)$ in this mode, since it robustly asymptotically stabilizes (3.1)-(3.2) for all initial states that belong to its invariant set $\mathcal{X}_N(0)$. See [68, 142] for details. This, combined with the above arguments, proves robust asymptotic stability of Algorithm 3.3. □

3.4.6 Example

In this section we present a numerical example that clearly illustrates the flaw in Algorithm 2* and shows the improvement obtained with Algorithm 3.3. Consider a system with $L = 2, m = 1, n = 2$, described by

$$A_1 = \begin{bmatrix} 1 & 0 \\ -0.3 & 1.3 \end{bmatrix}, \qquad A_2 = \begin{bmatrix} 1 & 0 \\ -0.1 & 1.1 \end{bmatrix}, \tag{3.29a}$$

$$B_1 = [0.2 \quad 0]^{\mathrm{T}}, \qquad B_2 = [1 \quad 0]^{\mathrm{T}}, \tag{3.29b}$$

with input constraint $|u| \leq 1$. Initial horizons $N_0 \in \{4, 5\}$ and input and state cost matrices $Q = \mathrm{diag}(1, 0.1)$ and $R = 0.001$ were chosen. The radius of the largest inscribed ball was chosen as $\rho_1 \in \{0.34, 0.35\}$. The radius of the smallest inscribed ball was found to be $\rho_0 = 0.27411$. Simulations were performed on a time-invariant system described by $[A_2\ B_2]$ with initial state $x_0 = [1\ 1]^{\mathrm{T}}$. The resulting total simulation control costs and computation times are given in Table 3.2, the sequence of $\theta^{\mathrm{o}}(k)$-values for $N_0 = 4$ is depicted in Figure 3.3. Algorithm 2* exhibits non-monotonically decreasing values of $\theta^{\mathrm{o}}(k)$ in mode 1 for all aforementioned values of ρ_1 and N_0 and becomes infeasible in mode 1 for $\rho_1 = 0.34, N_0 = 4$ due to (3.25d) and infeasible in mode 2 for $N_0 = 5$ due to (3.25c). Algorithm 3.3 exhibits monotonically decreasing values of $\theta^{\mathrm{o}}(k)$ in mode 1 and is feasible and asymptotically stable for all

	$\rho_1 = 0.34$	$\rho_1 = 0.35$	
Alg. 2*, $N_0 = 4$	inf. at $k = 2$	42.58	(0.39s)
Alg. 2*, $N_0 = 5$	inf. at $k = 29$	inf. at $k = 27$	(0.83s)
Alg. 3.3, $N_0 = 4$	30.74	30.73	(0.59s)
Alg. 3.3, $N_0 = 5$	30.79	30.78	(2.08s)

Table 3.2: Total simulation control cost for Algorithm 2* and Algorithm 3.3 for $\rho_1 \in \{0.34, 0.34\}$ and $N_0 \in \{4, 5\}$. The maximum computation time per iteration (P4-2GHz, MATLAB 6.5, LMI LAB 1.0.8) for $\rho_1 = 0.35$ is indicated between brackets.

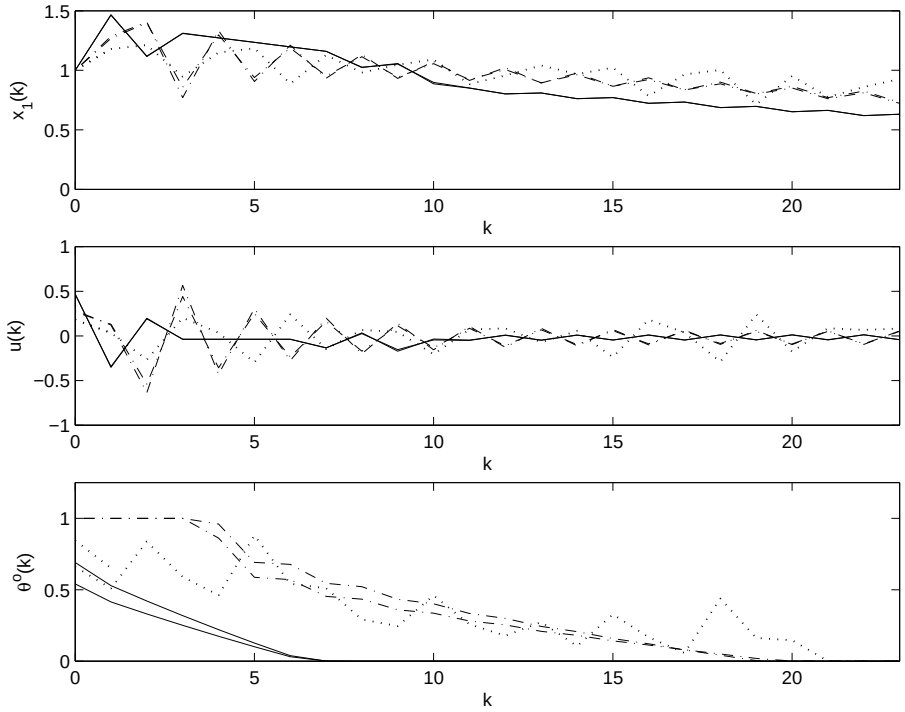


Figure 3.3: Values of $x_1(k), u(k)$ and $\theta^o(k)$ for Algorithm 2* (dotted), the new Algorithm 3.3 (solid) and the alternative Algorithm proposed in [143] (dash-dotted) for $\rho_1 = 0.34$ (larger θ-values) and $\rho_1 = 0.35$ (smaller θ-values) and $N_0 = 4$. Note that Algorithm 2* becomes infeasible for $\rho_1 = 0.34$ at time $k = 2$.

aforementioned values of ρ_1 and N_0, but at the cost of an increase in computational complexity.

3.4.7 Conclusion

In this section an existing algorithm [142] is successfully modified in order to guarantee recursive feasibility. A counterexample is provided indicating that the original algorithm was falsely claimed to be recursively feasible and that the new algorithm indeed solves the problem. No attempts have been made to improve other apparent deficiencies of the original algorithm, among which possible suboptimal behavior if R is given a relatively large value and the large maximum computational complexity per iteration.

The next section shows that a similar issue was already present in a less recently published algorithm, on which Algorithm $2^\star$ is partly based.

3.5 Corrections to [Casavola *et al.*, 2000]

3.5.1 Introduction

This section discusses an error present in [31] that is similar to the one discussed in the previous section. The paper is slightly less recent than [142], but no other publications appear to hint at possible errors in the paper under consideration, so it still is worthwhile to give an account of the misconceptions present in this paper.

The algorithms introduced in [31] are developed specifically for input-constrained linear systems with polytopic uncertainty description. The authors state that the algorithms are recursively feasible and asymptotically stable. In order to obtain these two properties, the authors make use of a terminal cost and a corresponding constraint set [82] that is recalculated at each time step using the results of [68]. Due to the lack of imposing the terminal constraint on the set of terminal states in the optimization problem of Algorithm 1 of [31] (in further sections referred to as Algorithm 1^*, with similar notation for Algorithm 2 of [31]) and the fact that an open-loop input sequence is used, it is not guaranteed that the algorithm is recursively feasible. Algorithm 2^* suffers from a similar deficiency, but for reasons of brevity we will focus on Algorithm 1^* in this section.

In the rest of this section *-notation is used to denote theorems, equations and algorithms of the original article [31]. We refer to table 3.3 and the original article for further details about other notations.

3.5.2 Deficiencies in original algorithm

Algorithm 1^* extends the results of [68] by adding N free control moves $u(k|k), \ldots, u(k+N-1|k)$ to the formulation. These control moves are meant to drive the system inside the time-dependent terminal constraint set $\mathcal{X}_N(k)$, which is taken to be the invariant ellipsoid corresponding to a time-dependent terminal controller gain $-K(k)$ constructed using [68]. At each time step $k \geq 0$, the terminal controller and

This section	[94]	[31]	
k	t	t	discrete time
i	k	k	within-horizon time index
n_x, n_u	n, m	n, m	state and input dimensionality
A_i, B_i	Φ_i, G_i	Φ_i, G_i	system matrices
Q, R	Ψ_x, Ψ_u	Ψ_x, Ψ_u	state and input weights
$\mathcal{U}$	Ω_u	Ω_u	input constraint set
Q_N	Q	Q	terminal cost matrix
$-K$	F	F	terminal controller gain
$\mathcal{X}_N$	$\mathcal{E}$	$\mathcal{E}$	terminal constraint set
γ, Y, Z	ρ, Y, P	ρ, Y, P	terminal cost, constraint and controller parameters
r	l	l	number of vertices describing Ω
$\mathbf{u}(k)$	$u(\cdot\|t)$	$u(\cdot\|t)$	open-loop input sequence
$\mathbf{u}_{\rm cl}(k)$	$\mathbf{u}(t)$	/	closed-loop input sequence
$u_{j_0,\ldots,j_{p-1}}(\cdot)$	$u_{j_{p-1},\ldots,j_0}(\cdot)$	/	closed-loop input vector
$u^{\rm f}(k+i\|k)$	$\bar{u}^*(t+k\|t)$	$\bar{u}^*(t+k\|t)$	candidate feasible input
$\lambda_{j_i}(k+i)$	$c_{i+1,j_{i+1}}$	$p_{j_i}(i)$	model uncertainty coefficients at time $k+i$

Table 3.3: Notational differences and similarities between this section, [94] and [31].

its invariant ellipsoid, are recomputed to take into account the (assumed) fact that the system has been driven closer to the origin. For reasons of brevity we refrain from entirely restating the algorithm, but refer to [31] for the details.

Two main deficiencies exist in Algorithm 1*:

- The terminal constraint $\mathcal{X}_N(k)$ is not explicitly imposed on the terminal states in optimization problem (28)*-(29)*, which corresponds to step 1 of Algorithm 1* and as a result feasibility of the terminal controller is not guaranteed.

- An open-loop sequence is employed and for reasons already stated in the previous sections, this in general does not guarantee recursive feasibility, even in the presence of an appropriate terminal constraint.

As a result of these deficiencies, Lemma 3* does not hold in general. Given an optimal input sequence $\mathbf{u}_N^{\rm o}(k)$ to (28)*-(29)* at time k, expression (37)* is used as a candidate feasible input sequence to (28)*-(29)* at time $k+1$. However, due to the fact that the terminal constraint $x \in \mathcal{X}_N(k), \forall x \in \text{vert}\{\mathcal{X}_{u_N^{\rm o}(k)}(k+N|k)\}$ is not explicitly imposed in (28)*-(29)*, this input sequence $\mathbf{u}_N^{\rm f}(k+1)$ is not guaranteed to be feasible, since $-K(k)x^{\rm o}(k+N|k)$ is not guaranteed to be feasible $\forall x^{\rm o}(k+N|k) \in \mathcal{X}_{\mathbf{u}_N^{\rm o}(k)}(k+N|k)$, which contradicts with the paragraph between (36)* and (38)*.

Assume that it would be guaranteed $\forall k$ that $\mathcal{X}_{\mathbf{u}_N^{\rm o}(k)}(k+N|k) \subseteq \mathcal{X}_N(k)$ and therefore that $-K(k)x^{\rm o}(k+N|k) \in \mathcal{U}$, even then monotonicity of the cost is not guaranteed. The expression right after (38)* assumes that the terminal cost value at time $k+1$

resulting from input sequence $\mathbf{u}_N^{\mathrm{f}}(k+1)$ can be upper bounded by

$$\max_{i\in\{1,\ldots,r\},z\in\mathrm{vert}\{\mathcal{X}_{\mathbf{u}_N^{\mathrm{o}}(k)}(k+N|k)\}} \|(A_i - B_iK(k))z\|^2_{Q_N(k+1)}, \tag{3.30}$$

which in turn is based on the incorrect assumption that in general

$$\begin{aligned}&\mathcal{X}_{\mathbf{u}_N^{\mathrm{f}}(k+1)}(k+N+1|k+1)\\ &\qquad\subseteq \{(A_i - B_iK(k))z | i\in\{1,\ldots,r\}, z\in\mathcal{X}_{\mathbf{u}_N^{\mathrm{o}}(k)}(k+N|k)\}.\end{aligned} \tag{3.31}$$

Due to invariance of $\mathcal{X}_N(k)$ it can be shown that the rhs of the above expression is a subset of $\mathcal{X}_N(k)$. On the other hand, due the specific choice of $\mathbf{u}^{\mathrm{f}}(k+1)$, it can be seen that

$$\begin{aligned}&\mathcal{X}_{\mathbf{u}_N^{\mathrm{f}}(k+1)}(k+N+1|k+1)\\ &\quad= \{A_iz - B_iK(t)x^{\mathrm{o}}(k+N|k) | i\in\{1,\ldots,r\}, z\in\mathcal{X}_{\mathbf{u}_N^{\mathrm{o}}(k)}(k+N|k)\},\end{aligned} \tag{3.32}$$

which, in general, is not a subset of $\mathcal{X}_N(k)$, since one cannot necessarily find a fixed control move (i.e. $-K(t)x^{\mathrm{o}}(k+N|k)$), that steers all terminal states further into the invariant ellipsoid $\mathcal{X}_N(k)$. This wrong assumption is similar to the error discussed in the previous section. The main consequence is that the inclusion (3.31) does not hold in general, that the terminal cost at time $k+1$ in general cannot be bounded by (3.30) and that therefore monotonicity of $W(k)$ is not guaranteed. Similarly, recursive feasibility cannot be guaranteed anymore, since in general it is not guaranteed that $\mathcal{X}_N(k+1)\subset\mathcal{X}_N(k)$.

By means of similar arguments one can also invalidate both claims of Lemma 2*.

3.5.3 Counterexample

In order to illustrate these findings, we consider a system of the form (1)*-(3)*, with $r=2$, defined as follows:

$$A_1 = \begin{bmatrix} 1 & 0 \\ -0.3 & 1.3 \end{bmatrix}, \qquad A_2 = \begin{bmatrix} 1 & 0 \\ -0.1 & 1.1+\delta \end{bmatrix}, \tag{3.33a}$$

$$B_1 = [0.2 \quad 0]^{\mathrm{T}}, \qquad B_2 = [1 \quad 0]^{\mathrm{T}}, \tag{3.33b}$$

with $\delta\in[0,1]$ a parameter that is fixed in time and known during the controller synthesis. For $\delta=0$, this system is identical to (3.29). The system is subject to input constraint $|u(k)|\leq 1, \forall k\geq 0$. Cost matrices are chosen as $Q=\mathrm{diag}(1,0.1)$ and $R=0.001$. The controller horizon is chosen as $N=2$.

Figure 3.4 shows simulation results starting from initial state $[1;1]$ for 4 different values of δ. The algorithm is initially feasible for all values of δ, but becomes infeasible after a few time steps for the two largest values of δ. Also the function $W(k)$ is not monotonically decreasing. The original example in [31] does not exhibit these problems because the system is already stable in open-loop.

3.5.4 Corrected algorithm

In order to correct the deficiencies of Algorithm 1*, two modifications are proposed:

- explicitly impose the terminal constraint set in the optimization problem (28)*-(29)*,
- use a closed-loop input sequence $\mathbf{u}_{\mathrm{cl},N}(k)$ instead of an open-loop input sequence.

As in the previous sections the closed-loop input sequence $\mathbf{u}_{\mathrm{cl},N}(k)$ and closed-loop state prediction sets $\mathcal{X}_{\mathbf{u}_{\mathrm{cl},N}(k)}(k+N|k)$ are defined as (3.8),(3.24) and (3.9) respectively.

We can now state the corrected algorithm in a straightforward way:

Algorithm 3.4. *At time $k=0$, given $x(0)$ solve the initialization step of Algorithm 1*, with the closed-loop input sequence $\mathbf{u}_{\mathrm{cl},N}(k)$ (instead of $\mathbf{u}_N(k)$) and state prediction polytope* (3.9)*. At every time $k \geq 0$ execute the following steps:*

1. *Given $x(k), Z(k), \gamma(k)$, solve the following optimization:*

$$\mathbf{u}^{\mathrm{o}}_{\mathrm{cl},N}(k) = \underset{J_{0\dots N}, \mathbf{u}_{\mathrm{cl},N}(k)}{\operatorname{argmin}} \sum_{i=0}^{N} J_i, \tag{3.34}$$

subject to

$$\begin{bmatrix} J_N & * \\ z & \gamma^{-1}(k)Z(k) \end{bmatrix} \geq 0, \qquad \forall z \in \operatorname{vert}\{\mathcal{X}_{\mathbf{u}_{\mathrm{cl},N}(k)}(k+i|k)\}, \tag{3.35}$$

$$J_N \leq \gamma(k), \tag{3.36}$$

$$u_{j_0,\dots,j_{i-1}}(k+i|k) \in \mathcal{U}, \qquad \begin{cases} j_m = 1,\dots,r, \\ m = 0,\dots,i-1, \\ i = 0,\dots,N-1, \end{cases} \tag{3.37}$$

$$\begin{bmatrix} 1 & * & * \\ Q^{\frac{1}{2}} x_{j_0,\dots,j_{i-1}}(k+i|k) & J_i I & * \\ R^{\frac{1}{2}} u_{j_0,\dots,j_{i-1}}(k+i|k) & 0 & J_i I \end{bmatrix} \geq 0, \qquad \begin{cases} j_m = 1,\dots,r, \\ m = 0,\dots,i-1, \\ i = 0,\dots,N-1. \end{cases} \tag{3.38}$$

2. *Apply $u^{\mathrm{o}}(k|k)$ to the plant.*
3. *Calculate $[Z(k+1), Y(k+1), \gamma(k+1)]$ using step 3 of Algorithm 1*, with the closed-loop definition of the state prediction polytope* (3.9)*.*

Theorem 3.2 (Feasibility and Stability). *Algorithm 3.4 is recursively feasible if the initialization step is feasible, in which case it also asymptotically stabilizes the system.*

Proof: Since step 1 is identical to the initialization step, except for the fixation of the terminal constraint and cost, it is straightforward that the former is feasible if the latter

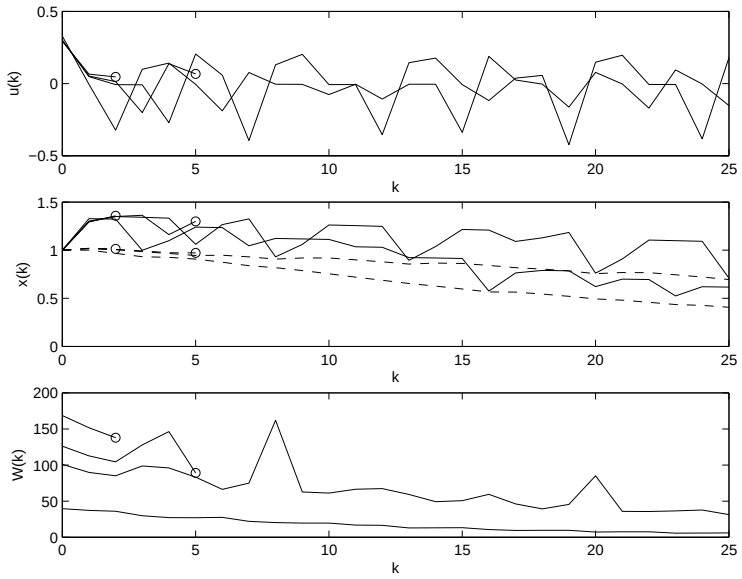


Figure 3.4: Application of Algorithm 1^* to system (3.33) for initial state $[1;1]$ and $\delta \in \{0, 0.016, 0.018, 0.020\}$. The algorithm does not result in a monotonically decreasing Lyapunov function $W(k)$ and becomes infeasible for the latter two values of δ at time steps $k = 3$ and $k = 6$ respectively.

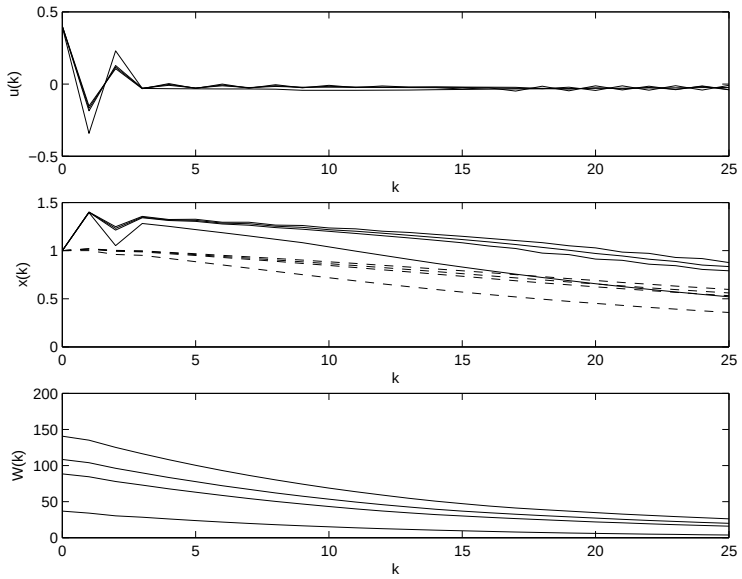


Figure 3.5: Application of Algorithm 3.4 to system (3.33) for initial state $[1;1]$ and $\delta \in \{0, 0.016, 0.018, 0.020\}$. The algorithm results in a monotonically decreasing Lyapunov function $W(k)$ and results in stable behavior for all 4 values of δ.

is feasible. This shows that step 1 is feasible at $k = 0$. If step 1 is feasible at time k, then step 3 is also feasible at time k. This is due to the fact that all terminal states lie inside $\mathcal{X}_n(k)$, since this is imposed by constraint (3.36). Therefore it is also guaranteed, due to invariance, that all states $(A_i - B_i K(k))z$ used in step 3 of Algorithm 1 also lie within $\mathcal{X}_N(k)$, which indicates that $Z(k), Y(k), \gamma(k)$ are feasible solutions to the optimization in this step. As a consequence the property $\gamma(k+1) \leq \gamma(k)$ also holds.

We now show that it is possible to construct a feasible solution to step 1 at time $k+1$. Because of (2)*, it is possible to find values $\lambda_1(k), \ldots, \lambda_r(k)$, such that $x(k+1) = \sum_{i=1}^{r} \lambda_r(k) x_i^{\mathrm{o}}(k+1|k)$. A feasible solution $\mathbf{u}_{\mathrm{cl},N}^{\mathrm{f}}(k+1)$ can now be constructed as

$$\mathbf{u}_{\mathrm{cl}}^{\mathrm{f}}(k+i|k+1) = \sum_{j=1}^{r} c_j \mathbf{u}_{\mathrm{cl},j}^{\mathrm{o}}(k+i|k), \qquad i = 1, \ldots, N-1,$$

and

$$\mathbf{u}_{\mathrm{cl},j_0,\ldots,j_{N-2}}^{\mathrm{f}}(k+N|k+1) = -K(k) x_{j_0,\ldots,j_{N-1}}^{\mathrm{f}}(k+N|k+1)), \qquad \begin{cases} j_m = 1, \ldots, r, \\ m = 0, \ldots, N-2. \end{cases}$$

The corresponding state predictions can be expressed similarly as

$$\mathbf{x}^{\mathrm{f}}(k+i|k+1) = \sum_{j=1}^{r} \lambda_j(k) \mathbf{x}_{\mathrm{cl},j}^{\mathrm{o}}(k+i|k), \qquad i = 1, \ldots, N,$$

and

$$x_{j_1,\ldots,j_N}^{\mathrm{f}}(k+N+1|k+1) = (A_{j_N} - B_{j_N} K(k)) x_{j_1,\ldots,j_{N-1}}^{\mathrm{f}}(k+N|k+1), \qquad \begin{cases} j_m = 1, \ldots, l, \\ m = 1, \ldots, N. \end{cases}$$

One can see that this input sequence satisfies (7)* for $i = 1, \ldots, N-1$ and, because all possible states $x(k+N|k+1) \in \mathcal{X}_N(k)$, also for $i = N$. Due to the specific choice of $\mathbf{u}_{\mathrm{cl}}^{\mathrm{f}}(k+N|k+1)$ and the way $\mathcal{X}_N(k+1)$ is calculated, one can see that the terminal constraint is also satisfied for the candidate input sequence proposed above. Therefore step 1 of Algorithm 1 is also feasible at time $k+1$, which then proves recursive feasibility. Furthermore, due to convexity and because the state predictions corresponding to $\mathbf{u}_{\mathrm{cl},N}^{\mathrm{f}}(k+1)$ can be expressed as convex combinations, one can see that

$$\mathcal{X}_{\mathbf{u}_{\mathrm{cl},N}^{\mathrm{f}}(k+1)}(k+i|k+1) \subseteq \mathcal{X}_{\mathbf{u}_{\mathrm{cl},N}^{\mathrm{o}}(k)}(k+i|k), \qquad i = 1, \ldots, N.$$

Combined with the observation that $Q_N(k)$ satisfies (19)* and that $\gamma(k+1) \leq \gamma(k)$ this then shows that $W(k+1) \leq W(k)$, which proves asymptotic stability, along the lines presented in [82]. □

Figure 3.5 shows the same simulation shown in Figure 3.4, now using Algorithm 3.4. Recursive feasibility is now obtained for all 4 values of δ. Also, $W(k)$ now behaves monotonically.

3.6 Conclusions

In this chapter the important issue of recursive feasibility has been discussed in the context of robust MPC. Compared to nominal MPC, this issue is somewhat more delicate and subtle and requires special attention when designing MPC algorithms. The use of open-loop input sequences, although attractive from a computational point of view, often does not lead to recursive feasibility. This is illustrated using two recently published papers [31, 142] that make errors in this respect.

An important observation that can be made is that recursive feasibility and computational simplicity seem to be incompatible design objectives. One of the aims of the following chapters is to provide methods to eliminate or alleviate this problem.

Chapter 4

Robust MPC using Polyhedral Invariant Sets

"Prediction is very difficult, especially about the future."

– Niels Bohr (1885-1962) –

This chapter discusses and extends three different types of robust MPC algorithms. First the MPC algorithm introduced by Kothare et al. [68] is discussed. Secondly, interpolation-based MPC algorithms are discussed and finally a robust quasi-infinite horizon algorithm is discussed. All three classes of algorithms are extended towards the use of polyhedral invariant sets instead of ellipsoidal invariant sets, leading to either reduced computational complexity, less conservative constraint handling or a combination of both. The first two classes are also extended on an algorithmic level towards more general cost objectives and constraints and to further enlarge the feasible region respectively. Numerical examples are provided, illustrating the obtained improvements.

4.1 Robust constrained linear state feedback synthesis

4.1.1 Introduction

This section combines and improves two complementary algorithms in order to obtain a computationally tractable control algorithm with improved control characteristics. On the one hand the robust MPC method introduced by Kothare *et al.* [68] is used. This algorithm is able to construct linear robust linear feedback laws for LPV systems subject to input and state constraints, but has the disadvantage that it uses ellipsoidal invariant sets. This can lead to conservative constraints handling, even when the

algorithm is applied in a receding horizon fashion, i.e. when a feedback law is recomputed at each time instant based on new state measurements. On the other hand the algorithm to construct polyhedral invariant sets for LPV systems, which is described in the previous chapter is used to obtain an improved characterization of the feasible region of the resulting controller. The downside of the latter algorithm is that it does not allow the simultaneous construction of a controller and the corresponding polyhedral invariant set, as does the algorithm discussed in [68].

This chapter extends these algorithms in two important ways. First the method of [68] is extended to also include mixed state/input constraints and cross-terms between states and inputs in the quadratic objective function. Secondly, the use of polyhedral invariant sets is introduced in the controller synthesis to assess the conservativeness of the constraint handling of the resulting controller and to iteratively recompute the feedback gain of this controller in order to improve the constraint handling. The resulting algorithm consists of the sequential solution of several SDPs. Both improvements were published in [93].

The method described in this section is related to MPC in several ways. First of all it also provides a method for off-line controller synthesis for systems subject to input/state constraints. It can be verified that the resulting controller, Lyapunov function and invariant sets satisfy the stability conditions (3.11) and as such can be used as terminal controller, cost and constraint respectively. On the other hand, the method can also be applied on-line in a receding horizon fashion by recalculating the feedback gain at every time step in order to further improve constraint handling. As such this method is also an MPC method in itself.

4.1.2 Problem formulation

This paper considers LPV systems (3.1) with polytopic uncertainty description (3.2), subject to state and input constraints (2.10)-(2.11). Later on in this section also mixed state and input constraints will be considered:

$$[x(k); u(k)] \in \mathcal{Y} \equiv \{y | A_{xu} y \leq \mathbf{1}\}, \qquad k \in \mathbb{N}. \tag{4.1}$$

The aim is to find a linear feedback controller

$$u(k) = -Kx(k), \qquad k \in \mathbb{N}, \tag{4.2}$$

that robustly asymptotically stabilizes (3.1)-(3.2) without violating constraints (2.10)-(2.11) and/or (4.1) for a given initial state $\bar{x} \in \mathbb{R}^n_x$. Optimality of the controller is defined using the following cost function, which is a slight generalization of (1.3):

$$J \equiv \sum_{k=0}^{\infty} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} Q & N \\ N^{\mathrm{T}} & R \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}, \tag{4.3}$$

with $N \in \mathbb{R}^{n_x \times n_u}$.

The problem discussed in this paper can more formally be summarized as follows:

Problem 4.1 (P4.1)**.** *Given a system* (3.1)-(3.2) *subject to constraints* (2.10), (2.11), (4.1)*, an optimality criterion defined as* (4.3) *and an initial state* $\bar{x} \in \mathbb{R}^{n_x}$*, find a feedback gain* K *such that the controller* (4.2) *results in a minimal worst-case (over all possible trajectories starting from the initial state* $\bar{x}$*) control cost* (4.3) *and without violating constraints* (2.10),(2.11),(4.1) *for any of the possible trajectories.*

The initial state $\bar{x}$ can be a state chosen off-line by the user in order to obtain a feedback controller with a desired feasible region. Alternatively P4.1 can be solved on-line at each time instant k, where $\bar{x}$ is then chosen as the current state measurement $x(k)$.

No exact solution to P4.1 exists, but a the following relaxation to P4.1 was solved in [68]:

Problem 4.2 (P4.2)**.** *Given a system* (3.1)-(3.2) *subject to constraints* (2.10)-(2.11)*, an optimality criterion defined as* (4.3) *with* $N = 0$ *and an initial state* $\bar{x} \in \mathbb{R}^{n_x}$*, find a feedback gain* K *such that*

- *the worst-case cost function* $V(\bar{x}) = \bar{x}^{\mathrm{T}} P \bar{x} \geq J(\bar{x})$ *is minimal with* $P \in \mathbb{S}^{n_x}_{++}$ *satisfying the Lyapunov inequality*

$$P - (A_i - B_i K)^{\mathrm{T}} P (A_i - B_i K) \succ Q + K^{\mathrm{T}} R K, \quad i = 1, \ldots, r, \tag{4.4}$$

- *the given initial state* $\bar{x}$ *lies within a feasible invariant ellipsoid* $\mathcal{E}$ *of the form*

$$\mathcal{E} \equiv \{x | x^{\mathrm{T}} P x \leq \gamma\}, \tag{4.5}$$

  with $P \in \mathbb{S}^{n_x}_{++}$ *and* $\gamma > 0$*.*

Appendix A describes the solution method to this relaxation. More details can be found in [68, 93].

4.1.3 Mixed state/input cost and constraints

This section eliminates the assumptions of P4.2 that $N = 0$ and that no mixed constraints (4.1) are present and shows how these more general situations still lead to a convex optimization problem.

4.1.3.1 Mixed state/input cost terms

In order to allow for $N \neq 0$ the Lyapunov inequality (4.4) needs to be adapted and reformulated in an LMI similar to (A.3c). By substitution of $u(k) = -Kx(k)$, the objective function (4.3) can be rewritten as

$$J(x(0)) \equiv \sum_{k=0}^{\infty} x(k)^{\mathrm{T}} \begin{bmatrix} I \\ -K \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} Q & N \\ N^{\mathrm{T}} & R \end{bmatrix} \begin{bmatrix} I \\ -K \end{bmatrix} x(k),$$

which then results in the following Lyapunov inequalities:

$$P - (A_i - B_i K)^{\mathrm{T}} P (A_i - B_i K) \succ$$

$$\begin{bmatrix} I \\ -K \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} Q & N \\ N^{\mathrm{T}} & R \end{bmatrix} \begin{bmatrix} I \\ -K \end{bmatrix}, \quad i = 1, \ldots, r. \quad (4.6)$$

After substitution of $K = -YZ^{-1}$ and $P = \gamma Z^{-1}$, left and right multiplication with $Z \in \mathbb{S}^{n_x}_{++}$ and division by $\gamma > 0$, these inequalities become

$$Z - (A_i Z + B_i Y)^{\mathrm{T}} Z^{-1} (A_i Z + B_i Y) \succ \frac{1}{\gamma} \begin{bmatrix} Z \\ Y \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} Q & N \\ N^{\mathrm{T}} & R \end{bmatrix} \begin{bmatrix} Z \\ Y \end{bmatrix}, \quad i = 1, \ldots, r. \quad (4.7)$$

By applying the Schur complement this can be formulated as the LMI

$$\begin{bmatrix} Z & * & * \\ A_i Z + B_i Y & Z & * \\ Q_{\mathrm{xu}}^{\frac{1}{2}} \begin{bmatrix} Z \\ Y \end{bmatrix} & 0 & \gamma I \end{bmatrix} \succ 0, \qquad i = 1, \ldots, r, \quad (4.8)$$

with $Q_{\mathrm{xu}} = \begin{bmatrix} Q & N \\ N^{\mathrm{T}} & R \end{bmatrix}$. This LMI replaces (A.3c) in case $N \neq 0$. If $N = 0$, both LMIs are easily shown to be equivalent.

4.1.3.2 Mixed state/input constraints

The aim of this subsection is to reformulate (4.1) into sufficient LMI conditions in the optimization variables γ, Y, Z. First, we rewrite (4.1) as

$$A_{xu,x} x(k) + A_{xu,u} u(k) \leq 1_v, \qquad k \in \mathbb{N},$$

where $A_{xu} = [A_{xu,x} \; A_{xu,u}]$. After substitution of $u(k) = -Kx(k)$ this can be rewritten as

$$[A_{xu,x} \; - A_{xu,u} K] x(k) \leq 1_v, \qquad k \in \mathbb{N},$$

which, for each row of A_{xu} separately, is satisfied if:

$$\max_{z \in \mathcal{E}} \| [(A_{xu,x})_{[j,:]} \; - A_{xu,u} K] z \| \leq 1, \qquad j = 1, \ldots, m_{xu},$$

with m_{xu} denoting the number of rows in A_{xu}. This is equivalent with

$$\bar{\sigma}([(A_{xu,x})_{[j,:]} \; - A_{xu,u} K] Z^{\frac{1}{2}}) \leq 1, \qquad j = 1, \ldots, m_{xu},$$

with $\bar{\sigma}(\cdot)$ denoting the largest singular value, which (similar to the derivation of the state constraint LMI in [68]) is satisfied if

$$\begin{bmatrix} Z & * \\ (A_{xu})_{[j,:]} \begin{bmatrix} Z \\ Y \end{bmatrix} & 1 \end{bmatrix} \succ 0, \qquad j = 1, \ldots, m_{xu}. \quad (4.9)$$

This LMI can be added as an additional constraint to (A.3) in case constraints of the form (4.1) are present.

4.1.3.3 Algorithm formulation

The two extensions described above are now summarized in the following algorithm:

Algorithm 4.1 (Constrained controller synthesis for LPV systems, [93]). *Given a system* (3.1)-(3.2) *subject to constraints* (2.10),(2.11),(4.1)*, optimality criterion* (4.3) *and an initial state* $\bar{x} \in \mathbb{R}^{n_x}$*, solve optimization problem* (A.3)*, with* (A.3c) *replaced with* (4.8) *and with additional constraint* (4.9)*. Return feedback gain* $K = -YZ^{-1}$.

4.1.4 Controller synthesis using polyhedral invariant sets

This section discusses how polyhedral invariant sets can be integrated in the synthesis process to reduce conservative constraint handling and obtain more optimal controllers. Two algorithms are formulated:

- An algorithm that consists of first applying Algorithm 4.1 after which a polyhedral invariant set is computed using Algorithm 2.4 resulting in an exact characterization of the feasible region of the closed loop system.
- An algorithm that consists of iteratively recomputing K using Algorithm 4.1 and $\mathcal{P}$ using Algorithm 2.4 in order to increase the optimality of K subject to the feasibility requirement $\bar{x} \in \mathcal{P}$.

The first algorithm obviously results in the same controller as Algorithm A.1, but returns a more exact characterization of the feasible region, whereas the second algorithm exploits the improved characterization of the feasible region to improve optimality of the controller.

Algorithm 4.2. *Given a system* (3.1)-(3.2) *subject to constraints* (2.10), (2.11), (4.1)*, optimality criterion* (4.3) *and an initial state* $\bar{x} \in \mathbb{R}^{n_x}$*, perform the following steps:*

- *Apply Algorithm 4.1 to obtain a feedback gain* K *and a Lyapunov function* $V(x) = x^{\mathrm{T}}Px$.
- *Apply Algorithm 2.4 to obtain the MAS* $\mathcal{P}$ *for the closed-loop system* (3.1), (3.2), (4.2) *subject to constraints* (2.10), (2.11), (4.1)*.*

Although Algorithm 4.2 is rather straightforward, there are a few interesting points to make:

- Since by construction $\bar{x} \in \mathcal{E}$, it is also guaranteed that $\bar{x} \in \mathcal{P}$,
- The Lyapunov function $V(x) = x^{\mathrm{T}}Px$, that is proven in [68] to be valid within $\mathcal{E}$ is also valid for all states $x \in \mathcal{P}$, since all the imposed constraints are satisfied for trajectories starting from such states.
- An upper bound to the control cost $J(\bar{x})$ is given by γ^{o}. This is due to the fact that in the optimum equation (A.3b) is satisfied with equality, i.e. $\gamma^{\mathrm{o}} = \bar{x}^{\mathrm{T}}P\bar{x} \equiv V(\bar{x})$. Since $V(x)$ is an upper bound to the worst case value of $J(x)$, γ serves as an upper bound to the control cost of trajectories starting from the initial value $\bar{x}$.

Based on these observations we can now formulate an algorithm that iteratively applies Algorithm 4.2 in order to find an optimal feedback gain K over all feedback gains for which $\bar{x} \in \mathcal{P}$ (instead of $\bar{x} \in \mathcal{E}$ in case of Algorithm 4.1).

Algorithm 4.3. *Given a system* (3.1)-(3.2) *subject to constraints* (2.10), (2.11), (4.1), *an optimality criterion* (4.3) *and an initial state* $\bar{x} \in \mathbb{R}^{n_x}$ *such that Algorithm 4.1 is feasible, solve the following optimization problem:*

$$\min_{c} \quad \gamma(c), \tag{4.10a}$$

$$\text{s.t.} \quad \bar{x} \in \mathcal{P}_{K(c)}, \tag{4.10b}$$

where $\gamma(c)$ *and* $K(c)$ *are the values obtained with Algorithm 4.1 for relaxed constraints* $\mathcal{X}', \mathcal{U}', \mathcal{Y}'$*:*

$$\mathcal{X}' = c\mathcal{X}, \qquad \mathcal{U}' = c\mathcal{U}, \qquad \mathcal{Y}' = c\mathcal{Y}, \tag{4.11}$$

with c *a positive scalar.* $\mathcal{P}_{K(c)}$ *is the MAS for the closed loop system* (3.1), (3.2), (4.2) *(with* $K = K(c)$*) subject to constraints* (2.10), (2.11), (4.1)*. Return* $K(c^{\mathrm{o}})$ *and* $\mathcal{P}_{K(c^{\mathrm{o}})}$*, with* c^{o} *denoting the optimal solution of* (4.10)*.*

Optimization problem (4.10) is a scalar optimization problem with a monotonically decreasing objective function $\gamma(c)$. Therefore the problem is reduced to finding the largest value of c for which (4.10b) is still satisfied. Since in typical situations the set $\mathcal{C} \triangleq \{c | \bar{x} \in \mathcal{P}_{K(c)}\}$ is convex, one can solve optimization problem (4.10) by means of interval reduction techniques, e.g. bisection search. In degenerate cases where $\mathcal{C}$ is not convex, one can still easily find a feasible solution due to the following lemma.

Lemma 4.1. *Optimization problem* (4.10) *is feasible for* $c = 1$.

Proof: One can see that for $c = 1$ the obtained values for $\gamma(c), K(c), \mathcal{P}_{K(c)}$ are identical to the values $\gamma, K, \mathcal{P}$ obtained with Algorithm 4.3. Since $\bar{x} \subset \mathcal{P} = \mathcal{P}_{K(c)}$ this implies that (4.10b) is satisfied for $c = 1$. □

Lemma 4.1 indicates that an interval reduction method initialized with the interval $[1, \bar{c}], \bar{c} > 1$ will always find a feasible solution to (4.10). Standard methods can find sufficiently accurate solutions ($\sim 10^{-10}$) in 10 to 20 iterations, with each iteration consisting of the computation of $K(c)$ and the corresponding $\mathcal{P}_{K(c)}$.

Theorem 4.1. *Consider the optimal value* $\gamma(c^{\mathrm{o}})$ *of Algorithm 4.3 and the optimal value* γ^{o} *of Algorithm 4.1, then the following property holds:*

$$\gamma(c^{\mathrm{o}}) \leq \gamma^{\mathrm{o}}. \tag{4.12}$$

Proof: Since $c = 1$ is always a feasible solution to (4.10) by virtue of Lemma 4.1 and $\gamma(1) \equiv \gamma^{\mathrm{o}}$ the theorem is trivially proven. □

Theorem 4.2. *The feedback gain* $K(c^{\mathrm{o}})$ *obtained with Algorithm 4.3 robustly asymptotically stabilizes* (3.1)-(3.2) *and satisfies constraints* (2.10), (2.11), (4.1) *for trajectories starting from initial state* $\bar{x}$*.*

Proof: Since $K(c^{\mathrm{o}})$ is essentially calculated using Algorithm 4.1, it robustly asymptotically stabilizes (3.1)-(3.2) with Lyapunov function $V(x) = x^{\mathrm{T}} P(c^{\mathrm{o}})x$, with $P(c^{\mathrm{o}})$ defined in a similar way as $K(c^{\mathrm{o}})$. Since $\bar{x} \in \mathcal{P}(c^{\mathrm{o}})$ and $\mathcal{P}(c^{\mathrm{o}})$ is a robust feasible invariant set with respect to the closed loop system (3.1), (3.2), (4.2) (with $K = K(c^{\mathrm{o}})$) subject to constraints (2.10), (2.11), (4.1), it is also guaranteed by construction that all trajectories starting from $\bar{x}$ satisfy the imposed constraints. □

As is done in [68], it is also possible to apply Algorithms 4.1, 4.2 and 4.3 on-line in a receding-horizon fashion, i.e. applying the algorithm at every time instant $k \in \mathbb{N}$ with $\bar{x} = x(k)$ and applying $u(k) = -Kx(k)$ to the system. We will refer to these algorithms as Algorithms 4.1b, 4.2b and 4.3b.

4.1.5 Example

We consider a numerical example describing a double integrator with polytopic model uncertainty described by

$$A_1 = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix}, \qquad B_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \tag{4.13a}$$

$$A_2 = \begin{bmatrix} 1 & 0.2 \\ 0 & 1 \end{bmatrix}, \qquad B_2 = \begin{bmatrix} 0 \\ 1.5 \end{bmatrix}. \tag{4.13b}$$

The system is subject to constraints (2.10), (2.11), (4.1) defined as

$$A_x = [0.01I;\ -0.1I], \tag{4.14a}$$

$$A_u = [1;\ -2], \tag{4.14b}$$

$$A_{xu} = [0.1\ \ 0\ \ -2]. \tag{4.14c}$$

The control objective (4.3) is defined as

$$Q = I, \qquad R = 0.01, \qquad N = [0.05; 0]. \tag{4.15}$$

Figure 4.1 depicts the resulting invariant sets and trajectories corresponding to controllers computed using Algorithms 4.2 and 4.3. Algorithm 4.2 computes identical controllers for symmetrically positioned initial states, although the imposed constraints are non-symmetrical, which illustrates that it cannot efficiently deal with this setting. The depicted polyhedral invariant sets (dashed) also show that the initial state in some cases lies well within the feasible region, which indicates that the feedback controller will not reach any of the imposed constraints for this initial state. Algorithm 4.3 results in controllers whose feasible region exactly contain the imposed initial states. Table 4.1 indicates that this improved constraint handling leads to more optimal controllers at the expense of an increased computation time.

Figure 4.2 shows trajectories using Algorithms 4.1b and 4.3b. The system behavior was chosen to be alternating between $[A_1\ B_1]$ and $[A_2\ B_2]$. Both algorithms lead to stable behavior and satisfy all imposed constraints, including the mixed state/input constraint. Algorithm 4.3b leads to more complex control behavior and has non-conservative constraint handling, as can be verified in Figure 4.2. This leads to

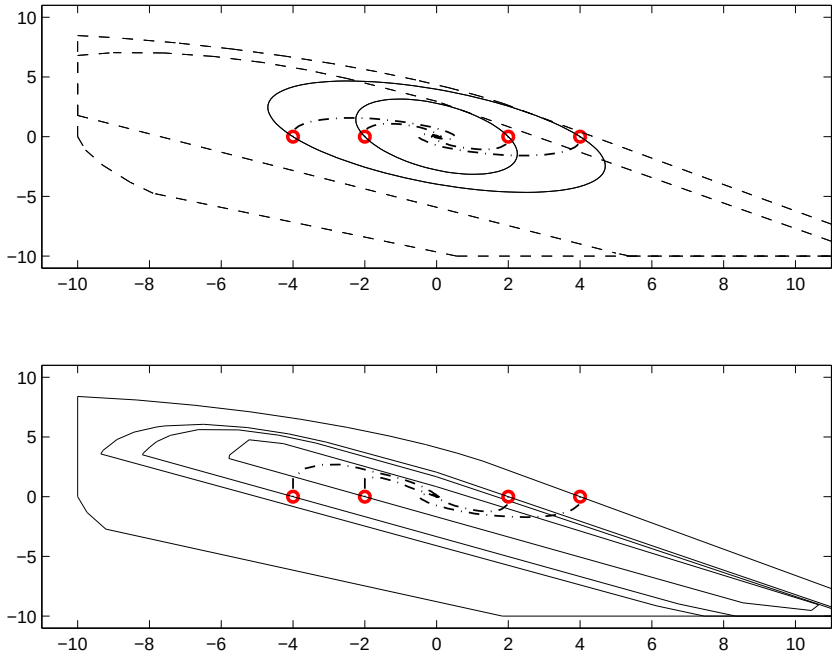


Figure 4.1: Top: Ellipsoidal (solid) and polyhedral (dashed) invariant sets and trajectories (dash-dotted) corresponding to feedback controllers computed using Algorithm 4.2 for different initial state $\bar{x}$ (depicted as circles) for the LPV system defined by (4.13) subject to constraints defined by (4.14) and for cost matrices (4.15). **Bottom:** Polyhedral invariant sets (solid) and trajectories (dash-dotted) corresponding to feedback controllers computed using Algorithm 4.3.

increased optimality, with Algorithm 4.1b resulting in a control cost of 1150.4 for both initial states and Algorithm 4.3b leading to a control cost of 606.3 and 1104.5 for initial state $[-8; 0]$ and $[8; 0]$ respectively.

4.1.6 Conclusions

This section extends the results of [68] in two ways. First the algorithm is extended to also deal with mixed state/input constraints and cost terms. Secondly Algorithm 2.4 is combined with this method to improve the constraint handling in the controller synthesis. The obtained algorithms can be applied either off-line to compute robustly stabilizing linear feedback controllers with guaranteed feasibility or can be applied on-line in a receding horizon fashion.

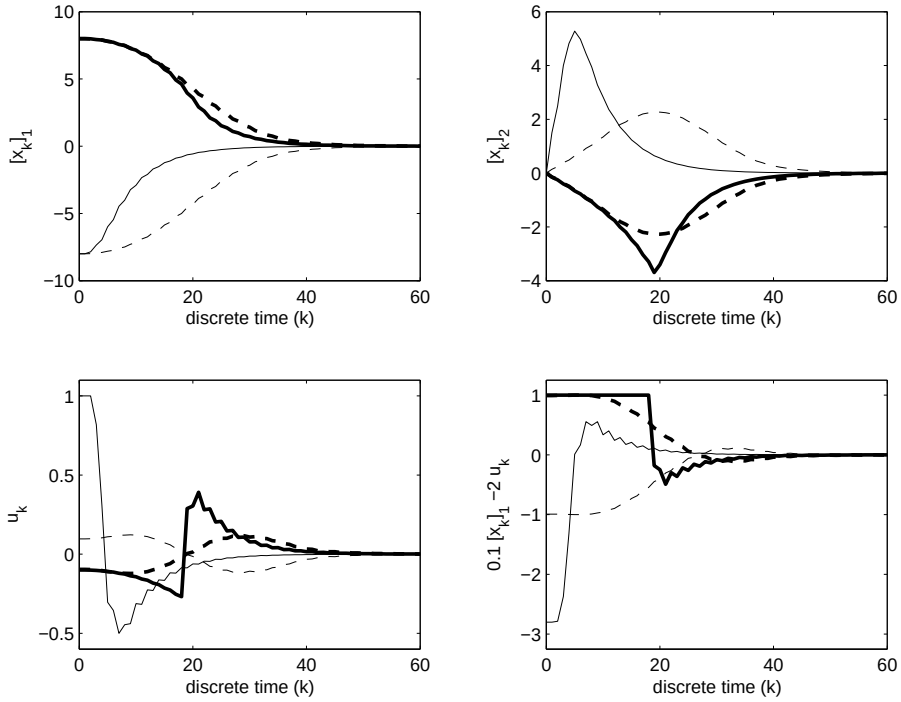


Figure 4.2: Input and state trajectories resulting from Algorithms 4.1b (dashed) and 4.3b (solid) for initial states $[-8;0]$ (thin lines) and $[8;0]$ (thick lines). The same system, constraints and cost matrices as in Figure 4.1 were used. The lower right subfigure illustrates that the imposed mixed state/input constraint is successfully taken into account.

$\bar{\mathbf{x}}$	$[-\mathbf{4};\mathbf{0}]$	$[-\mathbf{2};\mathbf{0}]$	$[\mathbf{2};\mathbf{0}]$	$[\mathbf{4};\mathbf{0}]$	$\mathbf{T}$
γ **for Algorithm 4.2**	282.78	58.70	58.70	282.78	0.41s
γ **for Algorithm 4.3**	207.70	48.41	53.82	270.58	101s

Table 4.1: Upper bounds for the total control cost (4.3) and for different values of $\bar{x}$ and average computation time using two different algorithms for computing the feedback controller. The same system, constraints and cost matrices as in Figure 4.1 were used.

4.2 Interpolation based robust MPC

4.2.1 Introduction

In this section a specific class of MPC algorithms with guaranteed stability is discussed, that is able to obtain large feasible regions in a markedly different way than the quasi-infinite horizon MPC Algorithms 1.2 and 3.2. With this latter class of quasi-infinite horizon algorithms, the feasible region can be enlarged by means of increasing the horizon length. However, when controlling systems with slow dynamics, or when relatively strict input constraints are applied, a large increase in horizon length might be needed in order to obtain a moderate enlargement of the feasible region. As a result, the maximum obtainable feasible region under given computational complexity constraints might be relatively small in these circumstances. A large feasible region can also be obtained by choosing a sub-optimal terminal controller with a large corresponding invariant set. However, local optimality is sacrificed in this case.

The class of interpolation-based MPC algorithms is able to achieve large feasible regions while maintaining local optimality and a relatively low computational complexity. Interpolation-based algorithms do not explicitly make use of a finite horizon, but rather interpolate between trajectories resulting from a priori fixed linear feedback laws. One of these linear feedback laws is typically chosen to be locally optimal, which then guarantees local optimality of the resulting interpolation-based control law. The other feedback laws are typically chosen such that the corresponding invariant sets are large in one or more dimensions of state space. In this way the feasible region of the resulting control law is guaranteed to be large, since it can be proven to be equal to the convex hull of the invariant sets corresponding to the different linear feedback laws.

General interpolation in MPC was initially introduced in [3] for LPV systems and made use of ellipsoidal invariant sets. [122] investigated the LTI case and made use of polyhedral invariant sets. However, the use of polyhedral invariant sets for the LPV case had not been investigated until recently. Based on the results presented in Chapter 2, this section describes this extension. This leads to improved constraint handling, reduced computational complexity – especially for low-dimensional systems – and a guaranteed enlargement of the feasible region. This contribution was discussed in [99].

A second contribution of this section is the extension of the feasible region of interpolation-based MPC algorithms beyond the convex hull of the invariant sets corresponding to the different linear feedback laws. This extension is applicable any general interpolation based algorithm regardless of the type of invariant sets that is used, but in the latter case of polyhedral sets, the enlargement of the feasible region typically is more significant. The disadvantage is the potentially large increase in the number of constraints, compared to the standard algorithms for general interpolation. This contribution was introduced in [118, 120].

First we introduce the concept of general interpolation for LTI systems. Then, the concept is generalized to the robust control setting, after which the use of polyhedral invariant sets is introduced. Finally, a new method is described to further enlarge the feasible region of interpolation based MPC.

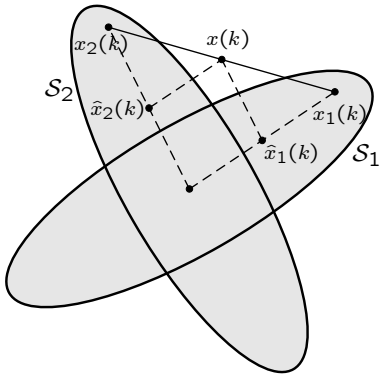


Figure 4.3: Depiction of the state decomposition used in general interpolation based MPC for the case $n_x = 2$ and $n = 2$.

4.2.2 General interpolation for LTI systems

In this section we consider the LTI systems of the form (1.1). The aim is to steer the system state towards the origin without violating state and input constraints (2.10)-(2.11). General interpolation combines large feasible regions and local optimality by interpolating the behavior of different linear feedback laws:

$$u(k) = -K_i x(k), \qquad i = 1, \dots, n, \tag{4.16}$$

with $2 \leq n \in \mathbb{N}$ denoting the number of linear control laws between which the interpolation is performed. The controllers are constructed off-line during the design phase together with feasible invariant sets $\mathcal{S}_1, \dots, \mathcal{S}_n$ for the corresponding closed-loop systems $(A - BK_1), \dots, (A - BK_n)$. At every time instant k, the current system state $x(k)$ is written as a convex combination of n vectors $x_1(k), \dots, x_n(k) \in \mathbb{R}^{n_x}$:

$$x(k) = \sum_{i=1}^{n} \lambda_i(k) x_i(k), \qquad \lambda_{1\dots n}(k) \geq 0, \qquad \sum_{i=1}^{n} \lambda_i(k) = 1, \tag{4.17}$$

where every vector x_i has to lie within the corresponding invariant set: $x_i \in \mathcal{S}_i, i = 1, \dots, n$. By introducing variables $\hat{x}_i(k) \triangleq \lambda_i(k) x_i(k), i = 1, \dots, n$, this decomposition can also be written as

$$x(k) = \sum_{i=1}^{n} \hat{x}_i(k), \quad \text{with} \quad \begin{cases} \sum_{i=1}^{n} \lambda_i(k) = 1, \ \lambda_i \geq 0, \\ \hat{x}_i(k) \in \lambda_i(k) \mathcal{S}_i. \end{cases} \tag{4.18}$$

This decomposition is depicted in Figure 4.3. Note that this decomposition can only be carried out if $x(k) \in \mathrm{Co}\{\mathcal{S}_1, \dots, \mathcal{S}_n\}$. Based on this state decomposition a control action is calculated as follows:

$$u(k) = -\sum_{i=1}^{n} K_i \hat{x}_i(k). \tag{4.19}$$

This control action is by construction feasible with respect to $\mathcal{U}$. Due to the fact that $\hat{x}_i(k) \in \lambda_i(k)\mathcal{S}_i$, one can see that $-K_i\hat{x}_i(k) \in \lambda_i(k)\mathcal{U}$ and therefore $u(k) \in \mathcal{U}$ is satisfied. One can also see that this control action results in the following next state $x(k+1)$:

$$x(k+1) = Ax(k) + Bu(k), \tag{4.20a}$$

$$= A\sum_{i=1}^{n} \hat{x}_i(k) - B\sum_{i=1}^{n} K_i\hat{x}_i(k), \tag{4.20b}$$

$$= \sum_{i=1}^{n} (A - BK_i)\hat{x}_i(k). \tag{4.20c}$$

At time $k+1$ one can therefore make a decomposition into vectors $\hat{x}_i(k+1) = (A - BK_i)\hat{x}_i(k)$ with coefficients $\lambda_i(k+1) = \lambda_i(k), i = 1, \ldots, n$. Because the sets $\mathcal{S}_i$ are invariant with respect to their respective closed-loop systems, one can see that $(A - BK_i)\hat{x}_i(k) \in \lambda_i(k+1)\mathcal{S}_i$ and that this decomposition at time $k+1$ is indeed valid. By applying this reasoning recursively the following input and state sequences are obtained:

$$u(k+i|k) = -\sum_{j=1}^{n} K_j(A - BK_j)^i\hat{x}_j(k), \qquad i \in \mathbb{N}, \tag{4.21a}$$

$$x(k+i|k) = \sum_{j=1}^{n} (A - BK_j)^i\hat{x}_j(k), \qquad i \in \mathbb{N}. \tag{4.21b}$$

The control cost corresponding to these input and state sequences can now be calculated as a function of the variables $\hat{x}_i(k), i = 1, \ldots, n$ by applying standard Lyapunov theory. By introducing the stacked vector $\tilde{x}(k) = [\hat{x}_1(k); \ldots; \hat{x}_n(k)]$ an upper bound $\tilde{x}^{\mathrm{T}}(k)P\tilde{x}(k)$ to this cost can be calculated as

$$P \succeq \Psi^{\mathrm{T}}P\Psi + \Gamma_u^{\mathrm{T}}R\Gamma_u + \Gamma_x^{\mathrm{T}}Q\Gamma_x, \tag{4.22}$$

with $P \in \mathbb{S}_{++}^{n.n_x}$ and $\Psi = \mathrm{diag}((A - BK_1), \ldots, (A - BK_n))$, $\Gamma_x = [I, \ldots, I]$, $\Gamma_u = [K_1, \ldots, K_n]$. One can calculate P by solving the SDP

$$\min_{P \in \mathbb{S}_{++}^{n.n_x}} \ \mathrm{tr}(P), \quad \text{subject to (4.22).} \tag{4.23}$$

This optimization problem is an SDP and can hence be solved efficiently. In the LTI case this optimization yields an exact expression for the expected control cost, and the condition (4.22) will be satisfied with equality.

General interpolation based MPC consists of calculating an optimal state decomposition at every time instant and applying the corresponding control action to the system. This can be formalized as follows:

Algorithm 4.4 (**MPC using general interpolation (GIMPC)).** *Given a system* (1.1)*, constraints* (2.10)-(2.11)*, cost weighting matrices* $Q \in \mathbb{S}_{++}^{n_x}, R \in \mathbb{S}_{++}^{n_u}$*, controllers* K_i *and invariant sets* $\mathcal{S}_i$*, perform the following steps.*

Off-line: *Calculate P by solving* (4.23)*.*
On-line: *At every time instant k, given the current state $x(k)$, perform the following steps*

- *Solve the optimization problem*

$$\min_{\hat{x}_{1\dots n}(k),\lambda_{1\dots n}(k)} \tilde{x}^{\mathrm{T}}(k)P\tilde{x}(k), \quad \textit{subject to } (4.18). \tag{4.24}$$

  in order to obtain an optimal state decomposition.

- *Apply the input* $u(k) = -\sum_{i=1}^{n} K_i\hat{x}_i(k)$.

Lemma 4.2 (**Recursive feasibility and asymptotic stability).** *Algorithm 4.4 is recursively feasible and is asymptotically stabilizing.*

Proof: The proof is straightforward based on the above arguments and is only sketched briefly. Calculation (4.20) suggests a possible decomposition at time $k+1$ based on the decomposition at time k. Based on invariance and feasibility of the sets $\mathcal{S}_i$, this suggested decomposition can be verified to be feasible which proves recursive feasibility. This feasible solution results in a reduced value of the cost function of the on-line optimization problem, which then shows asymptotic stability. □

This methodology was initially introduced in [3] for the LPV case, after which [122] described the LTI case together with several variants of the method that aim to further alleviate the computational complexity.

4.2.3 General interpolation for LPV systems

General interpolation can be extended to the LPV case in a straightforward way. In this section the main differences with the LTI case are highlighted. For more details we refer to [3] and to Section 4.2.4, where the specific case of interpolation based MPC using polyhedral invariant sets is discussed. A more general overview of interpolation based MPC algorithms can be found in [124,125]. In the LPV case a state decomposition is performed in the same way as described by (4.18). Due to the model uncertainty the input and state sequences induced by this decomposition now become:

$$u(k+i|k) = -\sum_{j=1}^{n} K_j \prod_{p=1}^{i} \Phi_j(k+i-p)\hat{x}_j(k), \qquad i \in \mathbb{N}, \tag{4.25a}$$

$$x(k+i|k) = \sum_{j=1}^{n} \prod_{p=1}^{i} \Phi_j(k+i-p)\hat{x}_j(k), \qquad i \in \mathbb{N}, \tag{4.25b}$$

with $\Phi_j(k+i) = A(k+i) - B(k+i)K_j, i \in \mathbb{N}, j = 1,\dots,n$. One can see that not only the state but also the input sequence depends on the future values of $A(k), B(k)$, which indicates that within-horizon feedback is present in this type of MPC. Also note that the input and state sequence under consideration cover an infinite horizon and hence the notion of a terminal cost and constraint is not applicable in this setting. In order to obtain recursive feasibility the feedback controllers $K_{1\dots n}$ need to

be robustly stabilizing and the corresponding invariant sets $\mathcal{S}_{1\dots n}$ need to be robustly positive invariant:

$$(A_j - B_j K_i)x \in \mathcal{S}_i, \qquad \forall x \in \mathcal{S}_i, \quad j = 1, \dots, r, \quad i = 1, \dots, n. \tag{4.26}$$

In the LTI case these different controllers $K_{1\dots n}$ can be constructed as LQR controllers calculated for different cost weighting matrices. In the LPV case the construction of these controllers is somewhat more complicated, but still tractable, since here the methods described in Section 4.1 can be used. This can be done by choosing n different values of $\bar{x}$ resulting in n different feedback gains K_i. This already gives an indication of how the improvements obtained there influence the performance of other MPC algorithms.

An upper bound $\tilde{x}^{\mathrm{T}}(k)P\tilde{x}(k)$ to the corresponding worst-case control cost can be obtained by solving the following modified optimization problem:

$$\min_{P \in \mathbb{S}^{n \cdot n_x}_{++}} \quad \mathrm{Tr}(P), \tag{4.27a}$$

$$\text{subject to} \quad P \succeq \Psi_i^{\mathrm{T}} P \Psi_i + \Gamma_u^{\mathrm{T}} R \Gamma_u + \Gamma_x^{\mathrm{T}} Q \Gamma_x, \qquad i = 1, \dots, r, \tag{4.27b}$$

Note that this optimization problem is still an SDP, similar to the LTI case, but with an increased number of LMI constraints. Typically, not all of these constraints can be satisfied with equality and hence the obtained upper bound $\tilde{x}^{\mathrm{T}}(k)P\tilde{x}(k)$ will be an over-estimate of the real worst-case control cost. This is the sacrifice that has to be made in order to obtain a quadratic function as an upper bound to the control cost and hence obtain a more efficient on-line optimization problem.

Although the off-line computations become visibly more complex in the LPV case, the on-line optimization problem remains identical to the LTI case. However, a second look also reveals an increase in computational complexity in the on-line optimization problem. This is due to the fact that invariant sets for LPV systems typically are more complex than those for LTI systems. In [3] ellipsoidal invariant sets are used, leading to an SDP formulation for the on-line computation. In [96] it is shown that this optimization problem can also be formulated as an SOCP. In the LTI case [122] polyhedral invariant sets are typically used, which results in a QP.

The next section describes in more detail the case when polyhedral invariant sets are used in the LPV case. This leads to a QP optimization problem, which is significantly easier to solve than the SDP presented in [3] and also leads to a significant improvement in the control performance.

4.2.4 General interpolation using polyhedral invariant sets

This section restates Algorithm 4.4 for the LPV case combined with the use of polyhedral invariant sets. A detailed proof of recursive feasibility and asymptotic stability are given. These results were initially described in [99].

Algorithm 4.5 (Robust GIMPC using polyhedral sets ($\mathcal{P}$-GIMPC)). *Given a system* (1.1)*, constraints* (2.10)-(2.11)*, cost weighting matrices* $Q \in \mathbb{S}^{n_x}_{++}, R \in \mathbb{S}^{n_u}_{++}$*, controllers* K_i *and polyhedral invariant sets* $\mathcal{S}_i = \{x | A_{\mathcal{S}_i} x \leq \mathbf{1}\}$*, solve on-line at*

each time instant k, given the current state $x(k)$, the following problem:

$$\min_{\hat{x}_{1\dots n}(k),\lambda_{1\dots n}(k)} \tilde{x}^{\mathrm{T}}(k)P\tilde{x}(k), \tag{4.28a}$$

$$\text{subject to} \quad x(k) = \sum_{i=1}^{n} \hat{x}_i(k), \tag{4.28b}$$

$$A_{\mathcal{S}_i}\hat{x}_i(k) \leq \lambda_i(k)\mathbf{1}, \qquad i = 1,\dots,n, \tag{4.28c}$$

$$\sum_{i=1}^{n} \lambda_i(k) = 1, \tag{4.28d}$$

$$\lambda_i(k) \geq 0, \qquad i = 1,\dots,n, \tag{4.28e}$$

and implement input $u(k) = -\sum_{i=1}^{n} K_i\hat{x}_i(k)$.

It is obvious that this optimization problem is a QP and can hence be solved efficiently.

Lemma 4.3. *Algorithm 4.5 guarantees robust satisfaction of* (2.10)-(2.11) *and is recursively feasible and asymptotically stable for all initial states $x(0) \in \overline{\mathcal{S}} \triangleq \mathrm{Co}\{\mathcal{S}_1,\dots,\mathcal{S}_n\}$.*

Proof: It is clear from (4.18) that (4.28) is feasible for all $x \in \overline{\mathcal{S}}$. Given the current state $x(k)$, components $\hat{x}_i(k)$ and factors $\lambda_i(k)$, it is possible to calculate the next state to be $x(k+1) = \sum_{i=1}^{n} \Phi_i(k)\hat{x}_i(k)$. Since the components $x_i(k)$ lie in their respective invariant sets $\mathcal{S}_i$, this will also be the case for the components $x_i(k+1) = \Phi_i(k)x_i(k)$, which shows that $x(k+1)$ will also lie within $\overline{\mathcal{S}}$. By recursively applying this argument it is proven that $x(k+i) \in \overline{\mathcal{S}}, i = 1,\dots,\infty$. Since all $\mathcal{S}_i$ are subsets of $\mathcal{X}$ and $\mathcal{X}$ is convex, $\overline{\mathcal{S}}$ will also be a subset of $\mathcal{X}$, which then proves robust satisfaction of the state constraints. Furthermore, since $\hat{x}_i(k) \in \lambda_i(k)\mathcal{S}_i$, it is clear that $\hat{u}_i(k) \triangleq -K_i\hat{x}_i(k) \in \lambda_i(k)\mathcal{U}$. Therefore (since $\mathcal{U}$ is a convex set) $u(k) \triangleq \sum_{i=1}^{n} \hat{u}_i(k) \in \mathcal{U}$, which proves robust satisfaction of the input constraints.

Asymptotic stability can be proven by considering components $\hat{x}_i(k)$, which are shown above to provide a feasible candidate decomposition $\hat{x}_i(k+1) = \Phi_i(k)\hat{x}_i(k)$ for time $k+1$. It is now easy to see, based on satisfaction of (4.27b) that this candidate decomposition already provides a lower value of the cost function of (4.28) than the optimal cost value at time k, which proves that the optimal value of the cost at time $k+1$ will also be lower than the optimal value at time k. This consequently proves that the optimal value of the cost function of (4.28) acts as a Lyapunov function of the closed-loop system, which proves asymptotic stability. □

In the sequel we refer to algorithm 4.5 as $\mathcal{P}$-GIMPC, whereas the same algorithm using ellipsoidal invariant sets will be referred to as $\mathcal{E}$-GIMPC. The former algorithm has several advantages compared to the latter. First of all, it has an enlarged feasibility region, since the individual invariant sets are larger than their ellipsoidal counterparts used in $\mathcal{E}$-GIMPC. Furthermore, the polyhedral invariant sets can efficiently cope with non-symmetrical constraints, which ellipsoids cannot. Due to the invariant sets being larger, less conservative satisfaction of the imposed input and state constraints can be expected, potentially leading to a reduction in control cost. Finally, the algorithm

is formulated as a QP, which is significantly less expensive to solve than the SDP formulation of $\mathcal{E}$-GIMPC. Section 4.2.7 clearly illustrates these advantages.

4.2.5 Improved general interpolation

This section discusses a further improvement of the GIMPC algorithm discussed in the previous section. Section 4.2.5.1 first shows that the constraint handling in GIMPC algorithms can still be conservative in many cases due to the fact that constraint handling is only done explicitly for each linear control law separately. Section 4.2.5.2 then shows how improved constraint handling can be obtained by constructing a mutual invariant set for an augmented autonomous system describing the dynamic behavior of the state decomposition vectors $\hat{x}_i(k)$. The improved constraint handling is applicable to both $\mathcal{P}$-GIMPC and $\mathcal{E}$-GIMPC leading to respectively $\mathcal{P}$-GIMPC2 and $\mathcal{E}$-GIMPC2. These results are published in [118, 120, 121].

4.2.5.1 Conservative constraint handling in GIMPC

This section explains a shortcoming in GIMPC that can lead to conservative constraint handling. For reasons of simplicity only the LTI case is considered, but the same arguments can be formulated for the LPV case.

Constraint handling in GIMPC is done by means of invariant sets $\mathcal{S}_j$ corresponding to the different linear control laws K_j. By imposing that $\hat{x}_j(k) \in \lambda_j(k)\mathcal{S}_j$ it is guaranteed that the corresponding input and state sequence components lie within their correspondingly scaled state and input constraint sets:

$$\hat{x}_j(k+i|k) \triangleq (A-BK_j)^i \hat{x}_j(k) \quad \in \lambda_j(k)\mathcal{X}, \quad i \in \mathbb{N}, j = 1, \ldots, n, \quad \text{(4.29a)}$$

$$\hat{u}_j(k+i|k) \triangleq -K_j(A-BK_j)^i \hat{x}_j(k) \in \lambda_j(k)\mathcal{U}, \quad i \in \mathbb{N}, j = 1, \ldots, n. \quad \text{(4.29b)}$$

As a result, the corresponding state and input sequence will also satisfy the input and state constraints:

$$x(k+i|k) = \sum_{j=1}^{n} \hat{x}_j(k+i|k) \in \mathcal{X}, \qquad i \in \mathbb{N}, \quad \text{(4.30a)}$$

$$u(k+i|k) = \sum_{j=1}^{n} \hat{u}_j(k+i|k) \in \mathcal{U}, \qquad i \in \mathbb{N}. \quad \text{(4.30b)}$$

If a state component vector $\hat{x}_j(k)$ lies close to the edge of its correspondingly scaled invariant set $\lambda_j(k)\mathcal{S}_j$ this will mean that either (4.29a) or (4.29b) will only be marginally satisfied for one or more values of i. Consequently if $x(k)$ lies close to the edge of $\overline{\mathcal{S}}$, all state component vectors $\hat{x}_j(k)$ will lie close to the edge of their correspondingly scaled invariant set $\lambda_j(k)\mathcal{S}_j$ and hence for each $j = 1, \ldots, n$ there will exist one or more values of i for which either (4.29a) or (4.29b) will only be marginally satisfied. However, this does not necessarily imply that the same will hold for either (4.30a) or (4.30b). Possible reasons for this are the following:

- (4.29a) or (4.29b) might have extremal values with different signs for different values of j, whose effect will be annihilated by summing these different sequences (over all j-values) into $x(k+i|k)$ and $u(k+i|k)$.
- (4.29a) or (4.29b) might achieve their extremal values for different values of i depending on j. (e.g., $\hat{x}_1(k+i|k)$ might be maximal for $i = 2$, whereas $\hat{x}_2(k+i|k)$ might be maximal for $i = 5$)
- Possibly some values of j might only have state component sequences that lie close to the constraints, whereas other values of j might only have input sequences that lie close to the constraints. (e.g., for $j = 1$ only $\hat{x}_1(k+i|k)$ might reach values close to the constraints, whereas for $j = 2$ only $\hat{u}_2(k+i|k)$ might reach values close to the constraints)

For any of these reasons the resulting state and input sequences (4.30a), (4.30b) might lie well within the imposed constraints even for values of $x(k)$ lying close to edge of $\overline{S}$. This implies that there exist states $x(k)$ outside of $\overline{\mathcal{S}}$ for which a state decomposition exists that results in a feasible corresponding input and state sequence. This effect is illustrated in Section 4.2.7. The aim of the next section is to improve the GIMPC algorithm to better cope with these effects.

4.2.5.2 Improved constraint handling (GIMPC2)

This section improves the constraint handling of the GIMPC algorithm by allowing all possible decompositions (also those with $\hat{x}_j \notin \lambda_j(k)\mathcal{S}_j$ for some j) that lead to feasible input and state sequences $u(k+i|k), x(k+i|k), i \in \mathbb{N}$. This is done by constructing an augmented system describing the dynamics of the considered state decomposition components $\hat{x}_j(k+i|k), j = 1,\ldots,n, i \in \mathbb{N}$. Two different methods are possible, based on the definition state vector $x_{\mathrm{aug}}(k+i|k)$ of this augmented system. This vector can either be defined as $x_{\mathrm{aug}}(i) \triangleq [\hat{x}_1(k+i|k);\ldots;\hat{x}_n(k+i|k)]$ or as $x_{\mathrm{aug}}(i) \triangleq [x(k+i|k);\hat{x}_1(k+i|k);\ldots;\hat{x}_{n-1}(k+i|k)]$. We first describe the former method. Afterwards the latter choice, which has some advantages over the former, is described.

4.2.5.2.1 **Method 1.** If we define $x_{\mathrm{aug}}(i) \triangleq [\hat{x}_1(k+i|k);\ldots;\hat{x}_n(k+i|k)]$ the state sequence (4.25b) can be described by the following autonomous system:

$$x_{\mathrm{aug}}(i+1) = \begin{bmatrix} \Phi_1(k+i) & & \\ & \ddots & \\ & & \Phi_n(k+i) \end{bmatrix} x_{\mathrm{aug}}(i), \qquad i \in \mathbb{N}, \tag{4.31}$$

where again $\Phi_j(k+i) \triangleq A(k+i) - B(k+i)K_j$. The state and input sequences can be easily expressed in terms of $x_{\mathrm{aug}}(i)$:

$$x(k+i|k) \equiv [I \ \ldots \ I]x_{\mathrm{aug}}(i), \qquad i \in \mathbb{N}, \tag{4.32}$$

$$u(k+i|k) \equiv -[K_1 \ \ldots \ K_n]x_{\mathrm{aug}}(i), \qquad i \in \mathbb{N}. \tag{4.33}$$

We now want to characterize all state decompositions defined by $x_{\mathrm{aug}}(0)$, for which the corresponding state and input sequences satisfy the constraints. In other words,

we want to characterize all augmented states $x_{\text{aug}}(0)$, that guarantee that the following constraints are satisfied:

$$[I, \ldots, I]x_{\text{aug}}(i) \in \mathcal{X}, \qquad \forall [A(k+i)\ B(k+i)] \in \Omega, \forall i \in \mathbb{N}, \tag{4.34}$$
$$-[K_1, \ldots, K_n]x_{\text{aug}}(i) \in \mathcal{U}, \qquad \forall [A(k+i)\ B(k+i)] \in \Omega, \forall i \in \mathbb{N}. \tag{4.35}$$

This can be done by calculating the MAS $\mathcal{S}_{\text{aug}}$ for the following autonomous LPV system:

$$x_{\text{aug}}(i+1) = \Psi_{\text{aug}}(i)x_{\text{aug}}(i), \qquad i \in \mathbb{N}, \tag{4.36}$$

where $\Psi_{\text{aug}}(i) \in \Omega_{\text{aug}}, \forall i \in \mathbb{N}$, with Ω_{aug} defined as

$$\Omega_{\text{aug}} \triangleq \text{Co}\{\Psi_{\text{aug},1}, \ldots, \Psi_{\text{aug},l}\}, \tag{4.37}$$
$$\Psi_{\text{aug},j} \triangleq \text{diag}(A_j - B_jK_1, \ldots, A_j - B_jK_n), \qquad j = 1, \ldots, r, \tag{4.38}$$

subject to constraints

$$[A_x\ \ldots\ A_x]x_{\text{aug}}(i) \leq \mathbf{1}, \qquad \forall i \in \mathbb{N}, \tag{4.39}$$
$$[-A_uK_1\ \ldots\ -A_uK_n]x_{\text{aug}}(i) \leq \mathbf{1}, \qquad \forall i \in \mathbb{N}. \tag{4.40}$$

It is clear that if $x_{\text{aug}}(0) \in \mathcal{S}_{\text{aug}}$, constraints (4.34),(4.35) are automatically satisfied. The following on-line optimization problem can now be formulated:

$$\min_{\hat{x}_{1\ldots n}(k)} \quad \tilde{x}^{\text{T}}(k)P\tilde{x}(k), \tag{4.41}$$
$$\text{subject to} \quad x(k) = \sum_{i=1}^{n} \hat{x}_i(k), \tag{4.42}$$
$$\tilde{x}^{\text{T}}(k) \in \mathcal{S}_{\text{aug}}. \tag{4.43}$$

Compared to GIMPC this optimization problem has a reduced number of optimization variables and if $\mathcal{S}_{\text{aug}}$ is polyhedral, the resulting optimization problem is a QP. However, this formulation has two disadvantages. First of all there are still equality constraints present in this formulation in order to make sure that the different state component vectors $\hat{x}_{1\ldots n}(k)$ sum up to $x(k)$. Second of all, finding an explicit description of the feasible region (in terms of $x(k)$), requires the computation of an oblique projection of $\mathcal{S}_{\text{aug}}$. These two disadvantages are eliminated in the second method, at the cost of a slightly more complicated construction of $\Psi_{\text{aug}}(k)$.

4.2.5.2.2 **Method 2.** If we define $x_{\text{aug}}(i) \triangleq [x(k+i|k); \hat{x}_1(k+i|k); \ldots; \hat{x}_{n-1}(k+i|k)]$, the following autonomous LPV system is obtained:

$$x_{\text{aug}}(i+1) = \Psi_{\text{aug}}(i)x_{\text{aug}}(i), \qquad i \in \mathbb{N}, \tag{4.44a}$$

where $\Psi_{\text{aug}}(i) \in \Omega_{\text{aug}}, \forall i \in \mathbb{N}$, with Ω_{aug} defined as

$$\Omega_{\text{aug}} \triangleq \text{Co}\{\Psi_{\text{aug},1}, \ldots, \Psi_{\text{aug},r}\}, \tag{4.44b}$$

with

$$\Psi_{\mathrm{aug},j} \triangleq \begin{bmatrix} A_j - B_j K_n & B_j(K_n - K_1) & \cdots & B_j(K_n - K_{n-1}) \\ 0 & A_j - B_j K_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_j - B_j K_{n-1} \end{bmatrix},$$
$$j = 1, \ldots, r, \quad (4.44c)$$

subject to constraints

$$A_x \Gamma_x x_{\mathrm{aug}}(i) \leq \mathbf{1}, \qquad \forall i \in \mathbb{N}, \quad (4.45a)$$
$$A_u \Gamma_u x_{\mathrm{aug}}(i) \leq \mathbf{1}, \qquad \forall i \in \mathbb{N}, \quad (4.45b)$$

with $\Gamma_x = [I\ 0\ \ldots\ 0], \Gamma_u = [-K_n \quad (K_n - K_1)\ \ldots\ (K_n - K_{n-1})]$. Similar to method 1, characterization of all state decompositions that lead to feasible input and state sequences can be done by calculating the MAS $\mathcal{S}_{\mathrm{aug}}$ for system (4.44)-(4.45) and demanding that $x_{\mathrm{aug}}(0) \in \mathcal{S}_{\mathrm{aug}}$.

An upper bound to the worst-case control cost in terms of the augmented state vector $x_{\mathrm{aug}}(0)$ can now be obtained as $x_{\mathrm{aug}}(0)^{\mathrm{T}} P' x_{\mathrm{aug}}(0)$ by solving the following optimization problem:

$$\min_{P' \in \mathbb{S}_{++}^{n.n_x}} \quad \mathrm{Tr}(P'), \quad (4.46a)$$
$$\text{subject to} \quad P' \succeq \Psi_{\mathrm{aug},j}^{\mathrm{T}} P' \Psi_{\mathrm{aug},j} + \Gamma_u^{\mathrm{T}} R \Gamma_u + \Gamma_x^{\mathrm{T}} Q \Gamma_x, \quad j = 1, \ldots, r, \quad (4.46b)$$

with $\Psi_{\mathrm{aug},j}$ defined as in (4.44). It should be noted that P' can also be obtained by means of simple algebraic manipulations of the solution P of (4.27). We can now formulate the following improved improved interpolation based MPC algorithm:

Algorithm 4.6 (Improved general interpolation (GIMPC2)). *Given a system* (3.1), (3.2), *constraints* (2.10)-(2.11), *cost weighting matrices Q, R and robustly stabilizing controllers $K_{1\ldots n}$, perform the following steps:*
Off-line:

- *Construct the augmented system* (4.44)-(4.45) *and calculate a feasible positive invariant set $\mathcal{S}_{\mathrm{aug}}$ for this system.*
- *Calculate P' by solving* (4.46).

On-line:
At every time instant k, given the current state $x(k)$, perform the following steps:

- *Solve the optimization problem*

$$\min_{\hat{x}_{1\ldots n-1}(k)} \ x_{\mathrm{aug}}^{\mathrm{T}}(0) P' x_{\mathrm{aug}}(0), \quad \textit{subject to} \quad x_{\mathrm{aug}}(0) \in \mathcal{S}_{\mathrm{aug}}, \quad (4.47)$$

  in order to obtain an optimal state decomposition.

- *Calculate* $\hat{x}_n(k) = x(k) - \sum_{j=1}^{n-1} \hat{x}_j(k)$.
- *Apply the input* $u(k) = -\sum_{j=1}^{n} K_j \hat{x}_j(k)$.

If the set $\mathcal{S}_{\text{aug}}$ is chosen to be a polyhedral invariant set the above algorithm will be referred to as $\mathcal{P} - GIMPC2$, whereas it will be referred to as $\mathcal{E} - GIMPC2$ if an ellipsoidal invariant set is used.

Note that due to the specific choice of $x(i)_{\text{aug}}$ the constraint $x(k) = \sum_{j=1}^{n} \hat{x}_j(k)$ is already eliminated from the formulation and therefore isn't included in the on-line optimization problem. Furthermore, the feasible region of the algorithm can be defined as

$$\mathcal{F} \triangleq \{x | \exists \hat{x}_{1\ldots n-1} : [x; \hat{x}_1; \ldots \hat{x}_{n-1}] \in \mathcal{S}_{\text{aug}}\}. \tag{4.48}$$

The feasible region hence is the projection of $\mathcal{S}_{\text{aug}}$ onto the first n_x dimensions, which can be done using standard techniques like e.g. Fourier-Motzkin elimination [146] in the polyhedral case, or using the Schur complement in the ellipsoidal case. The two disadvantages of method 1 are hence eliminated in method 2.

GIMPC2 maintains the properties of GIMPC regarding recursive feasibility, guaranteed robust constraint satisfaction and asymptotic stability, as is shown by the following lemmas.

Lemma 4.4. *Algorithm 4.6 is recursively feasible.*

Proof: Given a state decomposition $\hat{x}_{1\ldots n}(k)$ at time k satisfying $x_{\text{aug}}(0) \equiv [x(k); \hat{x}_1(k); \ldots;\ \hat{x}_{n-1}(k)] \in \mathcal{S}_{\text{aug}}$, the corresponding state at time $k+1$ is given by

$$x(k+1) = \sum_{j=1}^{n} (A(k) - B(k)K_j)\hat{x}_j(k), \tag{4.49}$$

which suggests a possible decomposition $\hat{x}_j(k+1) = (A(k) - B(k)K_j)\hat{x}_j(k), j = 1, \ldots, n$ at time $k+1$. This candidate decomposition at time $k+1$ can be verified to be feasible, since some straightforward algebraic manipulation yields that

$$\Psi_{\text{aug},j} x_{\text{aug}}(0) = \begin{bmatrix} \sum_{m=1}^{n} (A_j - B_j K_m)\hat{x}_m(k) \\ (A_j - B_j K_1)\hat{x}_1(k) \\ \vdots \\ (A_j - B_j K_{n-1})\hat{x}_{n-1}(k) \end{bmatrix}, \qquad j = 1, \ldots, r. \tag{4.50}$$

Since $\mathcal{S}_{\text{aug}}$ is a positive invariant set with respect to these dynamics, any convex combination of these right hand-sides will still lie inside $\mathcal{S}_{\text{aug}}$. Since $[x(k+1); \hat{x}_1(k+1); \ldots; \hat{x}_{n-1}(k+1)]$ can indeed be written as such a convex combination (because $[A(k)\ B(k)]$ can be written as a convex combination of the $[A_j\ B_j]$) the above proposed state decomposition at time $k+1$ is indeed feasible, which proves recursive feasibility. □

Lemma 4.5. *Algorithm 4.6 guarantees robust satisfaction of* (2.10)-(2.11) *if it is initially feasible.*

Proof: Due to recursive feasibility it is guaranteed that, if the on-line optimization (4.47) is initially feasible, $[x(k); \hat{x}_1(k); \ldots; \hat{x}_{n-1}(k)] \in \mathcal{S}_{\rm aug}, \forall k \in \mathbb{N}$. Because $\mathcal{S}_{\rm aug}$ is also feasible with respect to (4.45), it is guaranteed that

$$A_x x(k) \leq \mathbf{1}, \qquad \forall k \in \mathbb{N}, \tag{4.51}$$

which proves that the state constraints are satisfied $\forall k$, and it is also guaranteed that

$$-A_u \left(K_n x(k) + \sum_{j=1}^{n-1} (K_1 - K_n)\hat{x}_j(k) \right) \leq \mathbf{1}, \qquad \forall k \in \mathbb{N}, \tag{4.52}$$

which is equivalent with

$$A_u u(k) \leq \mathbf{1}, \qquad \forall k \in \mathbb{N}, \tag{4.53}$$

which guarantees that also the input constraint are satisfied $\forall k$, which proves the lemma. □

Lemma 4.6. *Algorithm 4.6 robustly asymptotically stabilizes* (3.1),(3.2) *if it is initially feasible.*

Proof: Condition (4.46b) guarantees that

$$\begin{aligned} &x_{\rm aug}(0)^{\rm T} P' x_{\rm aug}(0) - (\Psi_{{\rm aug},j} x_{\rm aug}(0))^{\rm T} P' (\Psi_{{\rm aug},j} x_{\rm aug}(0)) \geq \\ &\quad x(k)^{\rm T} Q x(k) + u(k)^{\rm T} R u(k), \qquad \forall x_{\rm aug}(0) \neq 0 \in \mathcal{S}_{\rm aug}, j = 1, \ldots, r, \end{aligned} \tag{4.54}$$

and, after making appropriate convex combinations, that

$$\begin{aligned} x_{\rm aug}(0)^{\rm T} P' x_{\rm aug}(0) \geq x_{\rm aug}(1)^{\rm T} P' x_{\rm aug}(1), \\ \forall x_{\rm aug}(0) \neq 0 \in \mathcal{S}_{\rm aug}, \forall [A(k)\ B(k)] \in \Omega. \end{aligned} \tag{4.55}$$

This shows that the feasible solution constructed in Lemma 4.4 already results in a lower value of the objective function of the on-line optimization problem if $x(k) \neq 0$. Hence the optimal value of the optimization problem at time $k+1$ will also be strictly lower than that obtained at time k if $x(k) \neq 0$, which shows that this value is monotonically decreasing as a function of k, which proves asymptotic stability. □

Lemma 4.7. *Algorithm 4.6 has a larger feasible region than $\mathcal{P}$-GIMPC if $\mathcal{S}_{\rm aug}$ is taken to be the MAS for the augmented system.*

Proof: If $\mathcal{S}_{\rm aug}$ is taken to be the MAS, the maximality of the set guarantees that all possible state decompositions that lead to feasible state and input sequences are captured within $\mathcal{S}_{\rm aug}$. Hence, also all decompositions allowed by $\mathcal{P}$-GIMPC are captured, which proves the lemma. □

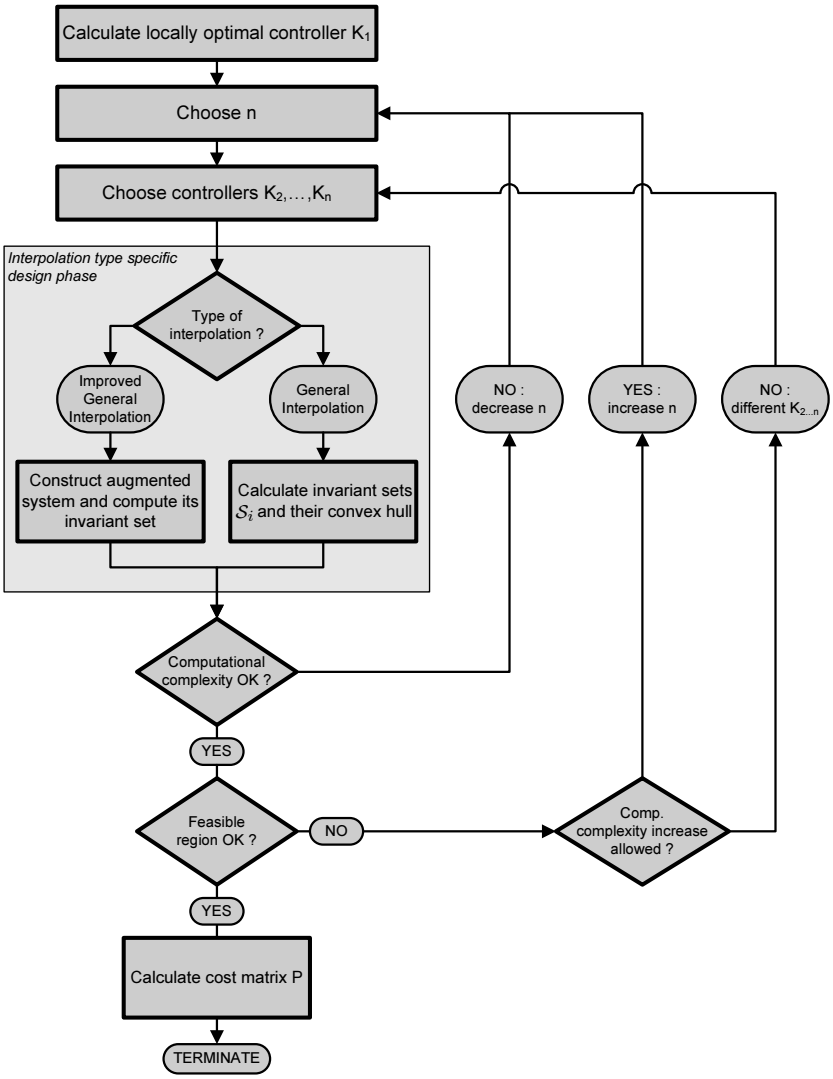


Figure 4.4: Flow chart of the design process of an interpolation based MPC controller. At first sight the design process is largely independent from the type of interpolation that is used, apart from the determination of the constraints to be used in the on-line optimization problem. However, as a result the computational complexity and the feasible region will be interpolation type dependent, influencing the decisions made during the design process. Still, the tuning parameters for modifying both aspects are identical for both types of interpolation.

4.2.6 Controller design

The design process of interpolation based MPC algorithms (both GIMPC and GIMPC2) is somewhat different than that of more standard MPC controllers. This process is depicted as a flow chart in Figure 4.4.

In the following subsections the 4 most important design parameters will be discussed: choice of n, choice of $K_{1\ldots n}$, choice of the invariant set type and choice of the interpolation type (GIMPC vs. GIMPC2).

4.2.6.1 Number of controllers n

The choice of n influences two important properties of the resulting controller: the feasible region and the computational complexity.

Larger values of n allow interpolation between a larger number of control laws and can therefore lead to larger feasible regions. Especially for higher-dimensional systems (large n_x), it might be necessary to choose larger values of n (proportional to n_x) in order to obtain a feasible region that is large in all dimensions.

On the other hand larger values of n also lead to more complex on-line optimization problems as is shown in Table 4.2. The computational complexity also depends on the type of invariant sets that is used, as well as the type of interpolation, as is explained in the following subsections.

4.2.6.2 Choice of feedback laws $K_{1\ldots n}$

Since the aim is to obtain an MPC controller with local optimality, at least one of the controllers should be chosen as the locally optimal controller. This is also depicted in Figure 4.4. In the LTI case this can be done by calculating the LQR controller for the given system, in the LPV case, Algorithm 4.2 with a value of $\bar{x}$ close to 0 can be used.

The other control laws $K_{2\ldots n}$ should be chosen to obtain a large feasible region. One can either chose these controllers as manually detuned versions of the locally optimal

	#variables	**#constraints**	**class**
$\mathcal{P}$**-GIMPC**	$n.(1+n_x)$	n + $\sum_{j=1}^{n} m_j$	QP
$\mathcal{E}$**-GIMPC**	$n.(1+n_x)+1$	n linear constr. $n+1$ LMIs $2n.n_x$ LMI rows	SDP
$\mathcal{P}$**-GIMPC2**	$(n-1).n_x$	m	QP
$\mathcal{E}$**-GIMPC2**	$(n-1).n_x+1$	0 linear constr. 2 LMIs $2n.n_x$ LMI rows	SDP

Table 4.2: Computational complexity of the on-line optimization problems of GIMPC and GIMPC2 in terms of the number of optimization variables, the number of inequality constraints and the optimization class. The variables m_j denote the number of constraint describing the invariant sets $\mathcal{S}_j$, while m denotes the number of constraints describing $\mathcal{S}_{\text{aug}}$.

controller [116] or one can use Algorithm 4.3 with larger values of $\bar{x}$ in order to obtain control laws with feasible regions that are large in specific directions. Figure 4.4 also shows that the choice of these controllers is an iterative process in order to achieve a desired resulting feasible region for the interpolation based controller.

4.2.6.3 Type of invariant sets

The choice of invariant set type (polyhedral vs. ellipsoidal) is largely determined by the computational complexity and the resulting feasible region. Polyhedral invariant sets lead to feasible regions that are guaranteed to be larger than those obtained with their ellipsoidal counterparts and lead to an on-line optimization class (QP) that can be solved efficiently. However, the number of constraints of the optimization problem is less predictable when using polyhedral invariant sets than when usign ellipsoidal invariant sets. In those cases when this number of constraints becomes too large, one can revert to using ellipsoidal invariant sets, otherwise one should stick with polyhedral invariant sets.

4.2.6.4 Choice of interpolation type

As shown by Lemma 4.7 the feasible region of GIMPC2 is larger than that of GIMPC. Simultaneously, GIMPC2 needs a smaller number of on-line optimization variables than GIMPC. Hence, from both points of view GIMPC2 is favorable. The downside, however, is that in the case polyhedral invariant sets are used, the number of constraints m can increase exponentially as a function of $n.n_x$. As a result, when controlling

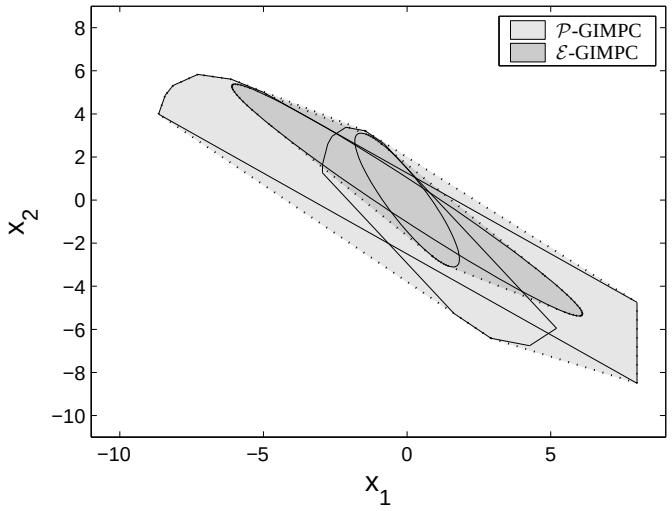


Figure 4.5: Comparison of $\mathcal{E}$-GIMPC and $\mathcal{P}$-GIMPC for system (4.56)-(4.57) using controllers (4.58). The invariant sets are depicted in solid lines, the convex hulls are depicted in dotted lines.

higher-dimensional systems, GIMPC should be favored above GIMPC2. In the other case GIMPC should be favored.

4.2.7 Example

In this section we use a numerical example with the same double integrator dynamics as the one used in Section 4.1.5 in order to illustrate the concepts described in the previous sections. We consider a model of the form (3.1),(3.2) with $r = 2$ and dynamics defined by

$$A_1 = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix}, \qquad B_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \tag{4.56a}$$

$$A_2 = \begin{bmatrix} 1 & 0.2 \\ 0 & 1 \end{bmatrix}, \qquad B_2 = \begin{bmatrix} 0 \\ 1.5 \end{bmatrix}. \tag{4.56b}$$

The system is subject to constraints (2.10), (2.11), with

$$A_x = [I/8;\ -I/10], \tag{4.57a}$$

$$A_u = [1;\ -2]. \tag{4.57b}$$

The control objective is defined by weighting matrices $Q = \mathrm{diag}(1, 0.01), R = 3$. In this section we construct and compare different interpolation based MPC controllers. In all cases $n = 2$ is used with

$$K_1 = [0.4858\ \ 0.3407], \qquad \text{(the LQR-optimal for } [A_1\ B_1]) \tag{4.58a}$$

$$K_2 = [0.3\ \ 0.4]. \tag{4.58b}$$

Both controllers are locally robustly asymptotically stabilizing for the given LPV system (4.56).

We first compare $\mathcal{E}$-GIMPC to $\mathcal{P}$-GIMPC. A depiction of the resulting feasible regions can be found in Figure 4.5. It is clear that $\mathcal{P}$-GIMPC has a significantly larger feasible region than $\mathcal{E}$-GIMPC, which is partly due to the fact that polyhedral invariant sets can efficiently cope with asymmetric constraints.

Figure 4.6 shows trajectories for both controllers starting from initial states $[-5.5; 5]$ and $[5.5; -5]$. $\mathcal{E}$-GIMPC results in symmetrical trajectories for the two different initial states and achieves a control cost of 82.45, while $\mathcal{P}$-GIMPC is able to take advantage of the asymmetrical constraints and achieves a control cost of 81.37 and 78.74 for the two initial states respectively.

Figure 4.7 shows the difference in feasible region between a nominal and robust $\mathcal{P}$-GIMPC controller. The nominal controller is designed for $[A_1\ B_1]$, whereas the robust controller is designed for $\mathrm{Co}\{[A_1\ B_1], [A_2\ B_2]\}$. The right subfigure shows trajectories for both controllers for initial condition $[-8; 4]$, when the real system behavior is taken to be the LTI system defined by $[A_2\ B_2]$. The nominal controller first steers the system outside the feasible region of the robust controller and afterwards even outside its own feasible region, after which an infeasibility occurs (indicated with a circle). As expected, the robust controller steers the system to the origin. This clearly indicates that model mismatch can cause unwanted control behavior when using nominal controllers, which can be avoided when including robustness measures in the controller design.

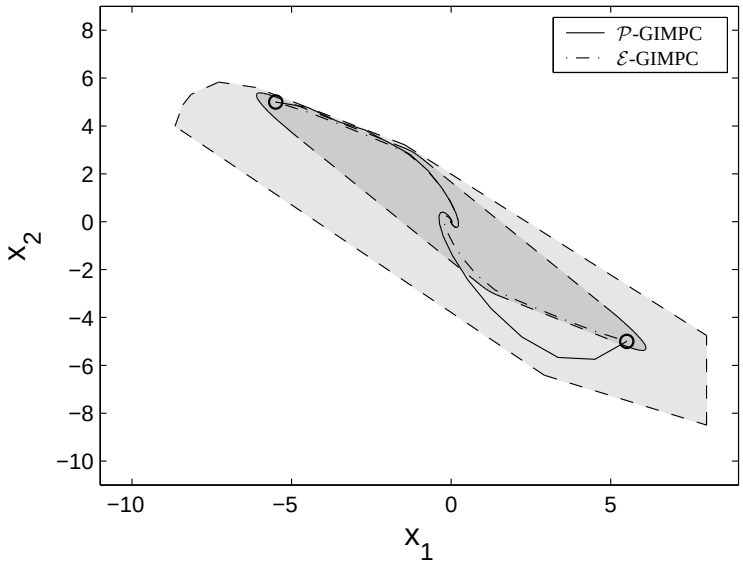


Figure 4.6: Comparison of $\mathcal{E}$-GIMPC and $\mathcal{P}$-GIMPC for system (4.56)-(4.57) using controllers (4.58). Trajectories from two symmetrically opposed initial states are depicted.

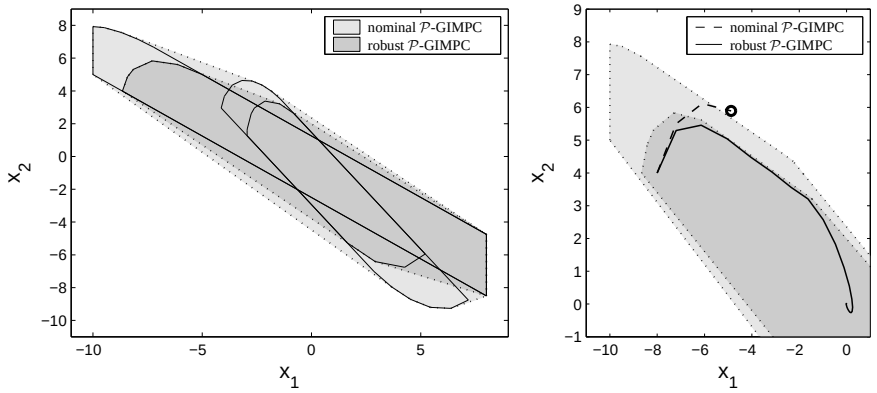


Figure 4.7: Comparison of robust and nominal $\mathcal{P}$-GIMPC. The nominal controller is designed for $[A_1 \ B_1]$, whereas the robust controller is designed for $\mathrm{Co}\{[A_1 \ B_1], [A_2 \ B_2]\}$. **Left:** The invariant sets are depicted in solid lines, the convex hulls are depicted in dotted lines. **Right:** Trajectories starting from initial state $[-8; 4]$ are depicted in solid (robust $\mathcal{P}$-GIMPC) and dashed (nominal $\mathcal{P}$-GIMPC) lines.

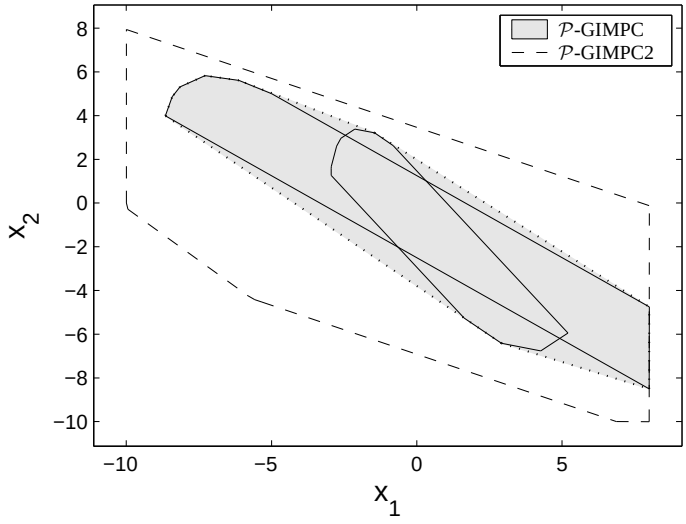


Figure 4.8: Comparison of $\mathcal{P}$-GIMPC and $\mathcal{P}$-GIMPC2 for system (4.56)-(4.57) using controllers (4.58). $\mathcal{P}$-GIMPC2 clearly extends the feasible region significantly beyond the convex hull of the individual invariant sets.

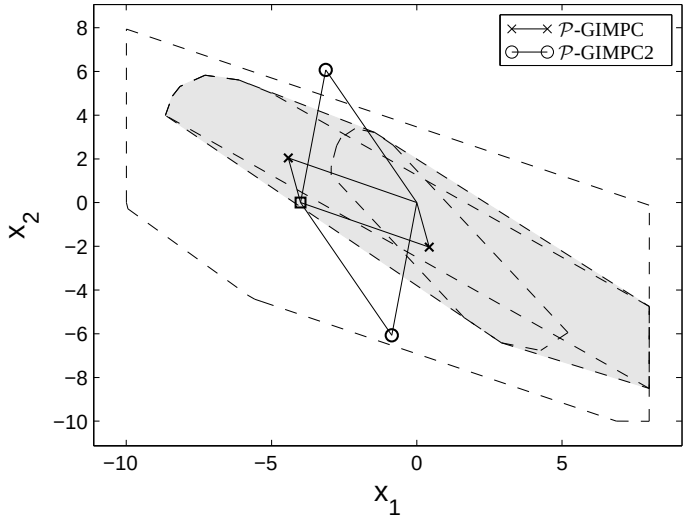


Figure 4.9: Comparison of the state decompositions obtained by $\mathcal{P}$-GIMPC and $\mathcal{P}$-GIMPC2 for system (4.56)-(4.57) using controllers (4.58) and current state $x(k) = [-4; 0]$. The values of $\hat{x}_1(k), \hat{x}_2(k)$ obtained with $\mathcal{P}$-GIMPC are indicated with $\times$-symbols, those obtained with $\mathcal{P}$-GIMPC2 with $\circ$-symbols. $x(k)$ is indicated with a $\square$-symbol.

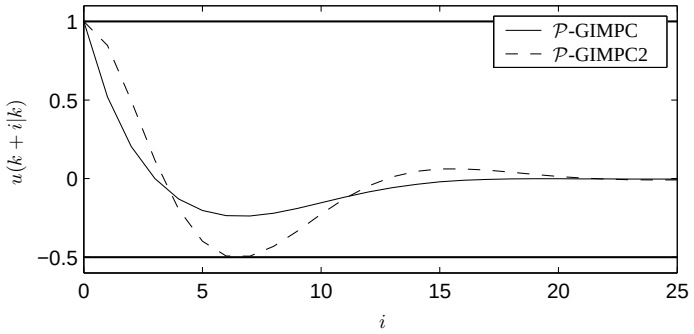


Figure 4.10: Comparison of the input sequences corresponding to the state decompositions depicted in Figure 4.9. Both input sequences satisfy the imposed input constraints, but the sequence obtained by $\mathcal{P}$-GIMPC2 comes closer to the imposed constraints, which is an indication of less conservative constraint handling. The real system behavior was taken as the LTI system defined by $[A_2\ B_2]$.

We now compare GIMPC and GIMPC2. Figure 4.8 shows the feasible region for $\mathcal{P}$-GIMPC and $\mathcal{P}$-GIMPC2. The second algorithm clearly has a significantly larger feasible region. Figure 4.9 shows the decomposition obtained by both algorithms for the state $x(k) = [-4; 0]$, which lies close to the border of the feasible region of $\mathcal{P}$-GIMPC. $\mathcal{P}$-GIMPC2 clearly obtains a state decomposition that is not allowed by $\mathcal{P}$-GIMPC, since the state component vectors lie outside both invariant sets. However, Figure 4.10 shows that the corresponding input sequences (4.25a) satisfy the imposed input constraints. The same can be verified for the state sequences (4.25b).

Finally we compare $\mathcal{P}$-GIMPC2 to $\mathcal{E}$-GIMPC2 in terms of the obtainable feasible region. This is depicted in Figure 4.11. Similar to $\mathcal{P}$-GIMPC2, $\mathcal{E}$-GIMPC2 is also able to enlarge the feasible region of $\mathcal{E}$-GIMPC, but is less successful than $\mathcal{P}$-GIMPC2. Furthermore, it should be noted that the feasible region of $\mathcal{E}$-GIMPC2 does not completely enclose that of $\mathcal{E}$-GIMPC.

4.3 Quasi-infinite horizon robust MPC

In this section we extend results presented in [123] and [70]. The common part of both methods is the use of a reparameterization of the input sequence over which the on-line optimization takes place. This reparameterization was initially introduced in [123] for LTI systems in order to improve numerical stability in MPC design. The results in [70] are presented in a different angle and emphasize the fact that this reparameterization allows the controller design to take place by calculating an invariant set and a Lyapunov function for an augmented system, very similar to the methods presented in Section 4.2.5.2. The latter method was introduced for LPV systems using ellipsoidal invariant sets.

In this section we modify the results from [70] towards the use of polyhedral sets. This allows the resulting controller to handle asymmetric constraints more efficiently,

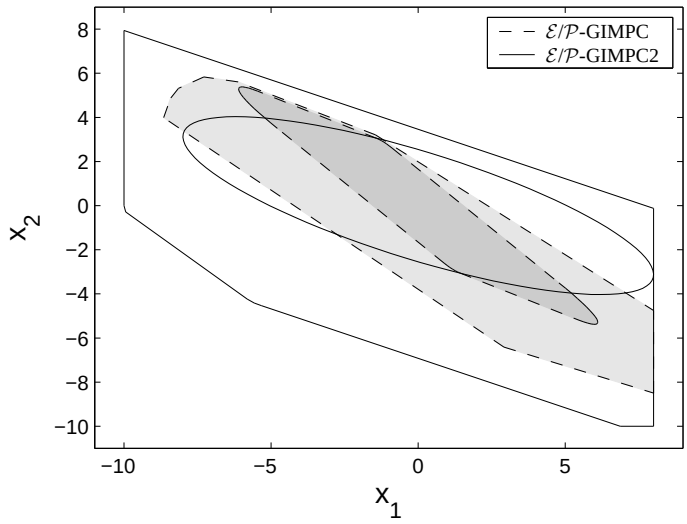


Figure 4.11: Comparison of the feasible regions of $\mathcal{E}/\mathcal{P}$-GIMPC and $\mathcal{E}/\mathcal{P}$-GIMPC2. The latter are depicted using solid lines, the former using dashed lines.

and results in a significantly larger feasible region for identical values of N. The downside is the fact that the number of constraints of the on-line optimization problem typically increases exponentially as a function of the horizon length N. Methods for tackling this disadvantage are presented in Chapter 5. Another contribution of this section is the discussion of the relationship of the use method to the open-loop and closed-loop quasi-infinite horizon MPC algorithms discussed in Sections 3.3.1 and 3.3.2.

The results presented in this section were published in [100]. A different version of this algorithm using multi-parametric programming (see e.g., [4, 6, 7, 21, 139]) was published in [119].

4.3.1 Input sequence parameterization

While in open-loop quasi-infinite horizon MPC controllers, an on-line optimization over the following input sequence takes place

$$\begin{cases} u(k+i|k) = \text{free control moves}, & i = 0, \dots, N-1, \\ u(k+i|k) = -Kx(k+i|k), & i \geq N. \end{cases} \tag{4.59}$$

In this section the following parameterization of the input sequence is used [106]:

$$\begin{cases} u(k+i|k) = -Kx(k+i|k) + Ec(k+i|k), & i = 0, \dots, N-1, \\ u(k+i|k) = -Kx(k+i|k), & i \geq N, \end{cases} \tag{4.60}$$

with $E \in \mathbb{R}^{n_u \times n_c}$ and the optimization taking place over the variables $c(k+i|k) \in \mathbb{R}^{n_c}$. This was originally proposed in [123] for the LTI case and with $E \equiv I$ and later

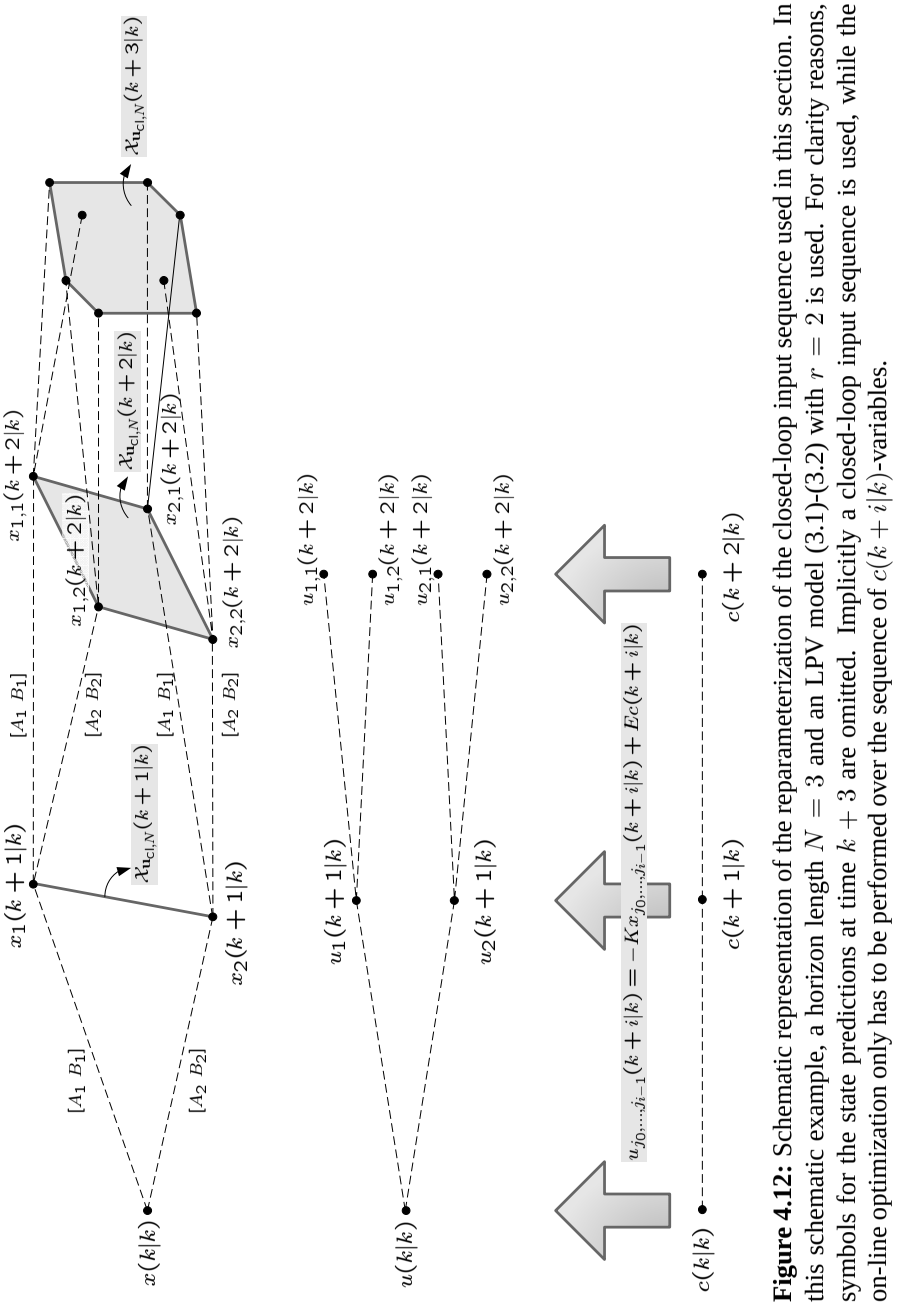


Figure 4.12: Schematic representation of the reparameterization of the closed-loop input sequence used in this section. In this schematic example, a horizon length $N = 3$ and an LPV model (3.1)-(3.2) with $r = 2$ is used. For clarity reasons, symbols for the state predictions at time $k + 3$ are omitted. Implicitly a closed-loop input sequence is used, while the on-line optimization only has to be performed over the sequence of $c(k + i|k)$-variables.

used for the LPV case in e.g., [34, 70] and the LTI case subject to disturbances [83]. A recent generalization of this reparameterization can be found in [54]. The introduction of the matrix E allows for additional degrees of freedom during the controller design. In both cases K is chosen to be a stabilizing feedback gain. One can see that this corresponds to actually controlling a pre-stabilized system $[A' \; B'] \triangleq [A - BK \;\; BE]$, with accordingly modified input constraints. Due to the fact that, opposed to A, the eigenvalues of A' are guaranteed to lie within the unity circle, the numerical conditioning of the resulting optimization problem is improved significantly, especially for large values of N. We refer to [123] for further details.

One can show [123] that in the LTI case, if one chooses $E = I$, the class of input sequences over which the optimization takes place is identical in (4.59) and (4.60). However, in the LPV case, the reparameterization (4.61) results in a different class of input sequences since the notion of feedback is introduced. Figure 4.12 shows that in the LPV case (4.60) actually induces the following parameterization of the closed-loop input sequence:

$$\begin{cases} u_{j_0,\ldots,j_{i-1}}(k+i|k) = -Kx_{j_0,\ldots,j_{i-1}}(k+i|k) + Ec(k+i|k), & i = 0,\ldots,N-1, \\ u_{j_0,\ldots,j_{i-1}}(k+i|k) = -Kx_{j_0,\ldots,j_{i-1}}(k+i|k), & i \geq N. \end{cases} \tag{4.61}$$

In this way one can see that by using the above parameterization, the on-line optimization actually takes place over a subset of the *fully parameterized* closed-loop input sequence introduced in Section 3.3.2. However, the main advantage is the fact that the notion of feedback is maintained while actually only needing the same number of optimization variables as the open-loop input sequence used in Section 3.3.1. A second advantage, which will become even more useful in the next chapter, is highlighted in the following section.

4.3.2 Controller synthesis using an augmented autonomous system

Similar to Section 4.2.5.2, this sections aims to construct an augmented autonomous system that models **a)** the closed-loop dynamics of the LPV system controlled by the MPC controller using input sequence (4.61) and **b)** the dynamics involving the construction of a feasible solution at time $k+1$ given a feasible solution at time k. This will allow us to construct the on-line objective function and inequality constraints in a straightforward way.

A straightforward choice for the augmented state vector is $x_{\text{aug}}(i) = [x(k+i|k); c(k+i|k); \ldots; c(k+i+N-1|k)]$. As suggested in [70] one can see that the following augmented system successfully models the closed-loop dynamics corresponding to the closed-loop input sequence (4.61):

$$x(i+1)_{\text{aug}} = \Psi_{\text{aug}}(k+i)x(i)_{\text{aug}}, \qquad i \in \mathbb{N}, \tag{4.62}$$

with $\Psi_{\text{aug}}(k+i) \in \Omega_{\text{aug}} \triangleq \text{Co}\{\Psi_{\text{aug},1}, \ldots, \Psi_{\text{aug},r}\}, i \in \mathbb{N}$, where

$$\Psi_{\text{aug},i} = \begin{bmatrix} A_i - B_i K & [B_i E \, 0 \, 0 \, \ldots \, 0] \\ 0 & S_{N,n_c} \end{bmatrix}, \quad S_{N,n_c} = \begin{bmatrix} 0 & I & 0 & \ldots & 0 \\ 0 & 0 & I & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \\ 0 & 0 & 0 & & I \\ 0 & 0 & 0 & \ldots & 0 \end{bmatrix}, \quad (4.63)$$

where $S_{N,n_c} \in \mathbb{R}^{N.n_c \times N.n_c}$ is the $N.n_c$-dimensional shift matrix with length n_c shifts. Given an augmented state vector $x_{\text{aug}}(i)$, the corresponding state vector $x(k+i|k)$ can be expressed as $x(k+i|k) = \Gamma_x x_{\text{aug}}(i)$, with $\Gamma_x = [I \; 0 \; \ldots \; 0]$, while the corresponding input vector $u(k+i|k)$ can be expressed as $u(k+i|k) = \Gamma_u x_{\text{aug}}(i)$ with $\Gamma_u = [-K \; E \; 0 \; \ldots \; 0]$. State and input constraints (2.10)-(2.11) can now be expressed in terms of $x_{\text{aug}}(i)$:

$$x_{\text{aug}}(i) \in \mathcal{X}_{\text{aug}} \equiv \{x | A_{\text{aug}} x \leq \mathbf{1}\}, \qquad i \in \mathbb{N}, \qquad (4.64a)$$

$$A_{\text{aug}} = [\Gamma_x; \Gamma_u]. \qquad (4.64b)$$

In order to guarantee that, given a set of feasible control actions $c(k|k), \ldots, c(k+N-1|k)$ at time k, the set of control actions at time $k+1$ generated by the dynamics (4.64) also will be feasible, we construct a set $\mathcal{S}_{\text{aug}}$ that is positive invariant with respect to (4.63) and feasible with respect to (4.64). In [70] this is done by means of ellipsoidal invariant sets. In this section $\mathcal{S}_{\text{aug}}$ is constructed using Algorithm 2.4 resulting in a polyhedral set.

An objective function $x_{\text{aug}}^{\text{T}}(i) P x_{\text{aug}}(i)$ that serves as an upper bound to the real worst-case value of the control cost and is guaranteed to be monotonically decreasing under the dynamics given by (4.63), can now be constructed by solving the following SDP:

$$\min_{P \in \mathbb{S}_{++}^{n.n_x}} \quad \text{Tr}(P), \qquad (4.65a)$$

$$\text{subject to} \quad P \succeq \Psi_{\text{aug},j}^{\text{T}} P \Psi_{\text{aug},j} + \Gamma_u^{\text{T}} R \Gamma_u + \Gamma_x^{\text{T}} Q \Gamma_x, \quad j = 1, \ldots, r, \qquad (4.65b)$$

4.3.3 Algorithm formulation and properties

We can now formulate the following algorithm, by summarizing the previous section:

Algorithm 4.7 (**Robust MPC using polyhedral sets ($\mathcal{P}$-RMPC)).** *Given a system* (3.1),(3.2)*, constraints* (2.10)-(2.11)*, cost weighting matrices Q, R and a horizon length $N \in \mathbb{Z}_0^+$, perform the following steps:*
Off-line:

- *Construct the augmented system* (4.63)-(4.64) *and calculate a feasible positive invariant set $\mathcal{S}_{\text{aug}}$ for this system using Algorithm 2.4.*
- *Calculate P by solving* (4.65)*.*

On-line:
At every time instant k, given the current state $x(k)$, perform the following steps:

- *Solve the optimization problem*

$$\min_{c(k|k),\ldots,c(k+N-1|k)} x_{\mathrm{aug}}^{\mathrm{T}}(0)Px_{\mathrm{aug}}(0), \quad \textit{subject to} \quad x_{\mathrm{aug}}(0) \in \mathcal{S}_{\mathrm{aug}}, \quad (4.66)$$

  in order to obtain optimal control actions $c(k|k), \ldots, c(k+N-1|k)$.

- *Apply the input $u(k) = -Kx(k) + c(k|k)$.*

This algorithm extends the algorithm introduced in [70] in two ways. First of all polyhedral invariant sets are used instead of ellipsoidal invariant sets. Secondly, an objective function is used that is more representative of the real control objective. The algorithm in [70] essentially uses $P = I$.

Theorem 4.3. *Algorithm 4.7 is recursively feasible and, if it is initially feasible, guarantees robust satisfaction of* (2.10)-(2.11) *and asymptotic stability of the closed-loop system.*

Proof: The proof is similar to the proofs of Lemma's 4.4, 4.5 and 4.6 and is hence omitted. For more details we refer to [100]. □

The following lemma establishes a link with the stability framework discussed in Section 3.3.2.

Lemma 4.8. *The control moves $c(k|k), \ldots, c(k+N-1|k)$ determined using optimization problem* (4.66) *for a given state $x(k)$ guarantee that the input sequence* (4.60) *drives the system in N time steps into a feasible invariant set for the closed-loop system $\{A_1 - B_1K, \ldots, A_r - B_rK\}$ subject to constraints* (2.10)-(2.11)*.*

Proof: By applying dynamics (4.63) one can see that $x_{\mathrm{aug}}(N) = [x(k+i|k); 0; \ldots; 0]$. This shows that $x(k+i|k)$ lies inside an intersection of $\mathcal{S}_{\mathrm{aug}}$ along the first n_x dimensions, which we will denote as $\mathcal{S}_{\mathrm{aug},n_x}$. By inspecting the first n_x dimensions of constraints (4.64) and dynamics (4.63), or similarly by filling in $x_{\mathrm{aug}}(N)$ in these expressions, and by observing that $\mathcal{S}_{\mathrm{aug}}$ is a feasible positive invariant set for (4.63),(4.64), it is clear that $\mathcal{S}_{\mathrm{aug},n_x}$ is a feasible positive invariant set for the system mentioned in the formulation of this lemma. □

Lemma 4.8 shows that the notion of a terminal controller and a terminal constraint is also present in Algorithm 4.7. A similar observation can be made about a link with a terminal cost by inspecting the upper left $n_x \times n_x$ submatrix of P and by writing out the same submatrix part of inequalities (4.65b).

4.3.4 Controller design

Figure 4.13 shows a possible design approach for a controller based on Algorithm 4.7. The three important tuning parameters are N, n_c and E. The influence of these parameters is discussed in the following sections.

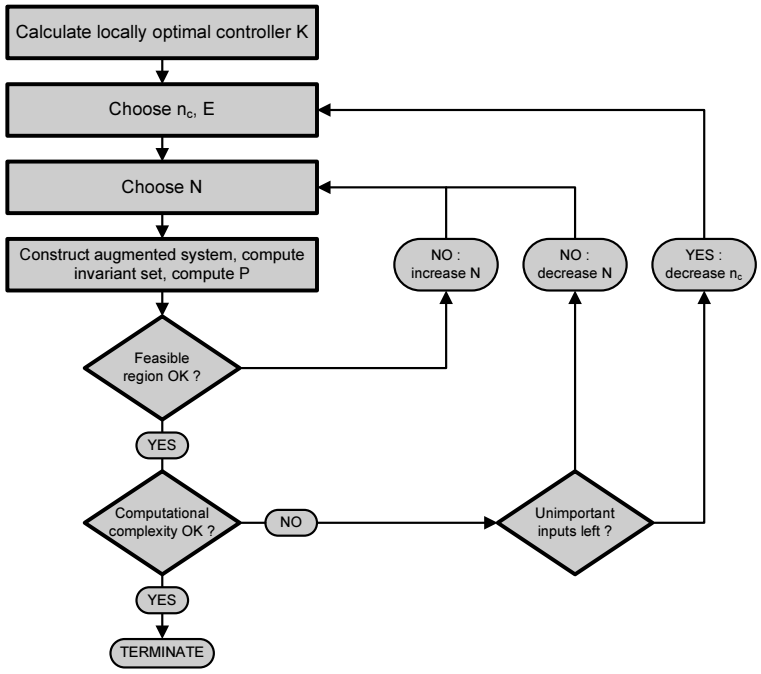


Figure 4.13: Flow chart of a possible design process of an interpolation based MPC controller. The main design choices consist of N, n_c and E. Larger N and larger $n_c \in \{1, \ldots, n_u\}$ typically lead to enlarged feasible regions and computational complexity increases.

4.3.4.1 Choice of N

As is also illustrated in Section 4.3.5, an increasing value of N leads to a larger feasible region. However, also the on-line computational complexity can increase dramatically. Although the number of optimization variables is equal to $N.n_c$ and hence only increases as a linear function of N, the number of inequality constraints describing $\mathcal{S}_{\text{aug}}$ typically increases exponentially as a function of N. As a result, in typical situations only small values of N can be chosen. This problem is tackled in the next chapter.

4.3.4.2 Choice of n_c, E

The introduction of variable E is aimed at introducing additional degrees of freedom in the controller design. The algorithms described in [70, 123] do not use this variable and hence implicitly make the choice $E \equiv I$. In many cases this is often also a good initial choice when designing the controller.

However, if a given MIMO system has several inputs that are *unimportant* for the given control problem it might not be needed to spend much effort to optimally choose control actions for these inputs. One can use this information to obtain a

decreased on-line computational complexity by decreasing n_c and omitting the rows of E corresponding to these inputs.

For example, when the second input of a 3-input system is considered unimportant one might want to choose $n_c = 2$ and $E = [1\ 0; 0\ 0;\ 0\ 1]$. Similarly, when the effect of the two first inputs is considered to be highly correlated, one might want to choose $E = [1\ 0; 1\ 0; 0\ 1]$.

4.3.5 Example

We retake the examples given in Section 4.2.7 and assess the different dimensions of performance of Algorithm 4.7. In order to make a fair comparison we choose $K = K_1$. Since there's only 1 input, we choose $E = I$.

Figure 4.14 shows the feasible regions for $\mathcal{P}$-RMPC for increasing values of N. Larger values of N result in larger feasible regions, but the increase is not as significant as the increase obtained when increasing n in $\mathcal{P}$-GIMPC(2). However, the on-line computational complexity does increase significantly when increasing n, as can be observed in Table 4.3. In this example interpolation based algorithms seem to result in a better trade-off between computational complexity and obtained feasible region.

Figure 4.15 shows trajectories for $\mathcal{P}$-RMPC ($N = 6$), $\mathcal{P}$-GIMPC and $\mathcal{P}$-GIMPC2. The real plant behavior is chosen as $[A_2\ B_2]$. The different initial states are chosen inside the intersection of the feasible regions of the different algorithms. Control behavior can be observed to be qualitatively identical. The average control cost per initial state was 51.41, 51.73 and 51.42 respectively. Hence, from an optimality point of view the three algorithms behave equally well in this example. Figure 4.16 shows the obtained input signals for the different algorithms and the different initial states. Also here no significant differences can be observed.

Finally, for the sake of completeness, Figure 4.17 shows trajectories starting close to the boundaries of the feasible regions of the different algorithms, in order to show that the algorithms are indeed stabilizing in the entire feasible region.

4.4 Conclusions

In this chapter several robust control techniques are introduced that make use of polyhedral invariant sets in order to handle constraints.

Both interpolation based and quasi-infinite horizon MPC algorithms are discussed and extended towards the use of polyhedral invariant sets. These extensions result in enlarged feasible regions, less conservative constraint handling in general and the ability to efficiently cope with asymmetric constraints in particular. Another advantage is that the on-line optimization reduces to solving a QP, which can be solved more efficiently than an SDP. The main disadvantage of the obtained algorithms is the questionable scalability properties towards large-dimensional systems or algorithms using many degrees of freedom. This is the main concern discussed in the next chapter.

Finally, the seminal robust constrained controller synthesis results introduced in [68] are extended towards using polyhedral instead of ellipsoidal invariant sets, leading to feedback laws with improved optimality. These results can prove to be useful when

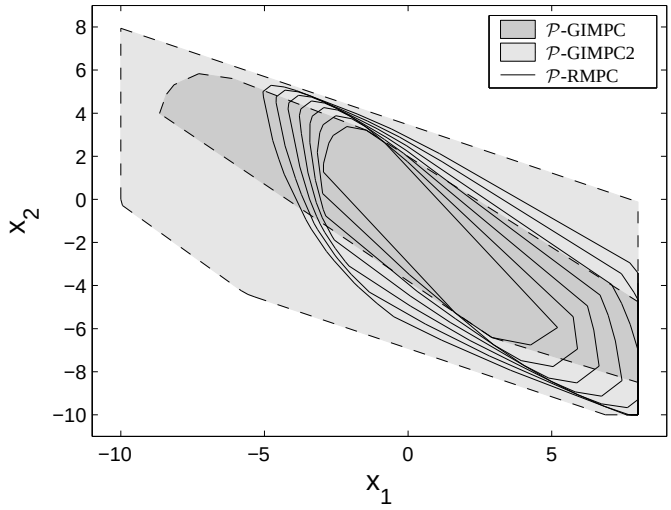


Figure 4.14: Comparison of the feasible region of $\mathcal{P}$-RMPC (solid) for $N = 0, \ldots, 6$ applied to system (4.56)-(4.57) using terminal controller $K = K_1$ and the feasible regions of $\mathcal{P}$-GIMPC (dark grey) and $\mathcal{P}$-GIMPC2 (light grey).

	#variables	**#constraints**	**vol.**
$\mathcal{P}$-**RMPC,** $N = 0$	0	12	27.94
$\mathcal{P}$-**RMPC,** $N = 1$	1	21	39.42
$\mathcal{P}$-**RMPC,** $N = 2$	2	40	51.26
$\mathcal{P}$-**RMPC,** $N = 3$	3	73	63.73
$\mathcal{P}$-**RMPC,** $N = 4$	4	135	76.59
$\mathcal{P}$-**RMPC,** $N = 5$	5	259	87.34
$\mathcal{P}$-**RMPC,** $N = 6$	6	506	96.88
$\mathcal{P}$-**GIMPC**	6	24	79.13
$\mathcal{P}$-**GIMPC2**	2	63	181.82

Table 4.3: Computational complexity of the on-line optimization problems and volume of the feasible regions of $\mathcal{P}$-RMPC for system (4.56)-(4.57) using terminal controller $K = K_1$ and $\mathcal{P}$-GIMPC(2) using controllers (4.58). Complexity is expressed in terms of the number of optimization variables and the number of inequality constraints.

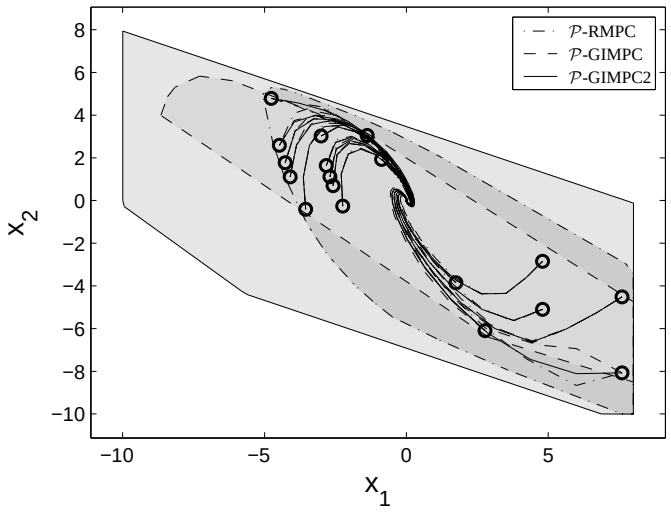


Figure 4.15: Comparison of the control behavior of $\mathcal{P}$-RMPC ($N = 6$) applied to system (4.56)-(4.57) using terminal controller $K = K_1$ and the control behavior of $\mathcal{P}$-GIMPC and $\mathcal{P}$-GIMPC2 for different initial states inside the intersection of the feasible regions of the three controllers.

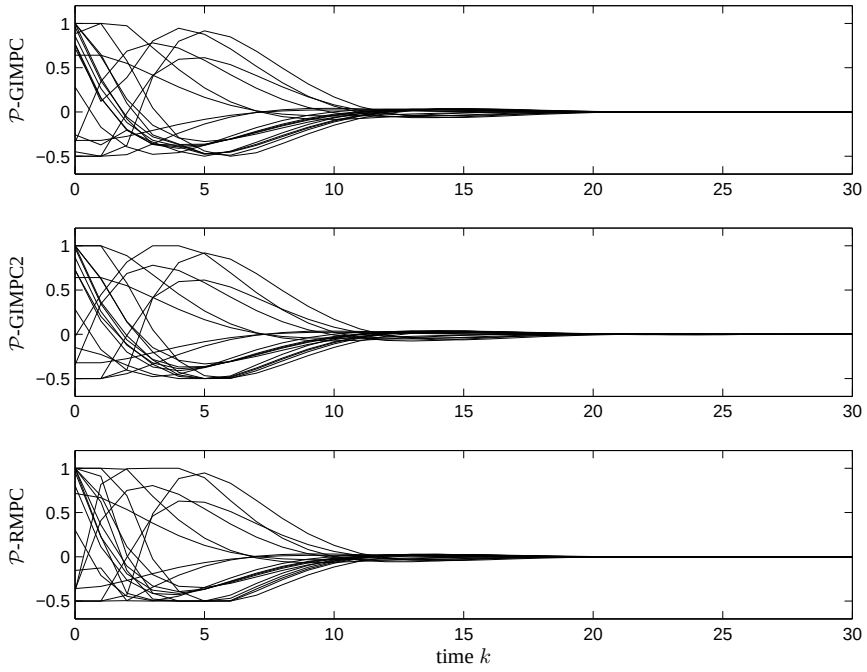
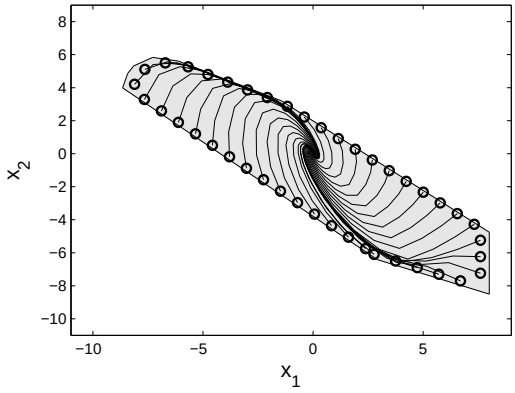
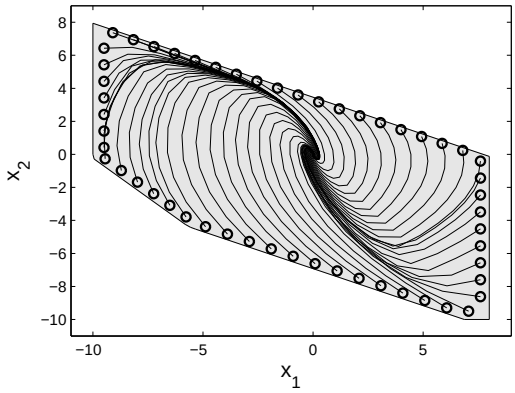


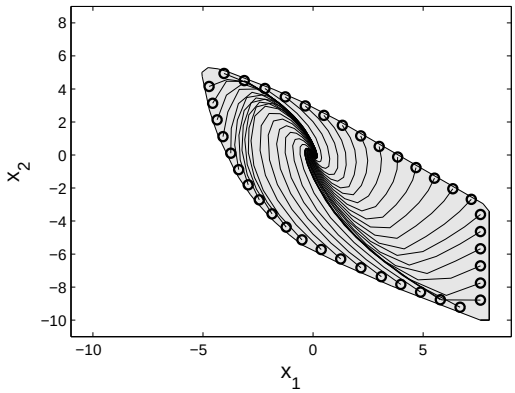
Figure 4.16: Comparison of the input sequences corresponding to the trajectories depicted in Figure 4.15. No significant differences can be observed.



(a) Trajectories for $\mathcal{P}$-GIMPC.



(b) Trajectories for $\mathcal{P}$-GIMPC2.



(c) Trajectories for $\mathcal{P}$-RMPC.

Figure 4.17: Trajectories for $\mathcal{P}$-GIMPC (a), $\mathcal{P}$-GIMPC2 (b) and $\mathcal{P}$-RMPC with $N = 6$ (c) applied to system (4.56)-(4.57) for different initial states inside their respective feasible regions.

constructing linear feedback laws for use in the interpolation based or quasi-infinite horizon MPC algorithms discussed in this and next chapters.

Chapter 5

Reduced-Complexity Invariant Sets in Robust MPC

"Any intelligent fool can make things bigger and more complex..."

– Albert Einstein (1879-1955) –

The contribution of this chapter is the formulation of new algorithms for the construction of invariant inner approximations of the MAS for LPV systems. The main aim is to obtain invariant sets described with a reduced number of constraints compared to the real MAS. Several examples show that the complexity of the obtained invariant sets has significantly improved scaling behavior as a function of the dimensionality. These sets are then used in the MPC algorithms presented in Chapter 4 resulting in an improved trade-off between the volume of the feasible region and the on-line computational complexity.

5.1 Scalability analysis

Chapter 2 showed that the computation of the MAS for low-dimensional LPV systems is computationally tractable, due to the fact that typically a large fraction of the constraints under consideration is redundant. The examples at the end of Chapter 4, however, seem to indicate that the number of constraints describing the MAS seems to increase exponentially as a function of the dimensionality of the system for which the invariant set is calculated. This can clearly be observed in Table 4.3: in case of the $\mathcal{P}$-RMPC algorithms, the number of constraints of the on-line optimization, which in turn is determined by the number of constraints describing the invariant set for the

augmented system (4.63), roughly doubles with every increase of N. Whether this is due to the specific structure that is present in (4.63) or whether this phenomenon is generally valid is not immediately clear. This section aims to shed light on this issue.

5.1.1 Monte-Carlo experiment

In this section we take a look at this scaling behavior for a class of LPV systems that do not exhibit the specific structure of the augmented systems constructed in Chapter 4. We randomly generate LPV systems with $r = 2$ as follows:

$$\Phi_1 = a(I + 0.1R_1), \qquad \Phi_2 = a(I + 0.1(R_1 + 0.1R_2)), \tag{5.1}$$

with $R_1, R_2 \in \mathbb{R}^{n_x \times n_x}$ square matrices whose entries are picked from a normal distribution with $\mu = 0, \sigma = \sqrt{n_x}$. In this way the norms of the R_i-matrices do not increase as n_x increases. The scalar a is chosen such that $\hat{\rho}(\{\Phi_1, \Phi_2\}) \leq 0.9$. This is done by calculating an upper bound $\overline{\rho}$ to $\hat{\rho}(\{\Phi_1, \Phi_2\})$ as $\overline{\rho} = \max_{\Phi(1\ldots k)\in\{\Phi_1,\Phi_2\}} \|\Phi(1) \cdot \ldots \cdot \Phi(k)\|^{\frac{1}{k}}$ with $k = 15$ (see [71]) and setting $a = \frac{0.9}{\overline{\rho}}$. Constraints are chosen as $A_x = [W; -2W]$, with $W \in \mathbb{R}^{n_x \times n_x}$ randomly generated in the same way as R_1 and R_2. These randomly generated systems can be seen as discretized versions (with sample time equal to 0.1 time units) of random continuous-time LPV systems with 10% uncertainty.

Table 5.1 shows that also in this case, the number of constraints increases exponentially as a function of the system dimensionality. Non-surprisingly the time needed to construct the sets also increases exponentially as a function of the dimensionality. This shows that the scaling behavior observed in the previous chapter is not a result of the specific structure of the augmented sets considered there, but that this behavior is inherent to polyhedral invariant sets for LPV systems.

Taking Theorem 2.2 into consideration as well as the fact that in these examples m_x only increases linearly with the dimensionality, only an increase in the admissibility index $k^\star$ can be the cause of this exponential increase. Furthermore, since by construction $\hat{\rho}(\Omega') < 0.9$ for all randomly generated systems, we can conclude that the variables a, b, c (see the proof of Theorem 2.2) are the determining factors in this case.

One insight that we would like to emphasize here is the fact that the product $a.b$ can be interpreted in terms of the shape of $\mathcal{X}$. One can verify that $a.b \approx 1$ if the shape of $\mathcal{X}$ is very close to a hyper-sphere, and that $a.b = \sqrt{n_x}$ if $\mathcal{X}$ is taken as a hyper-cube. Since in reality typically component-wise constraints are imposed, which corresponds to a hyper-box shaped constraint set $\mathcal{X}$ where similarly $a.b \sim \sqrt{n_x}$, one can now directly find a plausible cause for the observed scaling behavior.

Secondly, it can also be observed that the fraction of redundant constraints decreases as the dimensionality increases. A useful measure to inspect this effect is the *equivalent branching factor* r_{equiv}, which is defined by means of the following equation:

$$m_x \frac{1 - r_{\text{equiv}}^{k^\star+1}}{1 - r_{\text{equiv}}} = \text{rows}(A_{\overline{\mathcal{S}}}), \tag{5.2}$$

	rows$(A_{\overline{\mathcal{S}}})$	$\mathbf{k}^\star$	**CPU-t. (s)**	$\mathbf{r_{equiv}}$
$\mathbf{n_x = 2}$	10.93	4.22	10.78	0.49
$\mathbf{n_x = 3}$	29.82	6.33	19.09	0.70
$\mathbf{n_x = 4}$	53.04	7.21	29.42	0.86
$\mathbf{n_x = 5}$	94.76	7.95	53.58	0.95
$\mathbf{n_x = 6}$	154.91	8.76	98.87	1.03
$\mathbf{n_x = 7}$	257.25	9.06	204.59	1.10
$\mathbf{n_x = 8}$	363.12	9.00	336.81	1.16
$\mathbf{n_x = 9}$	469.30	9.70	513.78	1.17
$\mathbf{n_x = 10}$	556.90	9.10	671.87	1.21

Table 5.1: Results of the Monte-Carlo experiment discussed in Section 5.1.1. Average number of constraints, average tree depth and average computation time of invariant sets for 100 randomly generated systems are reported for $n_x = 2, \ldots, 10$. Due to time constraints the results for $n_x = 8$ were obtained with a sample size of 50, the results for $n_x = 9, 10$ were obtained with a sample size of 10.

and that can be interpreted as the value of r that, according to expression (2.23), would explain the observed number of constraints. Since expression (2.23) gives a worst-case value for rows$(A_{\overline{\mathcal{S}}})$, one can see that $r_{\text{equiv}} \leq r$ by definition. Larger values of r_{equiv} indicate that a larger fraction of all possible constraints is non-redundant. Table 5.1 shows that r_{equiv} increases as a function of the dimensionality.

5.1.2 Theoretical considerations

The Monte-Carlo experiment discussed in the previous section clearly shows that for randomly generated systems with a fixed value of the JSR, the complexity of the resulting invariant sets increases exponentially as the dimensionality of the system increases. We now state a few properties regarding the $\mathcal{P}$-GIMPC2 and $\mathcal{P}$-RMPC algorithms discussed in Chapter 4 that relate the above observations to the specific structure present in these two algorithms, both of which rely on the construction of a polyhedral invariant set for an augmented system. We first state the following Lemma that directly proves the corollaries below.

Lemma 5.1 (**JSR of upper block triangular matrices).** *Given a set $\mathcal{M} = \{M_1, \ldots, M_r\}$, with every M_i an upper block triangular matrix with diagonal blocks $M_{i,(j,j)}$, then*

$$\hat{\rho}(\mathcal{M}) = \max_j \hat{\rho}(\{M_{1,(j,j)}, \ldots, M_{r,(j,j)}\}). \tag{5.3}$$

Proof: This property directly follows from the fact that the diagonal blocks of products of such matrices are equal to the products of the respective diagonal blocks and from the fact that the norm of an upper block triangular matrix is given by the maximum of the norms of the diagonal blocks. □

It goes without saying that the same can be proven for lower block triangular

matrices. The following two corollaries follow directly and are of more importance in this context.

Corollary 5.1 (**Joint spectral radius and $\mathcal{P}$-GIMPC2).** *The joint spectral radius $\hat{\rho}_{\text{aug}}$ of augmented system* (4.44) *and the JSR $\hat{\rho}_i$ of the autonomous systems formed by LPV system* (3.1),(3.2) *and feedback gains K_i satisfy the following relation:*

$$\hat{\rho}_{\text{aug}} = \max_i \hat{\rho}_i. \tag{5.4}$$

Proof: Applying Lemma 5.1 to (4.44) directly yields (5.4). □

The JSR of the augmented system (4.44) hence depends on the largest JSR of the closed loop systems corresponding to the individual control laws between which the interpolation is performed. The disadvantage is that less agressive controllers typically result in slower convergence (and hence a larger JSR) while at the same time providing a larger feasible region.

Furthermore, the constraints (4.45) that are imposed are unbounded along the last $(n-1)n_x$ dimensions of the augmented system, which prevents the use of 2.2 to be used in order to determine $k^\star_{\max}$. However, it can be verified that if the different controllers $K_1, \ldots, K_n$ are unique, Corollary 2.1 can be applied with $j = n-1$. This shows that $k^\star_{\max}$ further increases as more different controllers are used.

These two observations show theoretically that there is an inherent trade-off in $\mathcal{P}$-GIMPC2 between the complexity of the obtained invariant set of the augmented system and the size of the feasible region, both of which increase as n increases or more detuned controllers are used.

Corollary 5.2 (**Joint spectral radius and $\mathcal{P}$-RMPC).** *The joint spectral radius $\hat{\rho}_{\text{aug}}$ of augmented system* (4.63) *and the JSR $\hat{\rho}_{\text{cl}}$ of the closed-loop system formed by LPV system* (3.1),(3.2) *and the feedback gain K are equal:*

$$\hat{\rho}_{\text{aug}} = \hat{\rho}_{\text{cl}}. \tag{5.5}$$

Proof: Since S_{N,n_c} is a nil-potent matrix and hence its spectral radius is 0, applying Lemma 5.1 to (4.63) directly yields (5.5). □

Similar to $\mathcal{P}$-GIMPC2, it can be verified that Corollary 2.1 can be applied with $j = N-1$ in order to assess the complexity of the resulting invariant set used to construct a $\mathcal{P}$-RMPC controller. Hence, although the JSR of the augmented system (4.63) is independent of N, the value of $k^\star_{\max}$ still increases as N is increased.

This shows that from a theoretical point of view also with the $\mathcal{P}$-RMPC algorithm a trade-off has to be made between the number of on-line constraints and the size of the feasible region of the controller, both of which increase as N is increased.

5.1.3 Conclusion

This section makes an analysis of the scaling behavior of invariant sets for randomly generated systems on the one hand, and of invariant sets for two classes of structured systems on the other hand. In both cases an exponential scaling behavior is observed

as a function of the dimensionality of the (augmented) system. This scaling behavior is due to the fact that we construct the MAS for the given systems exactly. The maximality of the volume of the invariant set seems to come at the expense of a high complexity.

Therefore in the following section new algorithms are discussed for constructing reduced-complexity invariant sets, in order to improve the scaling behavior of the resulting invariant sets. These sets are inner approximations to the MAS and hence represent a different trade-off between volume and complexity.

5.2 Reduced-complexity invariant sets

5.2.1 State of the art

Three different approaches for the construction of invariant sets can be identified based on the assumptions that are made from the outset or the restrictions that are imposed during the construction of the sets.

A first approach consists of imposing a fixed low complexity structure on the invariant set after which the aim is to maximize the volume without losing the invariance property. The results discussed in [28, 30, 68, 75] can be classified in this category. The method proposed in [68], which is already discussed in Chapter 2, imposes an ellipsoidal structure, whereas the methods proposed in [28, 30, 75] impose a parallellotopic structure. Both structures can be expressed as $\|Wx\|_p \leq 1$ with $W \in \mathbb{R}^{n_x \times n_x}$, where $p = 2$ for the ellipsoidal case and $p = \infty$ for the parallellotopic case. Imposing an ellipsoidal structure has some important disadvantages as is already discussed in Chapter 2, but leads to a formulation based on convex optimization. Parallellotopic invariant sets on the other hand have the additional disadvantages that they can only be constructed for a limited class of stable systems and that their construction is not based on convex optimization. Their advantage lies in the fact that they can be decribed by linear inequality constraints, which facilitates the on-line optimization. Recent results [30] use partial invariance to obtain larger invariant sets of low complexity but are still restricted to the same class of systems.

A different diametrically opposed approach is the construction of the MAS, where essentially the a maximal volume is imposed. Within this framework of maximality of the volume, the only degree of freedom for reducing the complexity of the resulting invariant set is the removal of redundant constraints. This approach hence leaves no degrees of freedom to the user.

A third approach, which is the approach pursued in this chapter, is that of making a trade-off between maximal volume and minimal complexity. The methods proposed here start from the MAS which is then approximated in order to reduce the number of constraints. The methods proposed here make these approximations either during (*pruning*) or before (*trimming*) the actual construction of the invariant set. In this way the real MAS is never explicitly constructed, which results in significant efficiency gains. By means of a few tuning parameters the trade-off between maximal volume and minimal complexity can be adjusted according to the user's preferences. These results are also discussed in [102].

Based on expression (2.23) two possible strategies can be laid out within this third class of methods. A first strategy is trying to reduce the number of parallel branches present in the tree, in order to reduced the effective value of r in expression (2.23). This method is called *pruning* and is discussed in Section 5.2.2. A second strategy is reducing the tree depth, in order to reduce the exponent present in (2.23). This method is called *trimming* and is discussed in Section 5.2.3.

5.2.2 Reducing branching: pruning

In this section an algorithm is introduced that constructs an invariant set in a similar way to Algorithm 2.4 but uses constraint tightening in order to reduce the number of constraints describing the invariant set. The first subsection describes how constraints can be tightened without losing the invariance property. In the second subsection we then formulate an algorithm that makes use of these insights to reduce the number of inequalities describing the invariant sets. A third subsection gives rules of thumb for parameter tuning.

5.2.2.1 Constraint tightening

First, we rephrase Algorithm 2.3 in terms of the sets $\mathcal{X}_i$ instead of $\mathcal{O}_i$ in order to create a theoretical framework within which we can formulate the concept of *pruning*.

Algorithm 5.1 (Approximate maximal $\langle\Omega'\rangle$-invariant set construction). *Given a system* (2.1),(2.2) *subject to constraints* (2.10)*, perform the following steps:*

1. *Initialize* $\mathcal{X}_0 := \mathcal{X}, i := 0$.
2. *Execute iteratively until* $\bigcap_{j=0}^{i} \mathcal{X}_j \subseteq Pre_{\langle\Omega'\rangle}(\mathcal{X}_i)$*:*
   (a) *Set* $i := i + 1$.
   (b) *Calculate* $\mathcal{X}_i$ *such that* $\mathcal{X}_i \subseteq Pre_{\langle\Omega'\rangle}(\mathcal{X}_{i-1})$.

Return the set $\overline{\mathcal{S}} \triangleq \bigcap_{j=0}^{i} \mathcal{X}_i$ *and admissibility index* $k^\star \triangleq i$.

Note that the equality in step 2b) of Algorithm 2.3 has been replaced with the above inclusion. It can be shown that the above algorithm, if it terminates still produces a valid $\langle\Omega'\rangle$-invariant set, but maximality is not guaranteed anymore:

Theorem 5.1 (Correctness of Algorithm 5.1). *If Algorithm 5.1 terminates, the resulting set* $\overline{\mathcal{S}}$ *is* $\langle\Omega'\rangle$*-invariant.*

Proof: One can see that by construction at the end of every iteration i, the set $\mathcal{O}_i = \bigcap_{j=0}^{i} \mathcal{X}_j$ satisfies $\mathcal{O}_i \subseteq \text{Pre}_{\langle\Omega'\rangle}(\mathcal{O}_{i-1})$. If the termination condition is satisfied, then one has that $\mathcal{O}_i \subseteq \text{Pre}_{\langle\Omega'\rangle}(\mathcal{O}_{i-1}) \cap \text{Pre}_{\langle\Omega'\rangle}(\mathcal{X}_i)$ which is equivalent with $\mathcal{O}_i \subseteq \text{Pre}_{\langle\Omega'\rangle}(\mathcal{O}_{i-1} \cap \mathcal{X}_i)$, which in turn is equivalent with $\mathcal{O}_i \subseteq \text{Pre}_{\langle\Omega'\rangle}(\mathcal{O}_i)$. Due to Lemma 2.1 this guarantees that $\overline{\mathcal{S}} \equiv \mathcal{O}_i$ is $\langle\Omega'\rangle$-invariant. □

The aim is now, in every iteration, to construct $\mathcal{X}_i$ such that it approximates $\text{Pre}_{\langle\Omega'\rangle}(\mathcal{X}_{i-1})$ as close as possible, but with a reduced number of inequality constraints. How this is done exactly is explained by means of Figure 5.1.

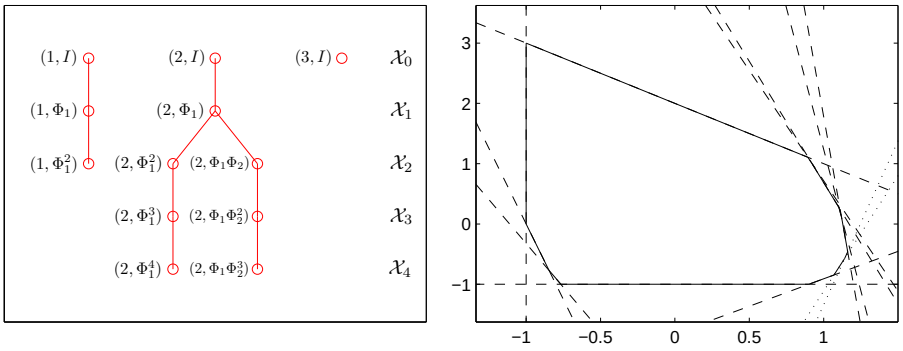


Figure 5.1: Illustration of the structure of the Maximal Admissible Set for an example system defined by $n_x = 2, r = 2, \Phi_1 = [1\ 0.2;\ -0.5\ 0.55]$ and $\Phi_2 = [1\ 0.2;\ -0.55\ 0.45]$, subject to constraints $A_x x \leq \mathbf{1}$ with $A_x = [-1\ 0; 0\ -1; 0.5\ 0.5]$. **Left:** the tree structure of the MAS, where the nodes of the tree represent the different constraints of the MAS, with the constraints of the different sets $\mathcal{X}_i$ appearing at different depths of the trees as indicated on the figure. The notation (i, M) denotes the constraint $a_i^{\mathrm{T}} M x \leq 1$, with a_i^{T} denoting the i-th row of A_x. **Right:** a depiction of the MAS. The constraints $(1, *)$ bound the MAS from the left; constraints $(2, *)$ bound the MAS from the lower right; constraint $(3, I)$ is the diagonal constraint bounding the MAS from above. Constraints $(2, \Phi_1^2)$ and $(2, \Phi_1\Phi_2)$ are depicted as dotted lines.

When inspecting Figure 5.1 one can notice that two separate branches appear below constraint $(2, \Phi_1)$, corresponding to the two vertices Φ_1, Φ_2 of the uncertainty polytope. This *branching effect*, which becomes more frequent with increasing state dimension, is the primary reason why polyhedral invariant sets for LPV systems can be dramatically more complex than invariant sets for LTI systems. Therefore, the main purpose of this section is to find a way to efficiently reduce this branching effect.

Eliminating the branching effect observed in Figure 5.1 is possible by tightening one of the two constraints $(2, \Phi_1^2)$ and $(2, \Phi_1\Phi_2)$. When comparing the two dotted lines it is clear that by tightening either of both constraints with only a small factor, the other becomes redundant. This actually corresponds to choosing $\mathcal{X}_2$ slightly smaller than $\mathrm{Pre}_{\langle\Omega'\rangle}(\mathcal{X}_1)$, which fits within the framework of Algorithm 5.1. Therefore, choosing some constraints slightly tighter than necessary can potentially result in a complexity reduction of the invariant set, while still guaranteeing $\langle\Omega'\rangle$-invariance of the resulting set $\overline{\mathcal{S}}$.

The problem of finding how much a constraint should be tightened in order to make another constraint redundant can be formulated as follows:

Problem 5.1 (Constraint tightening). *Given real matrices* $A \in \mathbb{R}^{m\times n}, a_1 \in \mathbb{R}^n, a_2 \in \mathbb{R}^n$ *with* $\nu > 1$,

$$\nu = \max_x\ a_2^{\mathrm{T}} x \quad \text{s.t.} \quad \begin{bmatrix} A \\ a_1^{\mathrm{T}} \end{bmatrix} x \leq \mathbf{1}, \tag{5.6}$$

find a scalar $\eta > 1$ such that $\nu' = 1$, with

$$\nu' = \max_x \; a_2^{\mathrm{T}} x \quad \text{s.t.} \; \left[\begin{array}{c} A \\ \eta a_1^{\mathrm{T}} \end{array} \right] x \leq \mathbf{1}. \tag{5.7}$$

Note that the property $\nu > 1$ indicates that $a_2^{\mathrm{T}} x \leq 1$ is not redundant with respect to $[A;\; a_1^{\mathrm{T}}] x \leq \mathbf{1}$. The scalar η is the factor with which $a_1^{\mathrm{T}} x \leq 1$ should be made stricter in order to make $a_2^{\mathrm{T}} x \leq 1$ redundant.

The following lemma allows this problem to be solved by solving a single LP:

Lemma 5.2. *Given Problem 5.1, and given the optimal solution $x^{\mathrm{o}}, \xi^{\mathrm{o}}$ of the following optimization problem*

$$\min_{x,\xi} \; \xi \quad \text{s.t.} \; \left[\begin{array}{cc} A & 0 \\ a_1^{\mathrm{T}} & -1 \end{array} \right] \left[\begin{array}{c} x \\ \xi \end{array} \right] \leq \left[\begin{array}{c} \mathbf{1} \\ 0 \end{array} \right], \; a_2^{\mathrm{T}} x = 1, \tag{5.8}$$

then $\eta = \frac{1}{\xi^{\mathrm{o}}}$ is a solution to Problem 5.1 if $\xi^{\mathrm{o}} > 0$.

Proof: Due to the assumption of Problem 5.1 that $\nu > 1$ it is guaranteed that there exists a vector x^{f} such that $[x;\; \xi] = [x^{\mathrm{f}};\; 1]$ is strictly feasible for (5.8) and hence $\xi^{\mathrm{o}} < 1$. By rewriting the constraints of (5.8) into the form of (5.7) it is clear that x^{o} is a feasible solution to (5.7) for $\eta = \frac{1}{\xi^{\mathrm{o}}}$ if $\xi^{\mathrm{o}} > 0$. Furthermore, by writing the optimality conditions of (5.8), it can be seen that x^{o} also is an optimal solution to (5.7) for $\eta = \frac{1}{\xi^{\mathrm{o}}}$. Due to the constraint $a_2^{\mathrm{T}} x = 1$ of (5.8) it is clear that $\nu' = 1$, which proves correctness of the lemma if $\xi^{\mathrm{o}} > 0$. □

The above lemma provides an efficient method for checking whether tightening a given constraint, can eliminate a branch in the constraint tree. In the following sections this method will be referred to as *pruning*.

5.2.2.2 Algorithm formulation using pruning

In this section constraint tightening is incorporated into Algorithm 2.4 in order to construct invariant sets with a reduced number of constraints compared to the MAS constructed using Algorithm 2.4. For simplicity the Algorithm will only be given for the case $r = 2$, but extensions to $r > 2$ are straightforward.

Algorithm 5.2 ($\langle\Omega'\rangle$-invariant set construction using pruning). *Given a system* (2.1),(2.2) *subject to constraints* (2.10)*, satisfying* (2.18) *for values $c = c^\star, \alpha = \alpha^\star$. Consider user-defined scalars $d_1 \geq c^\star, d_2 \in [\alpha^\star, 1)$ and $\gamma > 0$. Execute Algorithm 5.1 with the following implementation of step 2b):*

1. *Set $\mathcal{X}_i := Pre_{\langle\Omega'\rangle}(\mathcal{X}_{i-1})$ and define $\mathcal{X}_i = \{x | A_{\mathcal{X}_i} \leq \mathbf{1}\}$.*

2. *Check whether any* branching event *has occurred, meaning that two non-redundant constraints of $\mathcal{X}_i$ have the same parent constraint in $\mathcal{X}_{i-1}$. For every branching event involving constraint rows j_1 and j_2 of $A_{\mathcal{X}_i}$, check whether constraint tightening can make one of both redundant:*

(a) Use Lemma 5.2 to calculate the scalars $\eta_1 > 1, \eta_2 > 1$ (if both exist) needed to make one of the constraints redundant (with respect to $\bigcap_{j=0}^{i} \mathcal{X}_j$) by tightening the other. Assume without lack of generality that $\eta_1 < \eta_2$.

(b) If $\eta_1 \leq 1 + \gamma$ and $\eta_1 \|A_{\mathcal{X}_i}(j_1, :)\| \leq \max_k \|A_x(k, :)\| d_1 d_2^i$, then set $A_{\mathcal{X}_i}(j_1, :) := \eta_1 A_{\mathcal{X}_i}(j_1, :)$.

Convergence and correctness can be proven in a straightforward way.

Theorem 5.2 (Convergence of Algorithm 5.2). *Algorithm 5.2 terminates in a finite number of iterations.*

Proof: Due to the safeguards built into step 2b) of Algorithm 5.2 and due to the choice of d_1 and d_2, it is guaranteed that $\max_k \|A_{\mathcal{X}_i}(k, :)\| \leq \max_k \|A_x(k, :)\| d_1 d_2^i$. By means of a similar argumentation as used in the proof of Theorem 2.2 it can now be proven that Algorithm 5.2 terminates in a finite number of iterations. For details we refer to this proof. □

Theorem 5.3 (Correctness of Algorithm 5.2). *The set $\overline{\mathcal{S}}$ constructed using Algorithm 5.2 is $\langle \Omega' \rangle$-invariant.*

Proof: Due to the fact that Algorithm 5.2 is a specific implementation of Algorithm 5.1, it is guaranteed that $\overline{\mathcal{S}}$ is $\langle \Omega' \rangle$-invariant if in every iteration $\mathcal{X}_i$ is indeed constructed such that $\mathcal{X}_i \subseteq \text{Pre}_{\langle \Omega' \rangle}(\mathcal{X}_{i-1})$. This is guaranteed since $\mathcal{X}_i$ is initially constructed as $\mathcal{X}_i := \text{Pre}_{\langle \Omega' \rangle}(\mathcal{X}_{i-1})$, after which $\mathcal{X}_i$ is only made smaller, which proves the theorem. □

Figure 5.2b) shows how Algorithm 5.2 is able to construct a reduced complexity invariant set for the system depicted in Figure 5.1.

5.2.2.3 Parameter tuning

Parameters d_1 and d_2 are primarily used to impose convergence of the algorithm and need to satisfy $d_1 > c^\star$ and $d_2 \in [\alpha^\star, 1)$ in order to do so. Mostly they can be given the following default values:

$$d_1 = m_1 c^\star, \tag{5.9a}$$

$$d_2 = (1 - m_2).\alpha + m_2.1, \tag{5.9b}$$

with $m_1 = 1.5$ and $m_2 = 0.5$. Unless stated otherwise these will be the values used in all the following examples. An interesting property is that the worst-case volume reduction to be expected is proportional to $m_1^{n_x}$, where n_x is the state dimension of the system. Therefore, in cases where large volume reductions can be tolerated, larger values of m_1 can be chosen and vice-versa.

The parameter $\gamma \geq 0$ is the main parameter influencing the trade-off between low complexity and high volume since it defines an upper bound on the factor with which constraints can be tightened in every iteration. Larger values of γ lead to lower complexity invariant sets, while smaller values for γ lead to larger invariant sets. A good starting point usually is $\gamma \approx 0.1$. Systems with a larger amount of uncertainty typically require larger values for γ to obtain equally large complexity reductions.

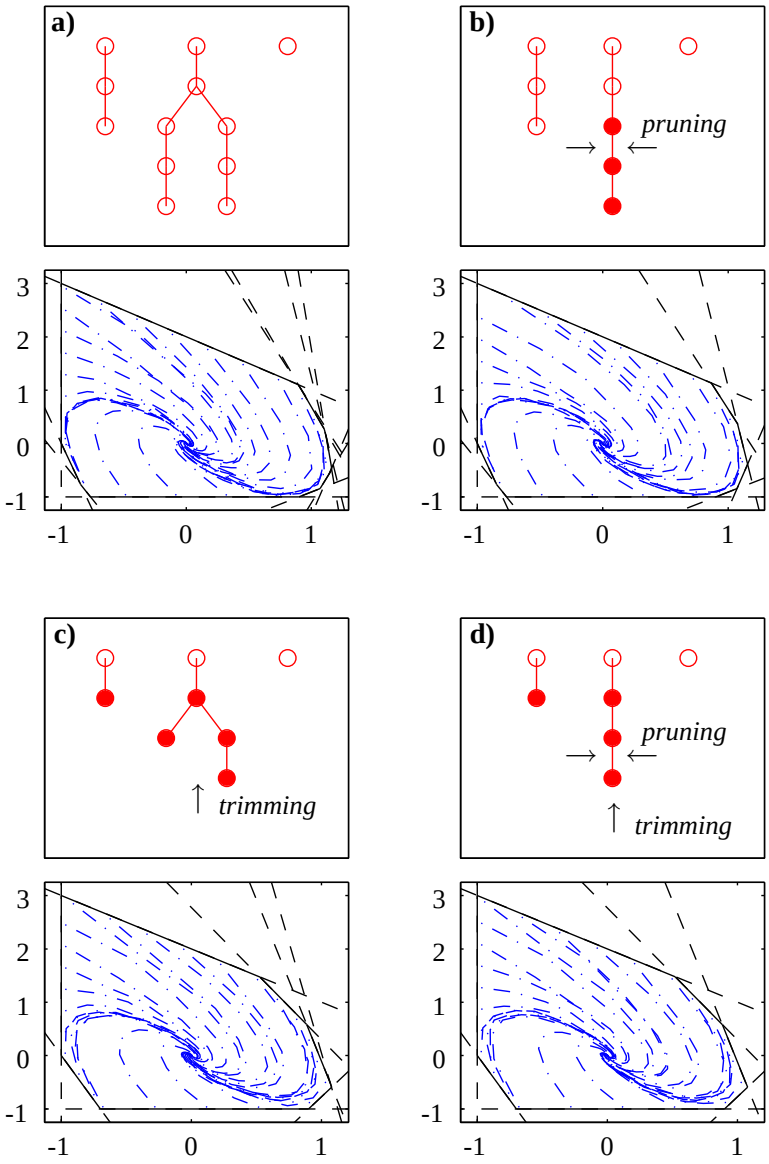


Figure 5.2: Results of pruning and trimming when applied to the system discussed in Figure 5.1. **Top subfigures:** Constraint trees of the resulting invariant sets. Solid circles depict constraints that are modified with respect to the MAS (i.e. figure a)). **Bottom subfigures:** Invariant sets along with trajectories starting near the boundaries of the sets. **Top left to bottom right: a)** the MAS of the system, **b)** invariant set obtained using *pruning* (cfr. Algorithm 5.2) with $\gamma = 0.1$, **c)** the MAS obtained using *trimming* (cfr. Section 5.2.3) with $c = 0.5$, **d)** result obtained using both pruning and trimming using identical parameters as in figures b) and c).

5.2.3 Reducing tree depth: trimming

In this section a new theorem is introduced that adds an additional tuning parameter to the algorithm for constructing reduced complexity invariant sets. It is shown that, given an LPV system, it is allowed to modify the system in a specific way and calculate the invariant set for the modified system. Under certain conditions the invariant set for the modified system is also invariant with respect to the original system. By tuning the involved parameter, one can potentially obtain invariant sets with a lower admissibility index $k^\star$. In the following sections this method will be referred to as *trimming*.

Theorem 5.4 (Invariant sets for modified system matrices). *Given a set $\mathcal{S} \in \mathbb{R}^{n_x}$ and two autonomous LPV systems defined by uncertainty polytopes Ω_1 and Ω_2 that are defined as*

$$\Omega_1 \triangleq \mathrm{Co}\{\Phi_1, \ldots, \Phi_r\}, \tag{5.10a}$$

$$\Omega_2 \triangleq \mathrm{Co}\{\Phi'_1, \ldots, \Phi'_r\}, \tag{5.10b}$$

with

$$\Phi'_i = (1+c)\Phi_i - cI, \qquad i = 1, \ldots, r, \tag{5.11}$$

where $c \in \mathbb{R}^+$. If $\mathcal{S}$ is convex and $\langle\Omega_2\rangle$-invariant, then it is also $\langle\Omega_1\rangle$-invariant.

Proof: If $\mathcal{S}$ is $\langle\Omega_2\rangle$-invariant then

$$\Phi'_i x \in \mathcal{S}, \qquad \forall x \in \mathcal{S}, i = 1, \ldots, r,$$

or equivalently

$$((1+c)\Phi_i x - cx) \in \mathcal{S}, \qquad \forall x \in \mathcal{S}, i = 1, \ldots, r.$$

Due to convexity of $\mathcal{S}$ and the fact that $c > 0$ it is therefore also guaranteed that

$$\left(\frac{1}{1+c}((1+c)\Phi_i x - cx) + \frac{c}{1+c}x\right) \in \mathcal{S}, \forall x \in \mathcal{S}, i = 1, \ldots, r,$$

which, after some algebraic manipulation, is equivalent to

$$\Phi_i x \in \mathcal{S}, \qquad \forall x \in \mathcal{S}, i = 1, \ldots, r,$$

which proves that $\mathcal{S}$ is $\langle\Omega_1\rangle$-invariant. □

The above theorem indicates that, given a system defined by uncertainty polytope Ω_1, one can calculate an invariant set $\mathcal{S}$ for the system defined by Ω_2, which is then invariant for both systems. The set $\mathcal{S}$ will be a subset of the MAS for the original system. The parameter c can be used to reduce the number of constraints of the resulting set $\mathcal{S}$.

The theorem can be further generalized by, among other things, allowing different values of c for each Φ_i-matrix.

Figure 5.2c) shows how Theorem 5.4 allows the construction of a reduced complexity invariant set for the system depicted in Figure 5.1. Figure 5.2d) shows the results when combining this theorem with Algorithm 5.2.

5.2.3.1 Parameter tuning

The above described method for computing reduced complexity invariant sets based on modified system matrices is especially useful for systems where the eigenvalues of the Φ_i matrices lie close to the point $1+0i$ on the unit circle, which is often the case in real-life examples. One can see that increasing the value of c moves the eigenvalues of the Φ_i'-matrices away from the unity point, which can intuitively be seen as accelerating the dynamics of the system. As a result one can see that c influences the JSR which in turn influences $k^\star$. In this way the number of constraints of the resulting invariant set can be reduced at the cost of a decrease in volume.

In order to reduce the number of constraints without taking the volume reduction into account one can obtain a value for c as

$$\min_{c} \ \hat{\rho}(\Omega_2).$$

However, since the computation of $\hat{\rho}(\Omega_2)$ for a single value of c is already an NP-hard problem, the above optimization problem cannot be solved in polynomial time. However, it is possible to minimize the ellipsoid norm approximation [19, 137] of the JSR instead:

$$\begin{aligned} \min_{c, P \in \mathbb{S}^{n_x}_{++}} \quad & \gamma, \\ \text{s.t.} \quad & \|\Phi_i'\|_P \leq \gamma, \qquad i = 1, \ldots, r. \end{aligned}$$

The ellipsoid norm can be computed by solving an SDP [24], and hence the optimal value of c can be found by means of an interval reduction method.

5.3 Linear scaling of $\mathcal{P}$-RMPC

This section shows that under certain conditions, the number of constraints of $\mathcal{P}$-RMPC can be reduced from an exponentially increasing number (as a function of N) to a linearly increasing number, when using Algorithm 5.2 instead of Algorithm 2.4 for computing the invariant set for the augmented system (4.63)-(4.64), used in the design phase of Algorithm 4.7 ($\mathcal{P}$-RMPC). These conditions essentially come down to the fact that the amount of model uncertainty must be sufficiently small and that the parameters of Algorithm 5.2 should be chosen appropriately.

In order to

We first derive bounds on the tightening factors obtained using Lemma 5.2 by making use of the structure present in (4.63)-(4.64) in order to come to a quantification of the performance of Algorithm 5.2 in this context. This chapter is of high importance in this thesis, because it combines results from Chapters 2 and 4 and Sections 5.1.2 and 5.2.2 and gives insight into the factors determining the efficiency of Algorithm 5.2

Lemma 5.3. *Given an LPV system* (3.1)-(3.2) *subject to constraints* (2.10)-(2.11) *and a stabilizing feedback gain* K*. Consider the application of Lemma 5.2 for tightening a constraint* $a_1^{\mathrm{T}} x \geq 1$ *in order to make a constraint* $a_2^{\mathrm{T}} x \leq 1$ *redundant during an arbitrary iteration* i *of Algorithm 5.2 when applied to* (4.63)-(4.64)*. Furthermore*

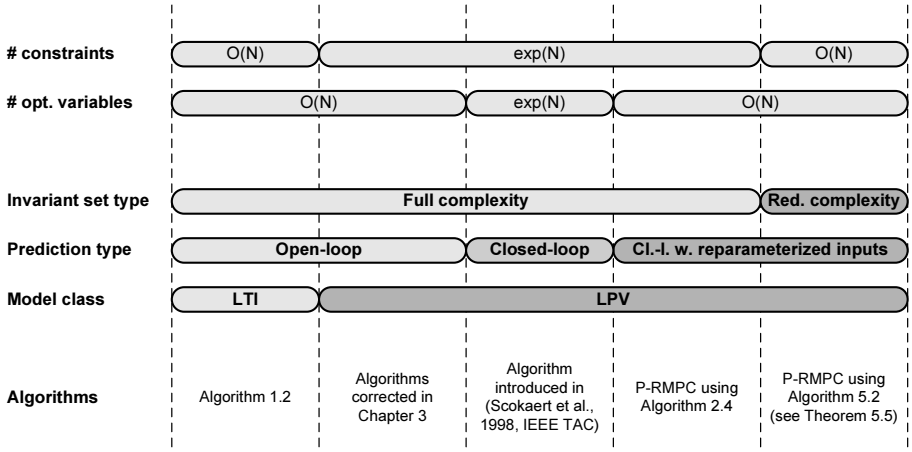


Figure 5.3: Evolution of the computational complexity of different MPC algorithms. In the LTI case one automatically has linear scaling behavior. In the LPV case one obtains exponential scaling behavior if no special complexity reduction measures are taken. Using reparameterized inputs (see Figure 4.12) one can reduce the number of optimization variables to a linearly scaling number, while using reduced-complexity invariant sets results in a linearly scaling number of constraints.

consider the fact that both constraints are children *of constraint* $b^{\mathrm{T}}x \leq 1$*, with* $b = [\overline{b};\ \underline{b}], \overline{b} \in \mathbb{R}^{n_x}, \underline{b} \in \mathbb{R}^{N \cdot n_u}$*. In that case the tightening factor* η *obtained with Lemma 5.2 is upper bounded by*

$$\eta \leq \frac{1}{1 - \sqrt{m}\left(1 + \frac{\|\lambda^{(+)}\|}{\min \lambda^{(+)}}\right) \|C^{\dagger}\| \|U\| \|\overline{b}\|}, \tag{5.12}$$

if the denominator is strictly positive, with

$$C = \begin{bmatrix} A_x^{\mathrm{T}} & -K^{\mathrm{T}} A_u^{\mathrm{T}} \\ 0 & A_u^{\mathrm{T}} \end{bmatrix}, \qquad U = \begin{bmatrix} (\Phi_2 - \Phi_1)^{\mathrm{T}} \\ (B_2 - B_1)^{\mathrm{T}} \end{bmatrix}, \tag{5.13}$$

$\lambda^{(+)}$ *a strictly positive vector satisfying* $C\lambda^{(+)} = U\overline{b}$*,* m *denoting the number of rows of* C *and* $\Phi_i = A_i - B_i K$*.*

Proof: See Appendix D. □

Expression (5.12) can be expected to be very conservative due to the many approximations made in Section D.1 when constructing this bound. The real tightening factor found using Lemma 5.2 is in most cases significantly smaller. Therefore, this upper bound only is of limited practical use, but shows that *a* bound exists that is independent of N. This property is used in the following theorem.

Theorem 5.5. *Given an LPV system* (3.1)-(3.2) *subject to constraints* (2.10)-(2.11) *and a stabilizing feedback gain* K*. If* γ *is chosen such that*

$$\gamma \in \left(\overline{c} - 1, d_2^{-1} - 1\right), \tag{5.14}$$

with $\overline{c}$ equal to the RHS of (5.12)*, then the following property holds:*

$$\textit{rows}(A_{\mathcal{S}}) = O(N), \tag{5.15}$$

with $\mathcal{S} = \{x | A_{\mathcal{S}} x \leq \mathbf{1}\}$ an invariant set for system (4.63)-(4.64) *computed using Algorithm 5.2.*

Proof: See Appendix D. □

The above theorem shows that if the amount of uncertainty is sufficiently small, the parameters of Algorithm 5.2 are chosen appropriately and the interval given by (5.14) is non-empty, the number of constraints of $\mathcal{P}$-RMPC increases as a linear function of N. Figure 5.3 sketches the broader context within which this result should be interpreted.

It is important to emphasize that if the above conditions are not satisfied (e.g., if the interval (5.14) is empty) one can still observe linear scaling behavior, or exponential scaling behavior with a lower base number. In most situations, as illustrated in the next section, the additional computational cost of verifying whether constraint tightening can be applied is relatively small.

Therefore, the main conclusion of this section is that in most cases there is no reason why not to use Algorithm 5.2 instead of Algorithm 2.4, especially when computing constraints for $\mathcal{P}$-RMPC.

5.4 Examples

Two sets of examples are provided. First the Monte-Carlo experiment discussed in Section 5.1.1 is repeated using Algorithm 5.2 (pruning) and the method described in Section 5.2.3 (trimming). Secondly, the examples discussed in Sections 4.2.7 and 4.3.5 are also repeated using the new algorithms in order to evaluate whether an improved trade-off is obtained between the number of on-line constraints and the obtained feasible regions.

5.4.1 Monte-Carlo experiment

For easy reference, we first summarize the results of the Monte-Carlo experiment discussed in Section 5.1.1. The experiment showed that for random LPV systems the number of constraints describing polyhedral invariant sets constructed using Algorithm 2.4 increases exponentially as a function of the dimensionality of the system, even if the convergence rate of the systems is kept fixed by construction. This was shown to be due to an increase of the admissibility index $k^\star$ as a function of the dimensionality. Expression (2.23) shows that the worst-case number of constraints describing polyhedral invariant sets increases exponentially as a function of $k^\star$, which explains the observed scaling behavior.

First, we recompute invariant sets using Algorithm 5.2 (pruning) with $\gamma = 0.1$. Results are presented in Table 5.2 and can be compared to the results shown in Table 5.1. A significant reduction of the number of constraints is obtained. Also the average computation time needed for the construction of the sets is reduced significantly,

enabling the construction of invariant sets for higher-dimensional systems is reasonable amounts of time.

Table 5.3 shows results obtained when using trimming. The parameter c was optimally chosen from $\{0, 0.1, 0.3, 1, 3, 10\}$ in order to minimize the JSR of the resulting modified systems. Also in this case significantly reduced numbers of constraints and significantly lower computation times were obtained. As expected trimming significantly reduces the admissibility index $k^\star$.

Table 5.4 finally shows results obtained when combining pruning and trimming. As expected this results in the lowest complexity invariant sets, but due to the small average values of $k^\star$ the further reduction is relatively small.

A volume comparison is not given here, since computing volumes in high-dimensional spaces is non-trivial, and volumes in different dimensions are difficult to compare. The next example lends itself better for volume comparisons because there the volume of the resulting feasible regions can be compared, even for different state dimensions of the involved augmented systems.

5.4.2 Robust MPC using reduced-complexity invariant sets

In this section we reconsider the example discussed in Section 4.3.5 and now compute the invariant sets using the new algorithms introduced in this chapter. The aim is to investigate whether the new algorithms can provide a better trade-off between the obtained feasible region and the number of constraints to be used in the on-line optimization problems.

Figure 5.4 compares the feasible regions reported in Figure 4.14 with those obtained using Algorithm 5.2 with $\gamma = 0.1$. The feasible regions obtained with the new algorithm are slightly smaller and are not nested for decreasing values of N. However,

	rows$(A_{\overline{\mathcal{S}}})$	$\mathbf{k}^\star$	**CPU-t. (s)**	$\mathbf{r_{equiv}}$
$\mathbf{n_x = 2}$	10.36	4.21	10.57	0.48
$\mathbf{n_x = 3}$	19.29	6.38	14.67	0.62
$\mathbf{n_x = 4}$	29.13	7.35	19.22	0.71
$\mathbf{n_x = 5}$	42.57	8.15	26.90	0.76
$\mathbf{n_x = 6}$	57.21	9.12	37.85	0.80
$\mathbf{n_x = 7}$	76.88	9.43	52.47	0.84
$\mathbf{n_x = 8}$	95.70	9.69	67.84	0.87
$\mathbf{n_x = 9}$	116.59	10.17	87.26	0.88
$\mathbf{n_x = 10}$	141.84	10.22	114.94	0.90
$\mathbf{n_x = 11}$	171.95	10.79	151.38	0.92
$\mathbf{n_x = 12}$	197.61	10.73	183.84	0.93

Table 5.2: Results of the application of Algorithm 5.2 (pruning) with $\gamma = 0.1$ to the sets used in the Monte-Carlo experiment discussed in Section 5.1.1. Average number of constraints, average tree depth, average computation time and average equivalent branching factor of invariant sets for 100 randomly generated systems are reported for $n_x = 2, \ldots, 12$.

	rows$(A_{\overline{\mathcal{S}}})$	$\mathbf{k}^\star$	**CPU-t. (s)**	$\mathbf{r_{equiv}}$
$\mathbf{n_x = 2}$	5.36	1.08	9.03	0.28
$\mathbf{n_x = 3}$	9.47	1.37	10.52	0.47
$\mathbf{n_x = 4}$	14.65	1.56	11.84	0.61
$\mathbf{n_x = 5}$	21.32	1.74	14.08	0.75
$\mathbf{n_x = 6}$	27.15	1.82	16.68	0.79
$\mathbf{n_x = 7}$	35.39	1.95	20.00	0.88
$\mathbf{n_x = 8}$	43.82	2.08	24.10	0.90
$\mathbf{n_x = 9}$	51.10	2.00	27.16	0.94
$\mathbf{n_x = 10}$	64.70	2.30	32.71	1.00

Table 5.3: Results of the application of trimming (with c optimally chosen from $\{0, 0.1, 0.3, 1, 3, 10\}$) to the sets used in the Monte-Carlo experiment discussed in Section 5.1.1. Average number of constraints, average tree depth, average computation time and average equivalent branching factors of invariant sets for 100 randomly generated systems are reported for $n_x = 2, \ldots, 10$.

	rows$(A_{\overline{\mathcal{S}}})$	$\mathbf{k}^\star$	**CPU-t. (s)**	$\mathbf{r_{equiv}}$
$\mathbf{n_x = 2}$	5.19	1.08	9.03	0.26
$\mathbf{n_x = 3}$	8.65	1.35	10.48	0.37
$\mathbf{n_x = 4}$	13.07	1.59	11.80	0.48
$\mathbf{n_x = 5}$	18.39	1.73	13.93	0.59
$\mathbf{n_x = 6}$	23.24	1.84	16.50	0.63
$\mathbf{n_x = 7}$	29.73	1.96	19.69	0.70
$\mathbf{n_x = 8}$	36.02	2.04	23.29	0.73
$\mathbf{n_x = 9}$	43.69	2.15	27.78	0.78
$\mathbf{n_x = 10}$	52.27	2.28	32.36	0.82
$\mathbf{n_x = 11}$	59.52	2.20	36.71	0.86
$\mathbf{n_x = 12}$	67.28	2.22	42.00	0.89

Table 5.4: Results of the application of trimming (with c optimally chosen from $\{0, 0.1, 0.3, 1, 3, 10\}$) and pruning (using $\gamma = 0.1$) to the sets used in the Monte-Carlo experiment discussed in Section 5.1.1. Average number of constraints, average tree depth, average computation time and average equivalent branching factors of invariant sets for 100 randomly generated systems are reported for $n_x = 2, \ldots, 12$.

	#constr.	Vol.
$\mathcal{P}$-**RMPC,** $N = 0$	10	27.81
$\mathcal{P}$-**RMPC,** $N = 1$	16	37.42
$\mathcal{P}$-**RMPC,** $N = 2$	26	46.41
$\mathcal{P}$-**RMPC,** $N = 3$	47	59.58
$\mathcal{P}$-**RMPC,** $N = 4$	74	68.00
$\mathcal{P}$-**RMPC,** $N = 5$	120	78.24
$\mathcal{P}$-**RMPC,** $N = 6$	171	87.25
$\mathcal{P}$-**GIMPC**	19	78.85
$\mathcal{P}$-**GIMPC2**	28	176.06

Table 5.5: Number of constraints and volume of the feasible region obtained for the same example described in Section 4.3.5 and Table 4.3. The results in this table are obtained by using Algorithm 5.2 (with $\gamma = 0.1$) for computing the invariant sets.

as Table 5.5 shows, the number of constraints is also significantly reduced compared to those reported in Table 4.3. By comparing both tables, it becomes clear that with Algorithm 5.2 and $N = 6$ a similar feasible region is obtained as with Algorithm 2.4 and $N = 5$, but the former results in a lower number of constraints. The difference is not dramatic in this case because there is a significant amount of model uncertainty, with up to 100% uncertainty on some coefficients. Table 5.6 and Figure 5.5 show the results obtained is also trimming is applied with $c = 0.25$. The number of constraints is further reduced, but the trade-off between complexity and volume of the feasible region is not further improved for the $\mathcal{P}$-RMPC algorithm.

It should be noted that, although Section 5.3 only gives guarantees for the complexity of $\mathcal{P}$-RMPC, the algorithms described in this chapter also seem very effective when

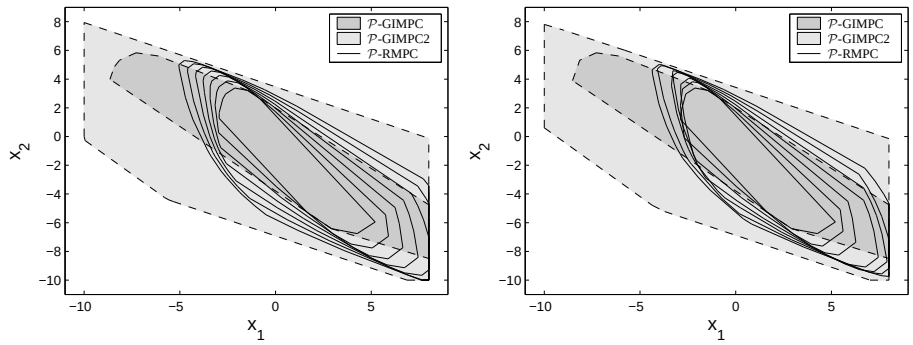


Figure 5.4: Left: Feasible regions of the different algorithms presented in Chapter 4 with invariant sets computed using Algorithm 2.4 as also depicted in Figure 4.14. **Right:** Feasible regions of the same algorithms using invariant sets computed using Algorithm 5.2 with $\gamma = 0.1$. Table 5.5 shows the number of on-line inequality constraints and the volume of the feasible regions.

used for constructing invariant sets for $\mathcal{P}$-GIMPC2. In this example the number of on-line constraints is reduced from 63 to 23, while the volume of the feasible region is only slightly reduced (from 181.82 to 165.56). Figure 5.6 shows the constraint structures of the invariant sets of $\mathcal{P}$-GIMPC2 obtained using three different methods. This shows clearly that pruning reduces the number of parallel branches of the constraint tree and that trimming reduces the tree depth, as already explained.

Finally, we investigate the behavior of Algorithm 5.2 for different values of γ and for different amounts of uncertainty. Figure 5.7 shows the relative number of on-line constraints and the relative volume of the feasible region as a function of γ for the $\mathcal{P}$-RMPC algorithm with $N = 6$. As expected, increasing values of γ lead to a reduced number of constraints and a lower volume of the corresponding feasible region. For two different values of γ the constraints structures are depicted, showing that higher γ-values result in less parallel branches.

Figure 5.8 shows the behavior of Algorithm 5.2 (with $\gamma \in \{0, 0.1\}$) for different amounts of model uncertainty. The amount of model uncertainty is varied by considering a modified system with $\Phi_1' = \Phi_1$ and $\Phi_2' = (1-a)\Phi_1 + a\Phi_2$, with $a \in [0, 2]$. The number of constraints and the volume of the feasible region are normalized with respect to $a = 0$. The figure shows that for smaller amounts of model uncertainty Algorithm 5.2 becomes increasingly efficient in reducing the number of constraints if $\gamma > 0$. Also, as the amount of model uncertainty approaches 0, the number of constraints approaches 27, which is identical to the number of constraints describing the MAS in the LTI case. This observation is in line with the general intuition that small amounts of model uncertainty should not substantially increase the off- and on-line computational burden, which is not what is obtained with Algorithm 2.4, which corresponds to the case $\gamma = 0$.

Figure 5.9 shows that Algorithm 5.2 leads to a linearly increasing number of con-

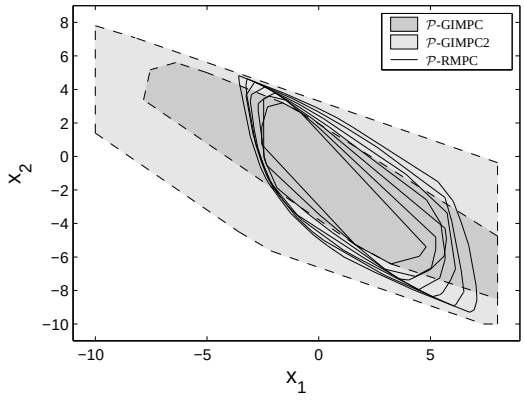
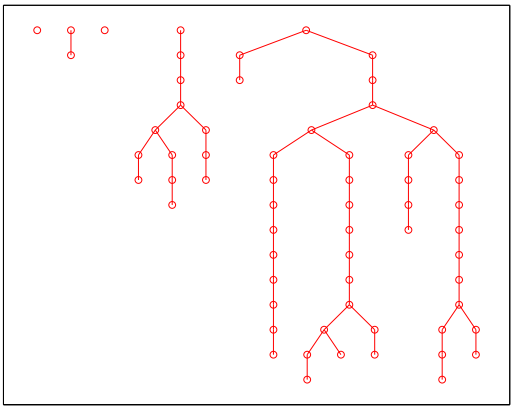
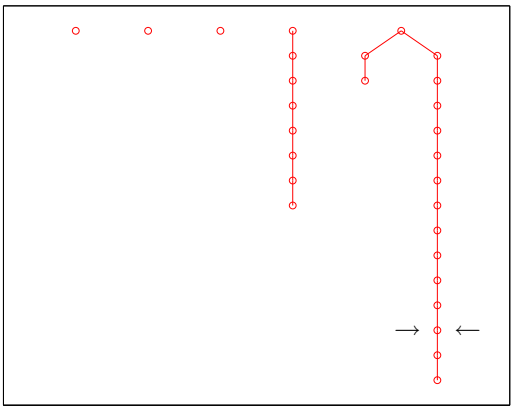


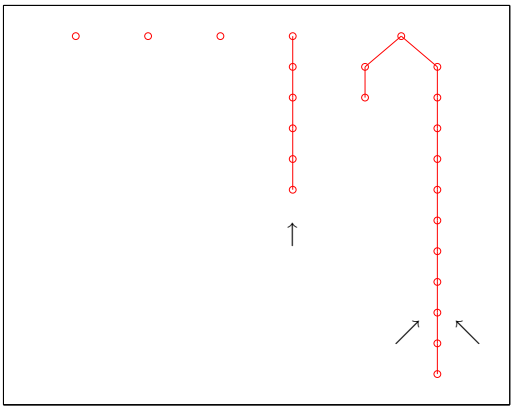
Figure 5.5: Feasible regions obtained for the same setting as those shown in Figure 5.4. In this figure both trimming ($c = 0.25$) and pruning ($\gamma = 0.1$) are used to construct the involved invariant sets. Table 5.6 shows the number of on-line inequality constraints and the volume of the feasible regions.



(a) No trimming, no pruning.



(b) No trimming, pruning with $\gamma = 0.1$.



(c) Trimming with $c = 0.25$, pruning with $\gamma = 0.1$.

Figure 5.6: Constraint structures of the polyhedral invariant sets used in $\mathcal{P}$-GIMPC2 for the settings depicted in Figures 4.8, 5.4 and 5.5.

straints when applied for computing constraints for $\mathcal{P}$-RMPC. Also the computation times reduce dramatically when using Algorithm 5.2 instead of Algorithm 2.4, which enables the use of significantly larger horizon lengths. The ability to use significantly larger horizon lengths allows the volume reductions caused by the constraint tightening to be more than compensated, which is shown in more detail in Chapter 8.

5.5 Conclusions

In this chapter new algorithms for the construction of polyhedral invariant sets are discussed. The obtained sets are invariant inner approximations of the MAS and allow the user to make a trade-off between low complexity and a large volume.

Two different methods were discussed: *pruning* and *trimming*. Both can be interpreted in terms of the constraint structure of the obtained invariant sets. Pruning aims to reduced the number of parallel branches in the constraint structure, while trimming aims to reduce the depth of the constraint structure. An additional important advantage is the reduced computation time needed to construct these reduced-complexity invariant sets, which is due to the fact that the size of the optimization problems that have to be solved for constructing these sets is reduced.

The algorithms are demonstrated using several examples, which show that in several cases significant complexity reductions can be obtained. When used in robust MPC algorithms, these reduced-complexity invariant sets can lead to better trade-offs between the size of the feasible region and the on-line computational complexity. In some cases these improvements can be orders of magnitude.

Finally, it should be noted that the proposed algorithms are only initial steps in the direction of constructing reduced complexity invariant sets. Extensions towards different model classes (PWA, hybrid, ...) are possible, as well as other ways for reducing the number of constraints within the framework laid out by Algorithm 5.1. Also, the formulation of expressions for the a priori quantification of the obtainable complexity reductions is a useful and interesting future research direction.

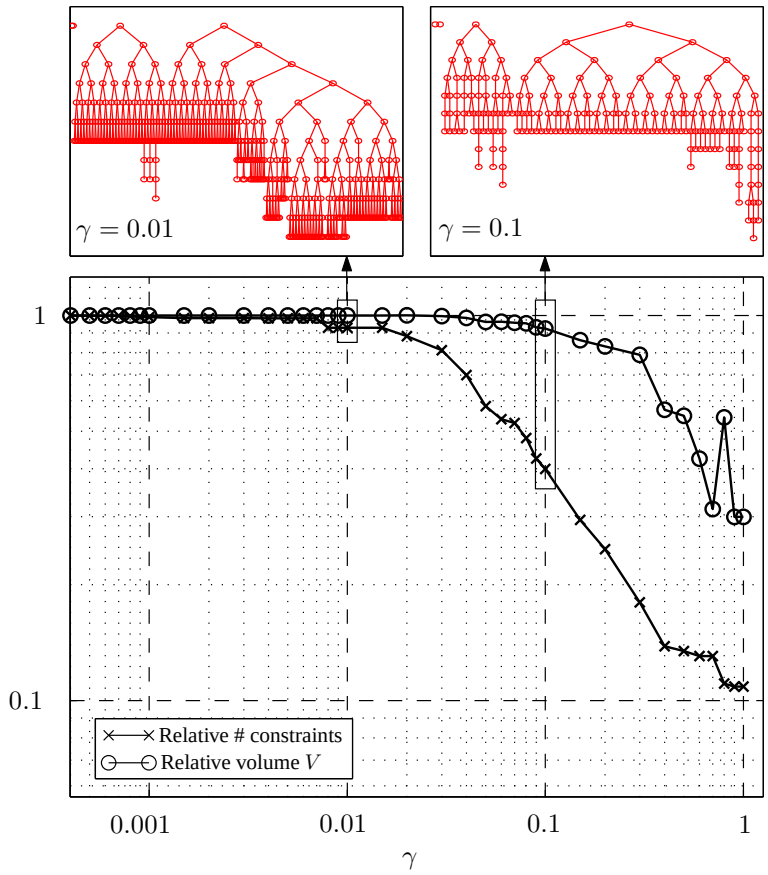


Figure 5.7: Relative number of constraints and relative volume V of the corresponding feasible region (both with respect to the MAS, i.e $\gamma = 0$) of invariant sets for $\mathcal{P}$-RMPC with $N = 6$ for the example discussed in Section 4.3.5. The constraint structures of the invariant sets corresponding to $\gamma = 0.01$ and $\gamma = 0.1$ are depicted at the top of the figure.

	#const.	Vol.
$\mathcal{P}$-**RMPC,** $N = 0$	9	25.75
$\mathcal{P}$-**RMPC,** $N = 1$	14	34.39
$\mathcal{P}$-**RMPC,** $N = 2$	27	41.87
$\mathcal{P}$-**RMPC,** $N = 3$	46	49.29
$\mathcal{P}$-**RMPC,** $N = 4$	69	56.16
$\mathcal{P}$-**RMPC,** $N = 5$	101	62.12
$\mathcal{P}$-**RMPC,** $N = 6$	144	70.40
$\mathcal{P}$-**GIMPC**	17	75.51
$\mathcal{P}$-**GIMPC2**	23	165.56

Table 5.6: Number of constraints and volume of the feasible region obtained for the same example described in Section 4.3.5 and Table 4.3. The results in this table are obtained by using trimming (with $c = 0.25$) and Algorithm 5.2 (with $\gamma = 0.1$) for computing the invariant sets.

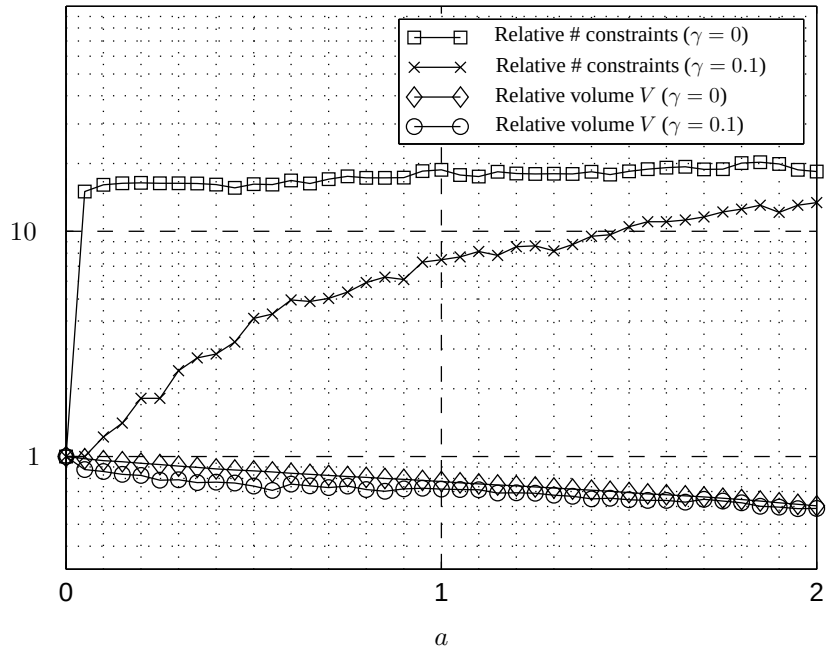


Figure 5.8: Relative number of constraints and relative volume V of the corresponding feasible region of invariant sets for $\mathcal{P}$-RMPC with $N = 6$ for the example discussed in Section 4.3.5 as a function of the amount of uncertainty. The amount of model uncertainty is varied by considering a modified system with $\Phi'_1 = \Phi_1$ and $\Phi'_2 = (1 - a)\Phi_1 + a\Phi_2$, with $a \in [0, 2]$. Values are normalized with respect to $a = 0$. Algorithm 5.2 with $\gamma \in \{0, 0.1\}$ is used to construct the invariant sets.

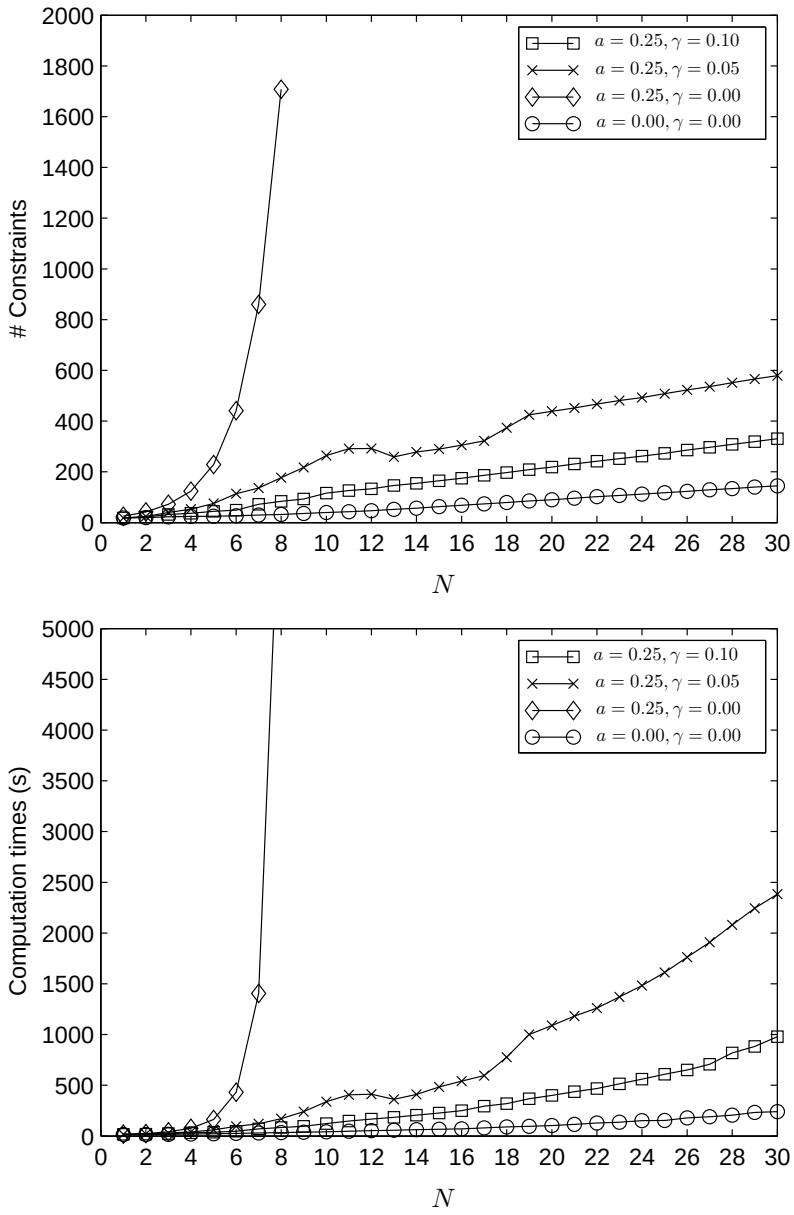


Figure 5.9: Number of constraints and computation times of invariant sets for $\mathcal{P}$-RMPC as a function of N for the example discussed in Section 4.3.5. The different curves represent different amounts of uncertainty (determined by a as in Figure 5.8) and different values of γ. Algorithm 5.2 is used to construct the invariant sets. Algorithm 5.2 clearly leads to a linearly increasing number of constraints as a function of N, if γ is given a strictly positive value or if $a = 0$. A corresponding reduction in the computation times is observed.

Chapter 6

Reduced-Complexity Control Invariant Sets

"One should not increase, beyond what is necessary, the number of entities required to explain anything."

– William of Ockham (1285-1349) –

In this chapter we extend the results of Chapter 5 from positive invariant sets to control invariant sets. First we show that the techniques of pruning *and* trimming *can also be extended to this more general context in order to obtain reduced-complexity sets. Secondly, we show how pruning can be applied to* Fourier-Motzkin elimination *in order to calculate reduced-complexity inner, un-biased or outer approximations of projections of polyhedral sets. These new techniques for projecting polyhedral sets can either be used to further reduce the complexity of control invariant sets or can be used for any other application involving the projection of H-polytopes or Fourier-Motzkin elimination.*

6.1 Control invariant sets

Before discussing reduced-complexity control invariant sets, it is necessary to introduce the necessary definitions and the state of the art concerning the construction of such sets, after which some basic properties are discussed.

While positive invariant sets are related to constrained autonomous dynamic systems, control invariant sets are related to constrained dynamic systems with inputs and can be defined as follows.

Definition 6.1 (**Control invariant set**, [14])**.** *Given an LPV system* (3.1)-(3.2) *subject to input and state constraints* (2.10)-(2.11) *and given* $\lambda \in (0, 1)$*, then the set* $\mathcal{S} \in \mathbb{R}^{n_x}$

is a feasible λ-contractive control invariant set (with $\lambda \in (0,1)$) iff $\mathcal{S} \subseteq \mathcal{X}$ and

$$\forall x \in \mathcal{S}, \exists u(x) \in \mathcal{U}: \quad Ax + Bu \in \lambda\mathcal{S}, \forall [A\ B] \in \Omega, \tag{6.1}$$

In what follows we will omit the term *feasible* unless where we want to emphasize that $\mathcal{S} \subseteq \mathcal{X}$. For briefness of notation, in future sections we will refer to these sets as $\langle \Omega, \mathcal{U}, \lambda \rangle$-invariant sets or, if $\lambda = 1$, $\langle \Omega, \mathcal{U} \rangle$-invariant sets. We will not explicitly consider bounded disturbances in this chapter, but this extension is relatively straightforward and similar to the method described in Section 2.4.2.

Control invariant sets can be seen as the equivalent concept of positive invariant sets for systems with inputs. These inputs provide additional degrees of freedom that can be used to keep states inside the set and therefore the system (3.1)-(3.2) does not necessarily be open-loop stable in order to have a control invariant set. Since the next chapter will use control invariant sets for guaranteeing constraint satisfaction, we are primarily interested in the largest possible control invariant set for a given system:

Definition 6.2 (**Maximal control invariant set (MCAS)**, [13])**.** *A set $\overline{\mathcal{S}}$ is the maximal $\langle \Omega, \mathcal{U}, \lambda \rangle$-invariant set iff it is $\langle \Omega, \mathcal{U}, \lambda \rangle$-invariant and all other feasible $\langle \Omega, \mathcal{U}, \lambda \rangle$-invariant sets are subsets of $\overline{\mathcal{S}}$.*

Similar to the MAS, also the MCAS can be proven to exist and to be uniquely defined in this way. As already pointed out by [63, Proposition 2.5], control invariant sets provide a powerful tool for robust constraint satisfaction, since for any initial state $x(0)$ one can show that there exists a control law $\mathcal{U} \ni u(k) = \kappa(x(k)), k \in \mathbb{N}$ that guarantees constraint satisfaction $\forall k \in \mathbb{N}$ iff $x(0) \in \overline{\mathcal{S}}$.

Similar to positive invariance, also control invariance can be expressed in terms of a geometric condition based on the pre-set defined in Chapter 2:

Lemma 6.1 (**Geometric condition for control invariance**, [63])**.** *Given an LPV system* (3.1)-(3.2) *subject to input and state constraints* (2.10)-(2.11) *and given $\lambda \in (0,1)$ and a set $\mathcal{S} \in \mathbb{R}^{n_x}$, then the set $\mathcal{S}$ is $\langle \Omega, \mathcal{U}, \lambda \rangle$-invariant iff*

$$\mathcal{S} \subseteq \mathit{Pre}_{\langle \Omega, \mathcal{U}, \lambda \rangle}(\mathcal{S}), \tag{6.2}$$

where, for reasons of consistency with the notation of Chapter 2, we define the operator $\text{Pre}_{\langle \Omega, \mathcal{U}, \lambda \rangle}(\cdot)$ as $\text{Pre}_{\langle \Omega, \mathcal{U}, \lambda \rangle}(\mathcal{S}) \triangleq \text{Pre}_{\langle \Omega, \{0\}, \mathcal{U}, \lambda \rangle}(\mathcal{S})$.

Proof: The proof is similar to the proof of Lemma 2.1 and is hence omitted. □

Checking whether a given polyhedral set $\mathcal{S} = \{x | A_{\mathcal{S}} x \leq \mathbf{1}\}$ satisfies condition (6.2) is a three-step procedure:

1. Calculate the set $\overline{\text{Pre}}_{\langle \Omega, \mathcal{U}, \lambda \rangle}(\mathcal{S})$[1] containing all state-input vectors $[x; u]$ that keep the next state inside $\lambda\mathcal{S}$:

$$\overline{\text{Pre}}_{\langle \Omega, \mathcal{U}, \lambda \rangle}(\mathcal{S}) \triangleq \{[x; u] \in \mathbb{R}^{n_x + n_u} | u \in \mathcal{U}, A_{\mathcal{S}}(A_i x + B_i u) \leq \lambda \mathbf{1}, i = 1, \ldots, r\}. \tag{6.3}$$

[1] In [15] the set $\overline{\text{Pre}}_{\langle \Omega, \mathcal{U}, \lambda \rangle}(\mathcal{S})$ is referred to as the "expanded" set.

2. Compute $\mathcal{S}' = \text{Pre}_{\langle\Omega,\mathcal{U},\lambda\rangle}(\mathcal{S})$ as the projection of $\overline{\text{Pre}}_{\langle\Omega,\mathcal{U},\lambda\rangle}(\mathcal{S})$ onto $\mathbb{R}^{n_x}$:

$$\text{Pre}_{\langle\Omega,\mathcal{U},\lambda\rangle}(\mathcal{S}) \equiv \text{proj}_{n_u}\big(\overline{\text{Pre}}_{\langle\Omega,\mathcal{U},\lambda\rangle}(\mathcal{S})\big), \tag{6.4}$$

   with $\text{proj}_{n_u}(\cdot)$ denoting the n_u-tuple application of $\text{proj}(\cdot)$.

3. Check whether $\mathcal{S} \subseteq \mathcal{S}'$.

The first two steps essentially correspond to the computation of $\text{Pre}_{\langle\Omega,\mathcal{U},\lambda\rangle}(\mathcal{S})$, while the third step only consists of checking the inclusion of both sets. The projection of step 2 can be performed using Fourier-Motzkin elimination [146], which is briefly described in Appendix B.

Similar to Algorithm 2.3, it is now possible to formulate an algorithm for the construction of the MCAS based on condition (6.2):

Algorithm 6.1 (**$\langle\Omega,\mathcal{U},\lambda\rangle$-invariant set construction**, [13])**.** *Given a system* (3.1)-(3.2) *subject to constraints* (2.10)-(2.11) *and variables* $\lambda, \lambda' \in \mathbb{R}^+$ *such that* $\lambda \in (0,1], \lambda' \in (0,1], \lambda' \leq \lambda$*, perform the following steps:*

1. *Initialize* $\mathcal{O}_0 := \mathcal{X}, i := 0$.
2. *Execute iteratively until* $\mathcal{O}_i \subseteq$ *Pre*$_{\langle\Omega,\mathcal{X},\mathcal{U},\lambda\rangle}(\mathcal{O}_i)$*:*
   (a) Set $i := i + 1$.
   (b) Calculate $\mathcal{O}_i :=$ *Pre*$_{\langle\Omega,\mathcal{X},\mathcal{U},\lambda'\rangle}(\mathcal{O}_{i-1}) \cap \mathcal{O}_{i-1}$.

Return the set $\overline{\mathcal{S}} \triangleq \mathcal{O}_i$.

The resulting sets have the same scalability issues as positive invariant sets, which will also become clear in the examples section:

- Based on expression (6.3), where it becomes clear that every constraint in the set $\mathcal{S}$ has the potential to generate r constraints in the set $\overline{\text{Pre}}_{\langle\Omega,\mathcal{U},\lambda\rangle}(\mathcal{S})$, it is clear that also in the case of control invariant sets the number of constraints in the worst-case can increase r-fold as the number of iterations increases.

- The projection needed for calculating the pre-set in every iteration can also increase the number of constraints dramatically, especially for higher-dimensional systems. This is due to the fact that counter-intuitively the number of constraints describing a projection of a set can be significantly higher[2] than the number of constraints describing the original set.

Apart from the increase in the complexity of the resulting sets, also the computational complexity is an important concern, since the added projection step is computationally heavy and largely determined by the number of constraints of the sets to which the operation is applied. We refer to Appendix B for more details.

The following section will now introduce several methods for decreasing the complexity of the resulting control-invariant sets, which will also directly have a positive effect on the computational complexity of the construction of these sets.

[2]The number of constraints describing the $(n-1)$-dimensional projection of an n-dimensional H-polytope described by m constraints, is in the worst case equal to $\lfloor \frac{m^2}{4} \rfloor$. The number of vertices describing the $(n-1)$-dimensional projection of an n-dimensional V-polytope described by m vertices is in the worst case equal to m. The projection of V-polytopes hence seems to be more efficient, but V-polytopes are not suited for use as constraint sets in optimization problems, such as those used in MPC . See [146] for details.

6.2 Reduced-complexity control invariant sets

In this section several methods are proposed for constructing reduced-complexity control invariant inner approximations of the MCAS. First of all the two main contributions of the previous chapter (*pruning* and *trimming*) are extended towards the setting of control invariant sets. Secondly the pruning method is also applied to the projection step involved in constructing control invariant sets. This second contribution allows the construction of reduced-complexity projections of polytopes which could also be useful in other settings.

Before describing the different methods for reducing the complexity of control invariant sets we describe a general framework similar to Algorithm 5.1 within which the new methods fit.

6.2.1 General framework

The main contribution of this section is the formulation of an algorithm for the construction of inner approximations to the MCAS, similar to Algorithm 5.1. In this way degrees of freedom are created that are then exploited in the next sections in order to obtain complexity reductions.

While Algorithm 5.1 deals with sets $\mathcal{X}_i$, with i denoting the iteration number, the algorithm introduced here deals with several different sets during a single iteration. This is due to the fact that a projection of a $(n_x + n_u)$-dimensional set to a n_x-dimensional set has to be computed. This operation is applied one dimension at a time and therefore we will consider sets $\mathcal{X}_i^{[n_x+n_u]} \subset \mathbb{R}^{n_x+n_u}, \mathcal{X}_i^{[n_x+n_u-1]} \subset \mathbb{R}^{n_x+n_u-1}, \ldots, \mathcal{X}_i^{[n_x]} \subset \mathbb{R}^{n_x}$, with again i denoting the iteration number and the superscripts denoting the dimensionality of the sets. For simplicity of notation we will also use sets $\mathcal{O}_i^{[n_x+j]} \triangleq \bigcap_{k=0}^{i} \mathcal{X}_k^{[n_x+j]}$ with $j = 0, \ldots, n_u, i \in \mathbb{N}$. This results in the following algorithm, whose work flow is depicted schematically in Figure 6.1:

Algorithm 6.2 (Approximate maximal $\langle \Omega, \mathcal{U}, \lambda \rangle$-invariant set construction). *Given a system* (3.1)-(3.2) *subject to constraints* (2.10)-(2.11) *and $\lambda \in (0, 1], \lambda' \in (0, 1], \lambda' \leq \lambda$, perform the following steps:*

1. *Initialize $\mathcal{X}_0^{[n_x+n_u]} := \mathcal{X} \times \mathcal{U}$ and set $i := 0$.*
2. *Compute $\mathcal{X}_0^{[n_x+n_u-j]}$ such that $\mathcal{X}_0^{[n_x+n_u-j]} \subseteq \mathrm{proj}\big(\mathcal{X}_0^{[n_x+n_u-j+1]}\big)$ for $j = 1, \ldots, n_u$.*
3. *Execute iteratively until $\mathcal{O}_i^{[n_x+n_u]} \subseteq \overline{Pre}_{\langle \Omega, \mathcal{U}, \lambda \rangle}(\mathcal{O}_i^{[n_x]})$:*
   (a) *Set $i := i + 1$.*
   (b) *Compute $\mathcal{X}_i^{[n_x+n_u]}$ such that $\mathcal{O}_i^{[n_x+n_u]} \subseteq \overline{Pre}_{\langle \Omega, \mathcal{U}, \lambda' \rangle}(\mathcal{O}_{i-1}^{[n_x]})$*
   (c) *Compute $\mathcal{X}_i^{[n_x+n_u-j]}$ such that $\mathcal{O}_i^{[n_x+n_u-j]} \subseteq \mathrm{proj}\big(\mathcal{O}_i^{[n_x+n_u-j+1]}\big)$ for $j = 1, \ldots, n_u$.*

Return the set $\overline{\mathcal{S}} \triangleq \mathcal{O}_i^{[n_x]}$.

If steps 2, 3b and 3c are implemented such that the inclusions are satisfied with equality – which can be done easily by respectively setting $\mathcal{X}_0^{[n_x+n_u-j]} :=$

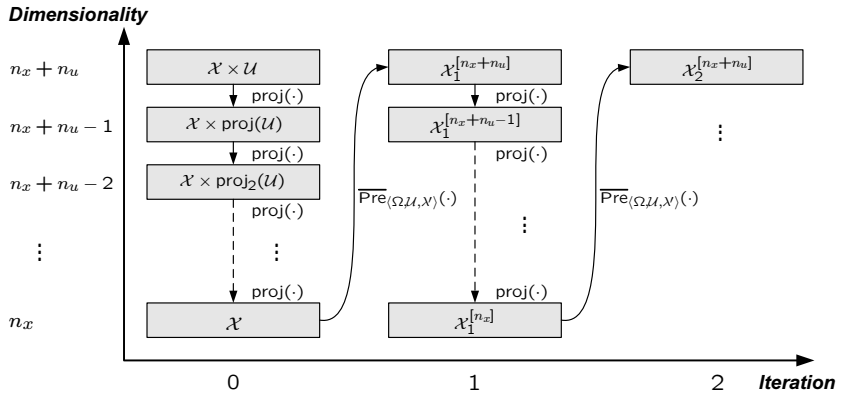


Figure 6.1: Work flow of Algorithm 6.2. The arrows indicate the order in which the different sets are constructed and based on which operators the sets are computed.

$\text{proj}\big(\mathcal{X}_0^{[n_x+n_u-j+1]}\big)$ in step 2, setting $\mathcal{X}_i^{[n_x+n_u]} := \overline{\text{Pre}}_{\langle\Omega,\mathcal{U},\lambda'\rangle}(\mathcal{X}_{i-1}^{[n_x]})$ in step 3b and setting $\mathcal{X}_i^{[n_x+n_u-j]} := \text{proj}\big(\mathcal{O}_i^{[n_x+n_u-j+1]}\big)$ in step 3c – the above algorithm can be shown to be identical to Algorithm 6.1. In the other case the resulting sets will be subsets of the ones obtained using Algorithm 6.1 but are still guaranteed to be $\langle\Omega,\mathcal{U},\lambda\rangle$-invariant.

Theorem 6.1. *Given a system* (3.1)-(3.2) *subject to constraints* (2.10)-(2.11) *and* $\lambda \in (0,1], \lambda' \in (0,1], \lambda' \leq \lambda$*, then the set* $\overline{\mathcal{S}}$ *resulting from Algorithm 6.2 is* $\langle\Omega,\mathcal{U},\lambda\rangle$*-invariant.*

Proof: The termination condition in step 3 guarantees that, when the algorithm terminates, the condition $\mathcal{O}_i^{[n_x+n_u]} \subseteq \overline{\text{Pre}}_{\langle\Omega,\mathcal{U},\lambda\rangle}(\mathcal{O}_i^{[n_x]})$ is satisfied. Also, by construction (step 3c) it is guaranteed that $\mathcal{O}_i^{[n_x]} \subseteq \text{proj}_{n_u}(\mathcal{O}_i^{[n_x+n_u]})$. Due to equivalence (6.4) it is then also guaranteed that $\mathcal{O}_i^{[n_x]} \subseteq \text{Pre}_{\langle\Omega,\mathcal{U},\lambda\rangle}(\mathcal{O}_i^{[n_x]})$. After substituting $\overline{\mathcal{S}} \triangleq \mathcal{O}_i^{[n_x]}$, Lemma 6.1 then shows that $\overline{\mathcal{S}}$ is $\langle\Omega,\mathcal{U},\lambda\rangle$-invariant. □

The following sections will now show how the degrees of freedom present in Algorithm 6.2 can be exploited in order to save computational time and obtain reduced-complexity control invariant sets. It goes without saying that also for the construction of control invariant sets one should regularly remove redundant constraints (i.e., *garbage collection*) in the sets $\mathcal{O}_i^{[n_x+j]}$ in order to reduce the computational overhead caused by these constraints. The algorithms described in the following sections do not explicitly mention the removal of redundant constraint, but the same rule of thumb described in Chapter 2 can still be used.

6.2.2 Pruning

An important similarity between Algorithms 6.2 and 2.3 is that the invariant sets are constructed iteratively and that the constraints of the resulting sets can be structured

into different layers of constraints, with each layer corresponding to a different iteration.

However, due to the additional projection step in Algorithm 6.2, there are no more unambiguous parent-child relationships between the constraints of consecutive layers, i.e. consecutive sets $\mathcal{X}_i^{[n_x]}$ and $\mathcal{X}_{i+1}^{[n_x]}$, which seems to rule out the possibility of applying pruning to eliminate the number of constraints with identical parent constraints. However, there *is* a clear relationship between the constraints of $\mathcal{X}_i^{[n_x]}$ and $\mathcal{X}_{i+1}^{[n_x+n_u]}$, since the latter are constructed using the $\overline{Pre}_{\langle\Omega,\mathcal{U},\lambda\rangle}(\cdot)$-operator. Therefore, the following algorithm proposes to use *pruning* in step 2b of Algorithm 6.2.

Algorithm 6.3 (**MCAS construction using pruning).** *Given a system* (3.1)-(3.2) *subject to constraints* (2.10)-(2.11) *and* $\lambda \in (0,1], \lambda' \in (0,1], \lambda' \leq \lambda, \gamma \geq 0$*, apply Algorithm 6.2 with the following implementations for steps 2, 3b and 3c:*

- **Step 2:** *Set* $\mathcal{X}_0^{[n_x+n_u-j]} := \mathrm{proj}\big(\mathcal{X}_0^{[n_x+n_u-j+1]}\big)$ *for* $j = 1, \ldots, n_u$.
- **Step 3b:** *Set* $\mathcal{X}_i^{[n_x+n_u]} := \overline{Pre}_{\langle\Omega,\mathcal{U},\lambda'\rangle}(\mathcal{X}_{i-1}^{[n_x]})$. *Apply* pruning *to constraints of* $\mathcal{X}_i^{[n_x+n_u]}$ *that originate from the same constraint in* $\mathcal{X}_{i-1}^{[n_x]}$ *in the same way as explained in Algorithm 5.2. No convergence constraint is imposed in this algorithm: constraints are always tightened if* $\eta_1 < 1 + \gamma$.
- **Step 3c:** $\mathcal{X}_i^{[n_x+n_u-j]} := \mathrm{proj}\big(\mathcal{O}_i^{[n_x+n_u-j+1]}\big)$ *for* $j = 1, \ldots, n_u$.

It is clear that the above implementations for these steps satisfy the conditions put forward in Algorithm 6.2 and hence the resulting set is guaranteed to be $\langle\Omega,\mathcal{U},\lambda\rangle$-invariant.

In step 3c) only those constraints have to be retained that are not yet present in the set $\mathcal{X}_{i-1}^{[n_x+n_u-j]}$ in order to exactly satisfy the condition present in step 3c) of Algorithm 6.2. This insight allows a significant amount of redundant computations to be eliminated. As a result practical implementations of this algorithm only need to store the sets $\mathcal{O}_i^{[n_x+n_u-j]}, j = 0, \ldots, n_u$.

Similar to Algorithm 5.2, larger values for γ lead to lower-complexity but also lower-volume sets. Also in this case, this trade-off improves as the amount of uncertainty decreases. The reasons for the absence of explicit convergence measures (prohibiting excessive amounts of constraint tightening) in this algorithm is the absence of quantitative results regarding the convergence of Algorithm 6.1 as is the case for Algorithm 2.4. However, if divergence or very slow convergence, decreasing the value of γ typically helps, so from a practical point of view, no fundamental problems should be expected.

6.2.3 Trimming

Similar to the previous section, where pruning is extended towards control invariant sets, this section extends the trimming method proposed in Section 5.2.3 towards this more general setting. More specifically, the applicability of Theorem 5.4 is extended in a natural way from autonomous systems to systems with inputs.

Theorem 6.2 (Control invariant sets for modified system matrices). *Given a set $\mathcal{S} \in \mathbb{R}^{n_x}$ and two autonomous LPV systems of the form* (3.1)-(3.2) *subject to constraints* (2.10)-(2.11)*, with uncertainty polytopes Ω_1 and Ω_2 that are defined as*

$$\Omega_1 \triangleq \mathrm{Co}\{[A_1\ B_1], \ldots, [A_r\ B_r]\}, \tag{6.5a}$$

$$\Omega_2 \triangleq \mathrm{Co}\{[A_1'\ B_1'], \ldots, [A_r'\ B_r']\}, \tag{6.5b}$$

with

$$[A_i'\ B_i'] = (1+c)[A_i\ B_i] - c[I\ 0], \qquad i = 1, \ldots, r, \tag{6.6}$$

where $c \in \mathbb{R}^+$. If $\mathcal{S}$ is convex and $\langle \Omega_2, \mathcal{U} \rangle$-invariant, then it is also $\langle \Omega_1, \mathcal{U} \rangle$-invariant.

Proof: If $\mathcal{S}$ is $\langle \Omega_2, \mathcal{U} \rangle$-invariant then

$$\exists u(x) : \Big(A_i'x + B_i'u(x) \Big) \in \mathcal{S}, \ \ \forall x \in \mathcal{S}, i = 1, \ldots, r,$$

or equivalently

$$\Big((1+c)A_ix - cx + (1+c)B_iu(x) \Big) \in \mathcal{S}, \ \ \forall x \in \mathcal{S}, i = 1, \ldots, r.$$

Due to convexity of $\mathcal{S}$ and the fact that $c > 0$ it is therefore also guaranteed that

$$\left(\frac{1}{1+c} \Big((1+c)A_ix - cx + (1+c)B_iu(x) \Big) + \frac{c}{1+c}x \right) \in \mathcal{S}, \ \ \forall x \in \mathcal{S}, i = 1, \ldots, r,$$

which, after some algebraic manipulation, is equivalent to

$$\Big(A_ix + B_iu(x) \Big) \in \mathcal{S}, \ \ \forall x \in \mathcal{S}, i = 1, \ldots, r,$$

which proves that $\mathcal{S}$ is $\langle \Omega_1, \mathcal{U} \rangle$-invariant. □

This theorem shows that also for the construction of control invariant sets, the system matrices can be modified in a certain way without losing the property that the resulting control invariant set is a control invariant set for the original system. By adding a controller $u(k) = -Kx(k), k \in \mathbb{N}$ to the above systems and constructing the corresponding autonomous LPV system, one can see that Theorem 5.4 is again obtained.

A disadvantage in this setting is the difficulty of tuning the parameter c. While positive invariant sets can only be constructed for asymptotically stable systems and the constraint tree depth can be related to the JSR of the autonomous system, some heuristics can be formulated for the tuning of c. However, control invariant sets can also be constructed for open-loop unstable systems and no quantitative convergence results similar to the ones for positive invariant sets are available. As a result, it is not yet well understood how c can be fine-tuned, apart from trial and error.

6.2.4 Reduced-complexity set projections

In Sections 6.2.2 and 6.2.3 the two main methods described in Chapter 5 are extended towards the construction of control invariant sets. Both methods influence the $\overline{\text{Pre}}_{\langle\Omega,\mathcal{U},\lambda'\rangle}(\cdot)$-operator and hence the way in which the $(n_x + n_u)$-dimensional sets $\mathcal{X}_{i+1}^{[n_x+n_u]}$ are constructed based on the n_x-dimensional sets $\mathcal{X}_i^{[n_x]}$ (see Figure 6.1).

However, as pointed out in Section 6.1, also the projection step contributes significantly to both the complexity of the resulting control invariant sets and the computation times of the algorithm. Therefore, this section focusses on more efficient algorithms for the calculation of such projections. Appendix B explains how such projections can be computed by means of Fourier-Motzkin elimination, which is the standard algorithm for such operations [146]. As explained its computational complexity typically increases exponentially as a function of the number of dimensions across which the projection takes place (i.e. the number of inputs n_u in Algorithms 6.1 and 6.2), unless some specific problem structure is present.

An alternative method is the so called Equality Set Projection algorithm (ESP, [62]), which has a computational complexity that is linear as a function of the number of constraints describing the end result. The main advantage of this approach is its independence of the complexity of projections in intermediate dimensions, since multi-dimensional projections can be computed directly instead of dimension per dimension. However, in the setting of control invariant sets, also the complexity of the projected set can be impractically large and hence also this method would have an impractically large computational complexity.

To conclude we can state that neither Fourier-Motzkin elimination, nor ESP are expected to exhibit favorable scaling behavior in the context of the computation of control invariant sets. Therefore, this section focusses on computing approximations of the exact projections, in order to obtain computational complexity reductions as well as reductions in the number of constraints describing the resulting projections. In order to obtain such approximations we modify the Fourier-Motzkin elimination algorithm (Algorithm B.1) by using ideas from previous sections and chapters. We will distinguish between two different variants:

- **Outer approximation:** the exact projection is guaranteed to lie inside the computed set.

- **Inner approximation:** the computed set is guaranteed to lie inside the exact projection.

The following sections discuss these two variants individually, after which it is explained how these new algorithms can be used in the framework of Algorithm 6.2 in order to obtain further complexity reductions. Both algorithms incorporate a parameter $\gamma \geq 0$ that allows a trade-off to be made between complexity and accuracy of the resulting projection. We only focus on one-dimensional projections; repeated application of the algorithms will yield reduced-complexity multi-dimensional projections.

6.2.4.1 Outer approximations

The following algorithm is a straightforward modification of Fourier-Motzkin elimination (Algorithm B.1) that only retains constraints if their significance exceeds the threshold $1+\gamma$.

Algorithm 6.4 (Outer approximation to $\mathrm{proj}(\mathcal{P})$). *Given a polytope $\mathcal{P} \triangleq \{x | A_{\mathcal{P}} x \leq b_{\mathcal{P}}\}$ and a scalar $\gamma \geq 0$, construct an outer approximation $\mathcal{P}'$ of its projection $\mathrm{proj}(\mathcal{P})$ by means of Algorithm B.1, with the following modifications:*

- *Only add a constraint $a^{\mathrm{T}} x \leq b$ to $\mathcal{P}'$ if it is $(1+\gamma)$-significant, i.e. if $\mathrm{sig}_{\mathcal{P}'}(a^{\mathrm{T}}) \geq 1+\gamma$.*
- *After termination of the algorithm remove all constraints that are not $(1+\gamma)$-significant.*

We will denote the result of this algorithm as $\overline{\mathrm{proj}}_{\gamma}(\mathcal{P}) \triangleq \mathcal{P}'$. It is now straightforward to prove that the following theorem:

Theorem 6.3. *Given a polytope $\mathcal{P} \triangleq \{x | A_{\mathcal{P}} x \leq b_{\mathcal{P}}\}$ and a scalar $\gamma \geq 0$, then the following property holds:*

$$\mathrm{proj}(\mathcal{P}) \quad \subseteq \quad \overline{\mathrm{proj}}_{\gamma}(\mathcal{P}) \quad \subseteq \quad (1+\gamma)\mathrm{proj}(\mathcal{P}). \tag{6.7}$$

Proof: The proof is straightforward and follows from the fact that omitting a constraint $a^{\mathrm{T}} x \leq b$ from $\mathcal{P}'$ that is c-significant, with $c > 1$, is identical to replacing this constraint with $a^{\mathrm{T}} x \leq cb$. Since Algorithm 6.4 only omits constraints that are c-significant with $c \in (1, 1+\gamma)$, this directly proves the lemma. □

When projecting an n_1-dimensional polytope $\mathcal{P}$ to an n_2-dimensional polytope by applying Algorithm 6.4 multiple times, the above inclusion becomes

$$\mathrm{proj}(\mathcal{P}) \quad \subseteq \quad \overline{\mathrm{proj}}_{\gamma}(\mathcal{P}) \quad \subseteq \quad (1+\gamma)^{n_1-n_2}\mathrm{proj}(\mathcal{P}).$$

The volumes of the intermediate projections at dimensions $n \in \{n_2, \ldots, n_1\}$ are easily shown to satisfy

$$\mathrm{vol}\big(\mathrm{proj}(\mathcal{P})\big) \quad \leq \quad \mathrm{vol}\big(\overline{\mathrm{proj}}_{\gamma}(\mathcal{P})\big) \quad \leq \quad (1+\gamma)^{n(n_1-n)}\mathrm{vol}\big(\mathrm{proj}(\mathcal{P})\big).$$

This property shows, that if $n_2 < \lfloor \frac{n_1}{2} \rfloor$, the maximal increase in volume with respect to the exact projection is not obtained for $n = n_2$, but for an intermediate dimension $n > n_2$.

6.2.4.2 Inner approximations

The following algorithm is a modification of Fourier-Motzkin elimination (Algorithm B.1) that uses pruning in order to obtain reduced-complexity inner approximations of the real projections.

Algorithm 6.5 (Inner approximation to $\mathrm{proj}(\mathcal{P})$). *Given a polytope $\mathcal{P} \triangleq \{x | A_{\mathcal{P}} x \leq b_{\mathcal{P}}\}$ and a scalar $\gamma \geq 0$, construct an inner approximation $\mathcal{P}'$ of its projection $\mathrm{proj}(\mathcal{P})$ by means of Algorithm B.1, with the following modifications:*

- *Only add a constraint $a^{\mathrm{T}}x \leq b$ to $\mathcal{P}'$ if it is significant, i.e. $\mathrm{sig}_{\mathcal{P}'}(a^{\mathrm{T}}) > 1$. For every additional constraint perform the following steps:*
  1. *Check which constraints of optimization problem* (2.13) *were active when computing $\mathrm{sig}_{\mathcal{P}'}(a^{\mathrm{T}})$ and denote these indices as $i_1, \ldots, i_n$ (without loss of generality we only consider the non-degenerate case where exactly n constraints are active). This gives an idea of the constraints of $\mathcal{P}'$ that lie in the 'proximity' of $a^{\mathrm{T}}x \leq b$.*
  2. *Check whether tightening the constraint $a^{\mathrm{T}}x \leq b$ would render any of the constraints with indices $i_1, \ldots, i_n$ redundant, using Lemma 5.2. Denote the respective tightening factors as $\eta_1, \ldots, \eta_n$, with $\eta_j \equiv 1$ if the constraint with index i_j cannot be rendered redundant.*
  3. *Replace the constraint $a^{\mathrm{T}}x \leq b$ with $\eta a^{\mathrm{T}}x \leq b$, with $\eta \triangleq \max_{j:\eta_j \leq (1+\gamma)} \eta_j$.*
- *After termination of the algorithm remove all redundant constraints (i.e., those where $\mathrm{sig}_{\mathcal{P}'}(\cdot) \leq 1$).*

We will denote the result of this algorithm as $\underline{\mathrm{proj}}_{\gamma}(\mathcal{P}) \triangleq \mathcal{P}'$. It should, however, be noted that Algorithm 6.5 does not define an unambiguous relationship between $\mathcal{P}$ and $\underline{\mathrm{proj}}_{\gamma}(\mathcal{P})$, since the end result $\mathcal{P}'$ is not independent of the order in which constraints are added. This in turn depends on how the indices i_1 and i_2 are selected in Algorithm B.1. For any of these choices, the following theorem holds:

Theorem 6.4. *Given a polytope $\mathcal{P} \triangleq \{x | A_{\mathcal{P}}x \leq b_{\mathcal{P}}\}$ and the polytope $\mathcal{P}'$ constructed using Algorithm 6.5 for a given value of $\gamma \geq 0$, then the following property holds:*

$$\frac{1}{1+\gamma}\mathrm{proj}(\mathcal{P}) \quad \subseteq \quad \underline{\mathrm{proj}}_{\gamma}(\mathcal{P}) \quad \subseteq \quad \mathrm{proj}(\mathcal{P}). \tag{6.8}$$

Proof: The proof is straightforward and along the same lines of the proof of Theorem 6.3. Due to the way η is calculated in step 3, it is guaranteed that all constraints are tightened with a factor $\eta \in [1, 1+\gamma]$, which directly proves the Theorem. □

When projecting an n_1-dimensional polytope $\mathcal{P}$ to an n_2-dimensional polytope by applying Algorithm 6.4 multiple times, similar bounds for the volumes of the intermediate projections can be obtained as in Section 6.2.4.1:

$$\left(\frac{1}{1+\gamma}\right)^{n(n_1-n)} \mathrm{vol}\big(\mathrm{proj}(\mathcal{P})\big) \quad \subseteq \quad \mathrm{vol}\big(\underline{\mathrm{proj}}_{\gamma}(\mathcal{P})\big) \quad \subseteq \quad \mathrm{vol}\big(\mathrm{proj}(\mathcal{P})\big).$$

Again, if $n_2 < \lfloor \frac{n_1}{2} \rfloor$, the maximal decrease in volume with respect to the exact projection is not obtained for $n = n_2$, but for an intermediate dimension $n > n_2$.

Finally, one can see that the computational complexity of Algorithm 6.5 is significantly higher than the complexity of Algorithm 6.4, due to the fact that for every additional constraint Lemma 5.2 has to be applied multiple times. An alternative method for computing inner approximations is calculating $\overline{\mathrm{proj}}_{\gamma}\left(\frac{1}{1+\gamma}\mathcal{P}\right)$. However, it is expected that this method, while computationally more attractive, will yield projections with less favorable volume/complexity trade-offs.

6.2.4.3 MCAS construction using reduced-complexity set projections

This section discusses how the algorithms discussed in Sections 6.2.4.2 and 6.2.4.2 can be incorporated in the framework of Algorithm 6.2. Because steps 2) and 3c) of Algorithm 6.2 only allow subsets of projections to be used it is clear to see that Algorithm 6.4 is not suited for use in the construction of control invariant sets. Therefore the following algorithm describes a modification of Algorithm 6.3 that uses Algorithm 6.5 for computing (approximate) set projections.

Algorithm 6.6 (Reduced complexity MCAS construction). *Given a system* (3.1)-(3.2) *subject to constraints* (2.10)-(2.11) *and* $\lambda \in (0,1], \lambda' \in (0,1], \lambda' \leq \lambda, \gamma_{\mathrm{pre}} \geq 0, \gamma_{\mathrm{proj}} \geq 0$*, apply Algorithm 6.3 with* $\gamma = \gamma_{\mathrm{Pre}}$ *and the following modification:*

- *Apply Algorithm 6.5 with parameter* $\gamma = \gamma_{\mathrm{proj}}$ *for computing set projections in steps 2) and 3c).*

Similar to Algorithm 6.3 this algorithm clearly fits within the framework laid out by Algorithm 6.2 and hence the resulting set will be $\langle \Omega, \mathcal{U}, \lambda \rangle$-invariant. The same efficiency gains can be obtained as in Algorithm 6.3 when implementing the projections such that redundant computations are eliminated.

It should be noted that if $\gamma_{\mathrm{proj}} = 0$ the above algorithm is identical to Algorithm 6.3. If $\gamma_{\mathrm{proj}} = 0$ and $\gamma_{\mathrm{pre}} = 0$ the algorithm is identical to Algorithm 6.1.

It should also be noted that, while Algorithm 6.3 can only result in reduced-complexity sets if $r > 0$, Algorithm 6.6 can also result in complexity reductions if $r = 1$ (i.e., the LTI case).

6.3 Examples

In this section we provide several numerical examples to illustrate the different techniques described in this chapter.

6.3.1 Triple integrator

Due to the low order of the double integrator example used in previous sections and the fact that it only has a single input, the MCAS of that example already has a low complexity, even when computed exactly by using Algorithm 6.1. Therefore we start this examples section with a slightly more challenging triple integrator example, defined as follows:

$$A_1 = \begin{bmatrix} 1 & 0.1 & 0 \\ 0 & 1 & 0.1 \\ 0 & 0 & 1 \end{bmatrix}, \qquad B_1 = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \tag{6.9a}$$

$$A_2 = \begin{bmatrix} 1 & 0.1 & 0 \\ 0 & 1 & 0.2 \\ 0 & 0 & 1 \end{bmatrix}, \qquad B_2 = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{6.9b}$$

The system is subject to constraints (2.10), (2.11) defined as

$$A_x = [0.1I;\ -0.1I], \tag{6.10a}$$

	Alg. 6.5		Alg. B.1	Alg. 6.4	
	$\gamma = 0.1$	$\gamma = 0.05$		$\gamma = 0.05$	$\gamma = 0.1$
$\mathbf{n = 8}$	16	16	16	16	16
$\mathbf{n = 7}$	31	34	56	36	30
$\mathbf{n = 6}$	62	80	112	107	93
$\mathbf{n = 5}$	103	137	140	268	189
$\mathbf{n = 4}$	101	152	112	233	142
$\mathbf{n = 3}$	43	53	56	81	46
$\mathbf{n = 2}$	13	17	16	21	14

Table 6.1: Number of constraints describing (approximate) projections of a randomly generated 8-dimensional polytope as described in Section 6.3.2. See also Table 6.2.

$$A_u = [10\ 0;\ -1\ 0;\ 0\ 1;\ 0\ -2]. \tag{6.10b}$$

Figure 6.2 shows the relative number of constraints and relative computation times of control invariant sets computed for system (6.9)-(6.10) using Algorithm 7.2 as a function of γ_{Pre} and γ_{proj}. Parameters $\gamma_{\mathrm{Pre}} = 0, \gamma_{\mathrm{proj}} = 0$ resulted in 352 constraints after a computation time of 1503 seconds. All constraint counts and computation times reported in Figure 6.2 are normalized with respect to these two values.

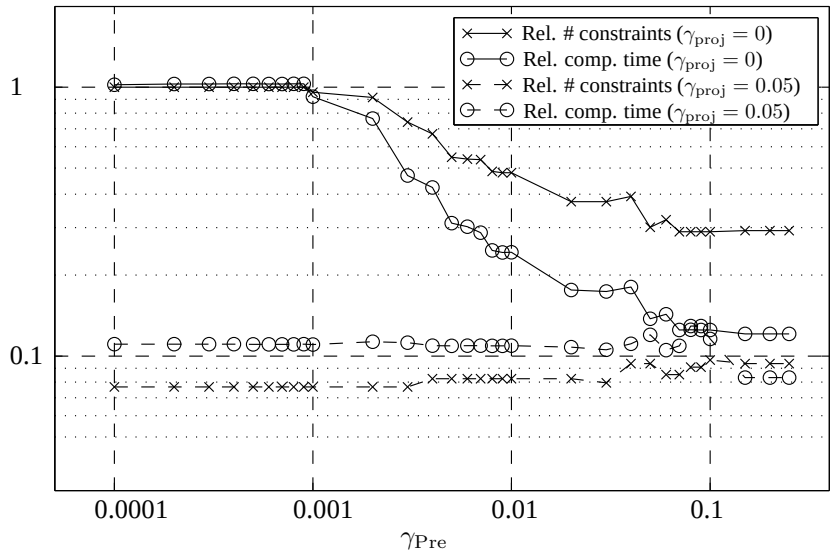
It can be observed that both pruning (governed by γ_{Pre}) and reduced-complexity projections (governed by γ_{proj}) can reduce the number of constraints of the resulting control invariant sets. In this example both methods also lead to computation time reductions of up to more than an order of magnitude. When both methods are combined further complexity reduction can be obtained. However, when both γ_{proj} and γ_{Pre} are given relatively large values, an increase in the number of constraints (compared to other parameter settings) is observed in this example. Using reduced-complexity projections without pruning seems to yield the most predictable and favorable results.

Figure 6.3 shows the resulting control invariant sets for $\gamma_{\mathrm{Pre}} \in \{0, 0.01\}$ and $\gamma_{\mathrm{proj}} \in \{0, 0.01\}$.

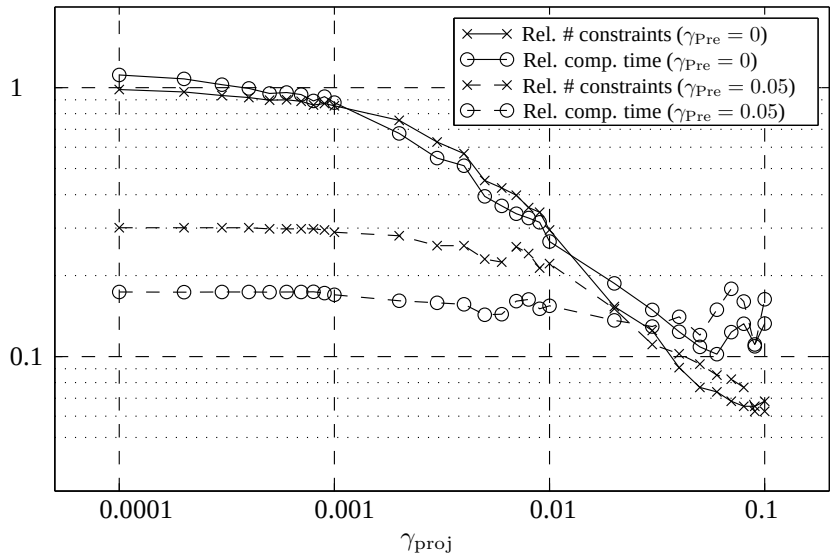
6.3.2 Reduced-complexity projections

We now investigate whether Algorithms 6.4 and 6.5 can prove useful to compute approximate projections in more general settings. Therefore we randomly generate a 8-dimensional polytope $Ax \leq \mathbf{1}$ with $A = [R; -R]$, where $R \in \mathbb{R}^{8\times 8}$ is a matrix whose elements are drawn from a standard normal distribution. Figure 6.4 depicts the approximate and exact projections. Tables 6.1 and 6.2 report the number of constraints and the computation times of the projections.

It seems that Algorithms 6.5 and 6.4 are much less efficient in this more general setting. None of the parameter settings reported in Tables 6.1 and 6.2 lead to convincing results, which suggests that the practical use of Algorithm 6.5 is limited to computing projections in Algorithm 6.6.

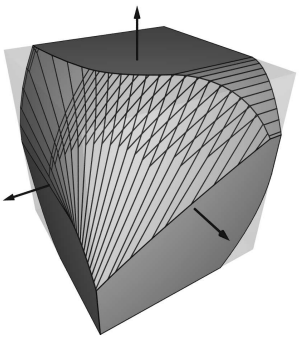


(a) $\gamma_{\mathrm{Pre}} \in [0.0001, 0.25], \gamma_{\mathrm{proj}} \in \{0, 0.05\}$

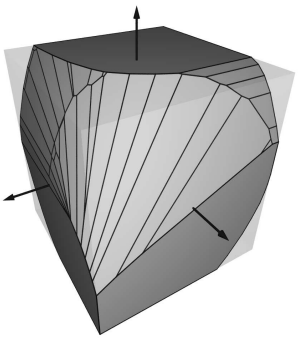


(b) $\gamma_{\mathrm{Pre}} \in \{0, 0.05\}, \gamma_{\mathrm{proj}} \in [0.0001, 0.1]$

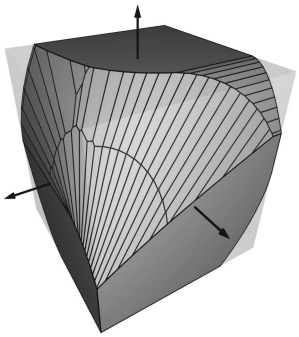
Figure 6.2: Relative number of constraints and relative computation times of control invariant sets for system (6.9)-(6.10) obtained with Algorithm 6.2. Parameters $\lambda = 1$ and $\lambda' = 0.999$ and different values for γ_{Pre} and γ_{proj} were used. All values are normalized with respect to the results obtained for $\gamma_{\mathrm{Pre}} = 0, \gamma_{\mathrm{proj}} = 0$ (352 constraints, 1503 seconds). Computations were performed on a 2.6GHz x86 CPU using MATLAB 7.1 and SeDuMi 1.1.



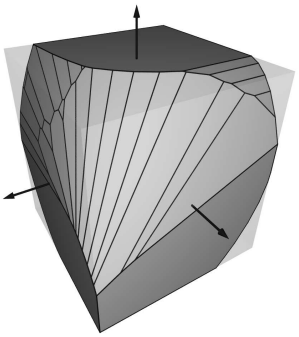
(a) $\gamma_{\mathrm{Pre}} = 0$, $\gamma_{\mathrm{proj}} = 0$, # constr.: 352, CPU-t.: 1503s.



(b) $\gamma_{\mathrm{Pre}} = 0$, $\gamma_{\mathrm{proj}} = 0.01$, # constr.: 104, CPU-t.: 395s.



(c) $\gamma_{\mathrm{Pre}} = 0.01$, $\gamma_{\mathrm{proj}} = 0$, # constr.: 169, CPU-t.: 358s.



(d) $\gamma_{\mathrm{Pre}} = 0.01$, $\gamma_{\mathrm{proj}} = 0.01$, # constr.: 90, CPU-t.: 346s.

Figure 6.3: Control invariant sets for system (6.9)-(6.10) computed using Algorithm 6.6 with $\lambda = 1, \lambda' = 0.999$ and different values of γ_{Pre} and γ_{proj}. The original constraint set $\mathcal{X}$ is depicted as a transparent box around each control invariant set. Computations were performed on a 2.6GHz x86 CPU using MATLAB 7.1 and SeDuMi 1.1. Computation times exclude visualization.

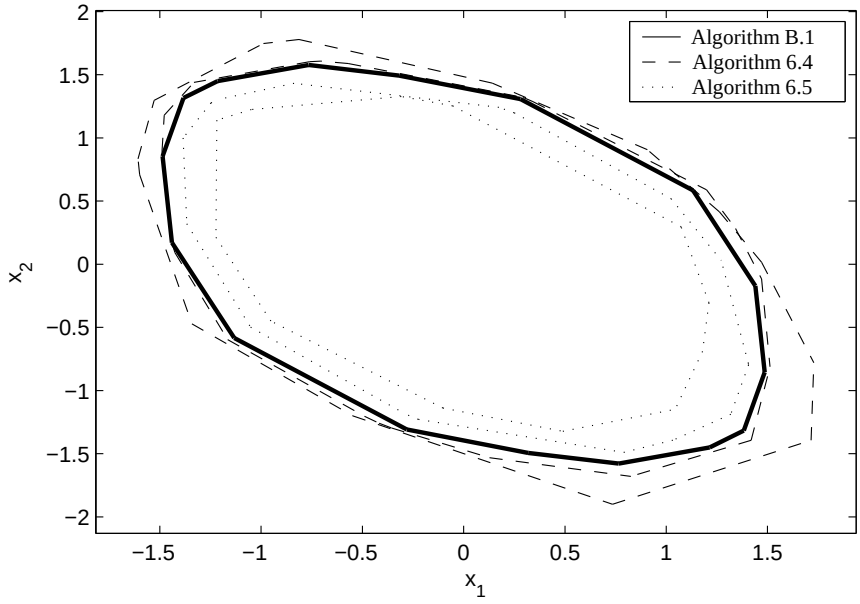


Figure 6.4: 2-dimensional (approximate) projections of a randomly generated 8-dimensional polytope, computed using Algorithm B.1 (solid), Algorithm 6.5 with $\gamma \in \{0.05, 0.1\}$ (dotted) and Algorithm 6.4 with $\gamma \in \{0.05, 0.1\}$ (dashed). See also Tables 6.1 and 6.2.

	Alg. 6.5		**Alg. B.1**	**Alg. 6.4**	
	$\gamma = 0.1$	$\gamma = 0.05$		$\gamma = 0.05$	$\gamma = 0.1$
$\mathbf{8 \to 7}$	29	29	18	12	10
$\mathbf{7 \to 6}$	91	106	157	50	37
$\mathbf{6 \to 5}$	254	451	529	365	218
$\mathbf{5 \to 4}$	331	640	536	1385	583
$\mathbf{4 \to 3}$	166	336	223	627	219
$\mathbf{3 \to 2}$	20	22	25	42	16
Total	**891**	**1584**	**1488**	**2480**	**1083**

Table 6.2: Computation times in seconds of (approximate) projections of a randomly generated 8-dimensional polytope as described in Section 6.3.2. Computations were performed on a 2.6GHz x86 CPU using MATLAB 7.1 and SeDuMi 1.1. See also Table 6.1.

6.4 Conclusions

In this chapter the two main methods discussed in Chapter 5 for computing reduced-complexity invariant sets (*pruning* and *trimming*) are extended towards the construction of control invariant sets. This is possible due to the fact that these sets can be constructed in a structurally similar way as invariant sets. However, constructing control invariant sets involves an additional step, namely the computation of projections of polytopes. In order to also reduce the complexity of this step in the algorithm, a modification to Fourier-Motzkin elimination is proposed that computes inner approximations of set projections.

Simulations on numerical examples indicate that these methods can reduce the complexity of polyhedral control invariant sets significantly, as well as the time to compute these sets. However, the modified Fourier-Motzkin elimination algorithm does not seem to lead to complexity reductions when applied in a more general context than that of constructing control invariant sets. This also suggests that the algorithms discussed in this chapter can be expected to have undesirable scaling behavior as a function of the input dimensionality of the system. Further research is needed to further investigate and improve this behavior.

Chapter 7

Robust MPC using Control Invariant Sets

"If everything seems under control,
you're just not going fast enough."

– Mario Andretti –

In this chapter the control invariant sets discussed in Chapter 6 are used to further enlarge the feasible regions of the robust MPC algorithms discussed in Chapter 4. The new algorithms are based on general interpolation between a control invariant set of the system to be controlled and feasible regions of the existing MPC algorithms. The new algorithms have a feasible region equal to the control invariant set and as such achieve the largest feasible region theoretically possible. Recursive feasibility and asymptotic stability are also maintained. Finally, it is also shown that this technique can be used to obtain recursive feasibility in settings, such as e.g. tracking problems, where the traditional algorithms are not guaranteed to be recursively feasible.

7.1 Introduction

In Chapter 4 several new robust MPC algorithms are introduced, with the main aim of obtaining an improved trade-off between the on-line computational complexity and the size of the feasible region, while retaining recursive feasibility, asymptotic stability and locally optimal control behavior. Chapter 5 further improves these results by constructing reduced-complexity invariant sets. However, obtaining large feasible regions still involves tuning several parameters, as can be seen in Figure 4.4 and 4.13,

and there is still no guarantee that sufficiently large feasible regions are obtained in the end.

As shown in [63], it is possible to use control invariant sets as terminal constraint in robust MPC instead of positive invariant sets, which can significantly enlarge the feasible region. However, the algorithms presented in Chapter 4 do not incorporate a terminal constraint separately from the within-horizon constraints, but rather construct the entire set of constraints by means of calculating an invariant set for an augmented system. This enables the use the algorithms presented in Chapter 5 in order to reduce the number of constraints, but makes it difficult to replace the terminal constraint with a control invariant set. Also, in interpolation-based algorithms, the notion of a terminal constraint is not explicitly present, which is also a complicating factor for incorporating control invariant sets in the formulation.

In this chapter, instead of replacing the terminal constraints, as suggested in [63], a more general method is proposed, that allows the extension of the feasible region of most recursively feasible robust MPC algorithms by means of incorporating control invariant sets. Locally optimal behavior is retained by applying the existing MPC algorithm if the current state lies inside its feasible region. If the state lies outside the feasible region, but inside a control invariant set, this set is used to drive the system state towards the feasible region of the MPC controller. This methodology can be captured in an interpolation-based theoretical framework for non-linear control laws, which also allows these results to be extended towards more general settings, like tracking problems.

This chapter is structured as follows. First, in Section 7.2 the interpolation based theoretical framework is laid out. Section 7.3 then shows how this framework can be used to combine control invariant sets with existing MPC algorithms. Section 7.4 then shows, as a proof-of-concept, how control invariant sets can be used to obtain recursive feasibility when applying the existing algorithms to tracking problems. Finally, Sections 7.5 and 7.6 give an example and conclusions.

7.2 General interpolation for non-linear control laws

In this section the concept of general interpolation is extended towards non-linear control laws. Although interpolation is based upon convex (hence linear) combinations of states and corresponding inputs, we show that only the dynamics of the system that is controlled need to be linear. Although general interpolation [3, 98, 122] until now has only been applied to linear systems controlled by linear control laws, these requirements are overly stringent. Linearity of the control law, and consequently linearity of the closed-loop system, allow the easy construction of a cost function for the on-line optimization problem, but in order to guarantee recursive feasibility linearity of the control law is not a necessary requirement. The only requirement, as is shown later in this section, is the existence of a convex invariant set for the closed-loop system of the (linear) open-loop system and the (non-linear) control law. This extension will allow us to formulate new MPC algorithms using control invariant sets in future sections. First some additional notation is introduced after which recursive feasibility is proven in this setting.

7.2.1 Problem formulation

As already mentioned we still consider LPV systems of the form (3.1)-(3.2) subject to constraints (2.10)-(2.11). The control laws to be used in the interpolation-based controller are defined as

$$u(k) = \kappa_i(x(k)), \qquad k \in \mathbb{N}, i = 1, \dots, n. \tag{7.1}$$

In order to guarantee constraint satisfaction we assume there exist n convex sets $\mathcal{S}_{1\dots n} \subset \mathbb{R}^{n_x}$ that are feasible and positive invariant with respect to the respective closed-loop systems:

$$\mathcal{S}_i \subseteq \mathcal{X}, \qquad i = 1, \dots, n, \tag{7.2a}$$
$$\kappa_i(x) \in \mathcal{U}, \qquad \forall x \in \mathcal{S}_i, \qquad i = 1, \dots, n, \tag{7.2b}$$
$$Ax + B\kappa_i(x) \in \mathcal{S}_i, \qquad \forall x \in \mathcal{S}_i, \forall [A\ B] \in \Omega, \qquad i = 1, \dots, n. \tag{7.2c}$$

It will become clear in Section 7.3 that this assumption is in no sense restrictive and that the above conditions can be easily satisfied based on the results obtained in the previous chapters.

The aim is to construct an interpolation-based control law, similar to the GIMPC algorithm discussed in Section 4.2, based on the control laws $\kappa_{1\dots n}(\cdot)$ and the corresponding invariant sets $\mathcal{S}_{1\dots n}$.

7.2.2 General interpolation

A general interpolation algorithm for non-linear control laws, in further sections referred to as NL-GIMPC, can now be formulated as follows:

Algorithm 7.1 (NL-GIMPC). *Consider an LPV system of the form* (3.1)-(3.2) *subject to constraints* (2.10)-(2.11)*, control laws* (7.1) *and corresponding invariant sets* $\mathcal{S}_{1\dots n} \subset \mathbb{R}^{n_x}$ *satisfying conditions* (7.2)*. At every time* k*, given the current state* $x(k)$ *of the system, calculate a state decomposition as follows*

$$\min_{\hat{x}_{1\dots n} \in \mathbb{R}^{n_x}, \lambda_{1\dots n} \in \mathbb{R}} \quad f(\hat{x}_{1\dots n}, \lambda_{1\dots n}), \tag{7.3a}$$
$$\text{s.t.} \quad \sum_{i=1}^{n} \hat{x}_i = x(k), \tag{7.3b}$$
$$\hat{x}_i \in \lambda_i \mathcal{S}_i \qquad i = 1, \dots, n, \tag{7.3c}$$
$$\lambda_i \geq 0, \qquad i = 1, \dots, n, \tag{7.3d}$$
$$\sum_{i=1}^{n} \lambda_i = 1, \tag{7.3e}$$

with $f(\cdot, \cdot) : \mathbb{R}^{n \cdot (n_x+1)} \to \mathbb{R}$ *an arbitrary cost function, and apply the following input to the system:*

$$u(k) = \sum_{i=1}^{n} \lambda_i \kappa_i \left(\frac{\hat{x}_i}{\lambda_i} \right). \tag{7.4}$$

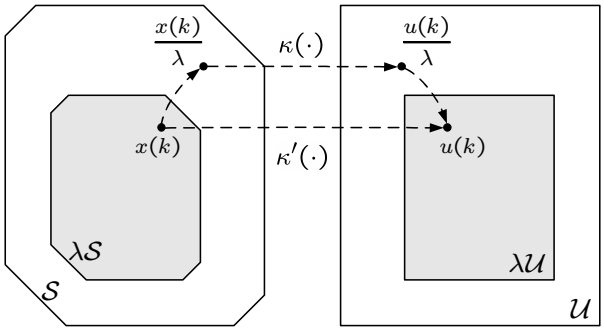


Figure 7.1: Illustration of the scaling that is used for obtaining scaled invariant sets in order to construct a GIMPC algorithm for non-linear control laws.

The state decomposition that is computed is identical to the one constructed using GIMPC. If the functions $\kappa_{1\ldots n}(\cdot)$ are linear, it is easy to verify that also the input $u(k)$ that is computed based on this state decomposition (see expression (7.4)), is identical to GIMPC. Since at this point we only want to prove recursive feasibility of NL-GIMPC (and not asymptotic stability), it is not important to already specify the function $f(\cdot,\cdot)$. This is done in Section 7.3 for the specific setting considered there. Before being able to prove recursive feasibility of Algorithm 7.1, we need to define the concept of *scaled control laws*:

Definition 7.1 (Scaled control law). *Given a control law $u(k) = \kappa(x(k)), k \in \mathbb{N}$ and a scalar $\lambda \in \mathbb{R}_0^+$, then a scaled control law $\kappa'(\cdot)$ is defined as*

$$\kappa'(x) \triangleq \lambda\kappa\left(\frac{x}{\lambda}\right). \tag{7.5}$$

Note that if $\kappa(\cdot)$ is a linear function, we have that $\kappa'(x) \equiv \kappa(x)$. However, in this section we are more interested in the non-linear case. The following lemma is instrumental in proving recursive feasibility of Algorithm 7.1 :

Lemma 7.1 (Invariant sets for scaled non-linear control laws). *Consider an LPV system of the form* (3.1)-(3.2) *subject to constraints* (2.10)-(2.11)*, a non-linear control law $u(k) = \kappa(x(k)), k \in \mathbb{N}$, a set $\mathcal{S} \subset \mathbb{R}^{n_x}$ satisfying* (7.2) *(omitting the subscripts i) and a strictly positive scalar $\lambda \in \mathbb{R}_0^+$. The set $\lambda\mathcal{S}$ then is a feasible, positive invariant set for the closed-loop system formed by* (3.1)-(3.2) *and the scaled non-linear controller defined in Definition 7.1 subject to state and input constraints $\lambda\mathcal{X}$ and $\lambda\mathcal{U}$.*

Proof: Due to (7.2a) we know that $\mathcal{S} \subseteq \mathcal{X}$ and hence $\lambda\mathcal{S} \subseteq \lambda\mathcal{X}$, which proves feasibility of $\lambda\mathcal{S}$ with respect to $\lambda\mathcal{X}$. Furthermore, $\forall x \in \lambda\mathcal{S}$ it is guaranteed that $\frac{x}{\lambda} \in \mathcal{S}$. Hence, due to (7.2b) it is guaranteed that $\kappa\left(\frac{x}{\lambda}\right) \in \mathcal{U}$. Therefore, it is also guaranteed that $\kappa'(x) = \lambda\kappa\left(\frac{x}{\lambda}\right) \in \lambda\mathcal{U}, \forall x \in \lambda\mathcal{S}$, which proves feasibility of $\lambda\mathcal{S}$ with respect to $\lambda\mathcal{U}$. Finally, due to (7.2c) we know that

$$Ax + B\kappa(x) \in \mathcal{S}, \qquad \forall x \in \mathcal{S}, \forall [A\ B] \in \Omega,$$

which, after multiplication of both sides of the inclusion with λ and substitution $x = \frac{x'}{\lambda}$, leads to

$$\lambda \left(A\frac{x'}{\lambda} + B\kappa \left(\frac{x'}{\lambda} \right) \right) \in \lambda\mathcal{S}, \qquad \forall \frac{x'}{\lambda} \in \mathcal{S}, \forall [A\ B] \in \Omega.$$

This is equivalent with

$$Ax' + B\lambda\kappa \left(\frac{x'}{\lambda} \right) \in \lambda\mathcal{S}, \qquad \forall x' \in \lambda\mathcal{S}, \forall [A\ B] \in \Omega,$$

which shows that $\lambda\mathcal{S}$ is positive invariant with respect to the closed-loop system formed by (3.1)-(3.2) and the controller $u(k) = \kappa'(x(k)), k \in \mathbb{N}$, which in turn completes the proof. □

This lemma is illustrated by means of Figure 7.1 and can now be used to prove recursive feasibility of Algorithm 7.1.

Theorem 7.1 (Recursive feasibility of NL-GIMPC). *Applied to a system of the form* (3.1)-(3.2)*, Algorithm 7.1 guarantees satisfaction of constraints* (2.10)-(2.11) *and is feasible at time* $k+1$ *if it is feasible at time* k*.*

Proof: Similar to other GIMPC algorithms, Algorithm 7.1 is feasible iff $x(k) \in \mathrm{Co}\{\mathcal{S}_1, \ldots, \mathcal{S}_n\}$. Hence, we have to prove that in that case also $x(k+1) \in \mathrm{Co}\{\mathcal{S}_1, \ldots, \mathcal{S}_n\}$ for any $[A\ B] \in \Omega$. Given a decomposition $\hat{x}_{1\ldots n}, \lambda_{1\ldots n}$ satisfying the constraints of optimization problem (7.3), we first show that $u(k) \in \mathcal{U}$. Due to Lemma 7.1 we know that $\lambda_i\kappa_i \left(\frac{\hat{x}_i}{\lambda_i} \right) \in \lambda_i\mathcal{U}$ and hence that $u(k) \in (\lambda_1\mathcal{U} \oplus \ldots \oplus \lambda_n\mathcal{U}) = \mathcal{U}$, where the latter equality is satisfied because $\mathcal{U}$ is convex and $\sum_{i=1}^{n} \lambda_i = 1$. This proves that (2.11) is satisfied. In order to prove feasibility at time $k+1$, we compute the state at time $k+1$:

$$\begin{aligned} x(k+1) &= Ax(k) + Bx(k), \\ &= A\sum_{i=1}^{n} \hat{x}_i + B\sum_{i=1}^{n} \lambda_i\kappa_i \left(\frac{\hat{x}_i}{\lambda_i} \right), \\ &= \sum_{i=1}^{n} \left(A\hat{x}_i + B\lambda_i\kappa_i \left(\frac{\hat{x}_i}{\lambda_i} \right) \right). \end{aligned}$$

Due to Lemma 7.1 we know that $A\hat{x}_i + B\lambda_i\kappa_i \left(\frac{\hat{x}_i}{\lambda_i} \right) \in \lambda_i\mathcal{S}_i, i = 1, \ldots, n, \forall [A\ B] \in \Omega$ and hence that $x(k+1) \in \mathrm{Co}\{\mathcal{S}_1, \ldots, \mathcal{S}_n\}, \forall [A\ B] \in \Omega$, which proves recursive feasibility. Finally, since $\mathrm{Co}\{\mathcal{S}_1, \ldots, \mathcal{S}_n\} \subseteq \mathcal{X}$, satisfaction of (2.10) is guaranteed, which concludes the proof. □

This theorem shows that linearity of the control laws $\kappa_{1\ldots n}(\cdot)$ is not strictly necessary for proving recursive feasibility. The main practical difficulties arising from non-linearity of these control laws is the difficulty of constructing a cost function $f(\cdot,\cdot)$ such that asymptotic stability is guaranteed and the construction of the invariant sets

$\mathcal{S}_{1\ldots n}$. The next section shows that in the specific case of interpolation between a control invariant set and the feasible region of an MPC controller, both aspects are not an issue.

7.3 Robust MPC using control invariant sets

In this section we show how the insights of Section 7.2 can be used to enlarge the feasible region of the algorithms discussed in Chapter 4. First we show that control invariant sets and feasible regions can be seen as positive invariant sets with respect to a non-linear control law. Section 7.3.3 then shows how NL-GIMPC can be applied in order to obtain an MPC controller with a maximal feasible region.

7.3.1 Control-invariant set induced controller

Although control invariant sets are constructed for open-loop systems without a priori considering a feedback controller with which the loop is closed, this section shows that these sets actually induce a state feedback control law. When using this control law to form a closed-loop system, one can show that the control invariant set is positive invariant with respect to this closed-loop system. First we show how any convex set containing the origin induces a control law.

Definition 7.2 (**Set-induced control law**[1], [13])**.** *Consider an LPV system of the form* (3.1)-(3.2)*, subject to input constraints* (2.11) *and a convex set $\mathcal{S} \in \mathbb{R}^{n_x}$ with the origin in its interior. The set-induced state feedback control law $u = \kappa_{\mathcal{S}}(x)$ is now defined as the result of the following optimization problem:*

$$\kappa_{\mathcal{S}}(x) \triangleq \underset{u \in \mathcal{U}}{\operatorname{argmin}} \quad \beta_{\mathcal{S}}(x, u), \tag{7.6a}$$

with $\beta_{\mathcal{S}}(x, u)$ defined as

$$\beta_{\mathcal{S}}(x, u) \triangleq \min_{\lambda \in \mathbb{R}} \quad \lambda, \tag{7.6b}$$

$$\text{s.t.} \quad A_i x + B_i u \in \lambda \mathcal{S}, \qquad i = 1, \ldots, r. \tag{7.6c}$$

One can see that the control law $\kappa_{\mathcal{S}}(x)$ is constructed such that it makes sure that at every time instant k the system is driven as far as possible inside the set $\mathcal{S}$. It should also be noted that if $\mathcal{S}$ is a polyhedral set, the above optimization problem reduces to an LP. Since the inputs are computed by solving a constrained optimization problem, the resulting controller will in general be non-linear. If $\mathcal{S}$ is control invariant with respect to the given open-loop system, the following lemma applies.

Lemma 7.2 (**Positive invariance of control invariant sets).** *A control invariant set $\mathcal{S}$ is positive invariant with respect to the closed-loop system formed by* (3.1)-(3.2) *and the induced control law $u(k) = \kappa_{\mathcal{S}}(x(k)), k \in \mathbb{N}$.*

[1] This type of controller is also referred to as *min-max control* in [13, 39, 40].

Proof: Consider a state $x \in \mathcal{S}$ and the corresponding input $u^\star$ and contraction factor $\lambda^\star$ resulting from solving (7.6). Since $\mathcal{S}$ is control invariant with respect to (3.1)-(3.2), it is guaranteed that $\exists u^{\mathrm{f}} \in \mathcal{U} : Ax + Bu^{\mathrm{f}} \in \mathcal{S}, \forall [A\ B] \in \Omega$. This guarantees that $\lambda^\star \leq 1$, which then shows that $Ax + B\kappa_{\mathcal{S}}(x) \equiv Ax + Bu^\star \in \mathcal{S}, \forall [A\ B] \in \Omega$, which proves the lemma. □

We now have shown that, once a control invariant set $\mathcal{S}$ is constructed (which in the previous chapter has been shown to be possible), its induced control law $u = \kappa_{\mathcal{S}}(x)$ can be used as a control law in NL-GIMPC. An interesting point to make is that this is the opposite of what is done in Section 4.2, where one first constructs a set of control laws, after which the corresponding positive invariant sets are computed. In this section, the existence of a control invariant set is assumed, after which a controller is constructed such that the control invariant set is positive invariant with respect to the resulting closed-loop system.

In the next section a similar result is obtained for MPC controllers where an exact characterization of the feasible region is known.

7.3.2 Feasible region as positive invariant set

In order to be able to use MPC controllers, which in general are non-linear control laws, in the NL-GIMPC algorithm, it is necessary to construct a positive invariant set for the corresponding closed-loop system. The problem is that there are no algorithms that are able to construct such sets in a general setting. However, the following lemma shows that under certain conditions the feasible region of the MPC controller will be a positive invariant set.

Lemma 7.3 (Positive invariance of the feasible region). *Consider an LPV system of the form* (3.1)-(3.2) *subject to constraints* (2.10)-(2.11) *and consider a robust MPC controller designed for this system, that guarantees satisfaction of* (2.10)-(2.11) *and is recursively feasible. Assume that the MPC controller can be written as a state feedback law* $u(k) = \kappa_{\mathrm{MPC}}(x(k)), k \in \mathbb{N}$ *and that we have an exact characterization* $\mathcal{F} \subset \mathbb{R}^{n_x}$ *of the feasible region of the controller, i.e.* $x \in \mathcal{F} \Leftrightarrow x$ *leads to a feasible MPC optimization problem. Under these assumptions the set* $\mathcal{F}$ *is a feasible, positive invariant set for the closed-loop system formed by* (3.1)-(3.2) *and the MPC controller.*

Proof: Due to the fact that the MPC controller guarantees satisfaction of (2.10), it will per definition be infeasible for all $x(k) \notin \mathcal{X}$ and hence we have that $\mathcal{F} \subseteq \mathcal{X}$. Furthermore, since the MPC controller guarantees satisfaction of (2.11), we have that $\kappa_{\mathrm{MPC}}(x) \in \mathcal{U}, \forall x \in \mathcal{F}$. This shows that $\mathcal{F}$ is a feasible set.

Due to recursive feasibility we know that if the controller is feasible at time k, i.e. if $x(k) \in \mathcal{F}$, it is also feasible at time $k+1$. Since by assumption $\mathcal{F}$ is an exact characterization of the feasible region, it is therefore guaranteed that also $x(k+1) \in \mathcal{F}$. This proves that $\mathcal{S}$ is positive invariant with respect to the closed-loop system and completes the proof. □

The assumptions made in this lemma seem rather restrictive and hard to fulfill, but the opposite is true. The three main algorithms discussed in Chapter 4 (GIMPC, GIMPC2, RMPC) all satisfy these conditions.

First of all, these three algorithms can be written as a static state feedback law. This cannot be done explicitly (which isn't necessary), but because the involved optimization problems only depend upon the current state of the system and not upon the past states, one can see that there is a static functional dependence between $u(k)$ and $x(k)$ in all three algorithms.

Secondly, all three algorithms have are recursive feasible and asymptotically stable. Only the former is a necessary condition in Lemma 7.3, but the latter property will prove useful in the next section.

Thirdly, the feasible regions can be exactly characterized. In the case of GIMPC the feasible region is exactly equal to the convex hull of the invariant sets $\mathcal{S}_{1\ldots n}$. In the case of GIMPC2 and RMPC the feasible region is equal to the projection of $\mathcal{S}_{\text{aug}}$ onto the first n_x dimensions.

Finally, Lemma 7.3 is perfectly illustrated by Figure 4.17, where trajectories starting from the boundary of the respective feasible regions of $\mathcal{P}$-GIMPC, $\mathcal{P}$-GIMPC2 and $\mathcal{P}$-RMPC are shown. These trajectories show that the feasible regions indeed are positive invariant sets for the closed-loop systems.

7.3.3 Interpolation between control invariant sets and feasible regions

In this section it is shown how the algorithms discussed in Chapter 4 can be combined with the control invariant sets constructed in Chapter 6. To this end, Algorithm 7.1 (NL-GIMPC) is applied in a specific way such that adding the control invariant set results in only a small additional on-line computational complexity. We choose $n = 2$ and make the following choices for the different controllers between which the interpolation takes place :

- κ_1 is taken as a recursively feasible MPC controller, e.g. one of those discussed in Chapter 4. $\mathcal{S}_1$ can then be chosen as the feasible region of that MPC controller. If the MPC controller is based on polyhedral invariant sets, the feasible region is also polyhedral, which evidently has computational advantages. We will refer to this controller as the *local MPC controller*.

- $\mathcal{S}_2$ is chosen as a λ-contractive control invariant set (with $\lambda \leq 1$) for the system that is to be controlled. κ_2 should then be taken as the control law induced by $\mathcal{S}_2$ (cfr. Definition 7.2). We will refer to this controller as the *set-induced controller* or the *outer controller*.

We can now choose $f(\hat{x}_{1\ldots n}, \lambda_{1\ldots n}) = \lambda_2$. In this way, we make sure that if the current state lies within $\mathcal{S}_1$ the control behavior is fully determined by the local MPC controller. This can now be formalized into the following algorithm.

Algorithm 7.2 (MPC using control invariant sets). *Given an LPV system of the form* (3.1)-(3.2) *subject to constraints* (2.10)-(2.11)*. Perform the following steps:*
Off-line:

- *Design a robust MPC controller (either $\mathcal{P}$-GIMPC, $\mathcal{P}$-GIMPC2 or $\mathcal{P}$-RMPC) for the given system and constraints and denote this controller as the static state*

feedback law $\kappa_1(\cdot)$. Construct the corresponding feasible region $\mathcal{F} = \{x \in \mathbb{R}^{n_x} | A_{\mathcal{F}} x \leq 1\}$ (see Section 7.3.2) and set $\mathcal{S}_1 := \mathcal{F}$.

- *Construct a control invariant set (e.g. by means of Algorithm 6.6) and denote it as $\mathcal{S}_2 = \{x \in \mathbb{R}^{n_x} | A_{\mathcal{S}_2} x \leq 1\}$. Define $\kappa_2(\cdot)$ as the corresponding set-induced control law $\kappa_{\mathcal{S}_2}(\cdot)$ (see Definition 7.2).*

On-line:

At every time instant k, given the current state $x(k)$, perform the following steps:

- *Solve the NL-GIMPC optimization problem* (7.3) *for the current state $x(k)$ with κ_1, κ_2 and $\mathcal{S}_1, \mathcal{S}_2$ defined as above and $f(\cdot,\cdot) \triangleq \lambda_2$.*
- *Apply the input $u(k)$ defined by* (7.4) *to the system.*

One can see that this algorithm, compared to simply applying $\kappa_1(\cdot)$, enlarges the feasible region to $\mathrm{Co}\{\mathcal{S}_1, \mathcal{S}_2\}$, which can be significantly larger than $\mathcal{S}_1$, since $\mathcal{S}_2$ is typically significantly larger. This advantage comes at the additional cost of solving optimization problems (7.3) and (7.6), which in this case are LP's. The size of these LP's is typically also moderate compared to the optimization problem involved in evaluating $\kappa_1(\cdot)$. Furthermore, the evaluation of $\kappa_1(\cdot)$ and $\kappa_2(\cdot)$ can be parallelized and therefore only solving (7.3) should be considered as an additional computational cost.

Lemma 7.4. *Algorithm 7.2 is recursively feasible.*

Proof: Due to Lemma 7.2 and 7.3 we know that $\mathcal{S}_1$ and $\mathcal{S}_2$ are feasible positive invariant sets with respect to the respective closed-loop systems formed by (3.1)-(3.2) and controllers $u(k) = \kappa_1(x(k)), k \in \mathbb{N}$ and $u(k) = \kappa_2(x(k)), k \in \mathbb{N}$ subject to constraints (2.10)-(2.11). Therefore $\mathcal{S}_1, \mathcal{S}_2$ and $\kappa_1(\cdot), \kappa_2(\cdot)$ satisfy the conditions put forward in Theorem 7.1, which proves this lemma. □

On top of recursive feasibility, also asymptotic stability is obtained if $\mathcal{S}_2$ is λ-contractive with $\lambda < 1$.

Lemma 7.5. *If $\kappa_1(\cdot)$ is asymptotically stabilizing and $\lambda < 1$, then Algorithm 7.2 is also asymptotically stabilizing.*

Proof: Consider an optimal decomposition $\hat{x}_1^{\mathrm{o}}(0), \hat{x}_2^{\mathrm{o}}(0), \lambda_1^{\mathrm{o}}(0), \lambda_2^{\mathrm{o}}(0)$ obtained at time 0 by solving optimization problem (7.3). Due to Lemma 7.1 we can now construct a feasible decomposition at time 1 as follows:

$$\hat{x}_1^{\mathrm{f}}(1) = A(0)\hat{x}_1^{\mathrm{o}}(0) + B(0)\lambda_1^{\mathrm{o}}\kappa_1\left(\frac{\hat{x}_1^{\mathrm{o}}(0)}{\lambda_1^{\mathrm{o}}}\right), \qquad \lambda_1^{\mathrm{f}}(1) = \lambda_1^{\mathrm{o}}(0), \tag{7.7a}$$

$$\hat{x}_2^{\mathrm{f}}(1) = A(0)\hat{x}_2^{\mathrm{o}}(0) + B(0)\lambda_2^{\mathrm{o}}\kappa_2\left(\frac{\hat{x}_2^{\mathrm{o}}(0)}{\lambda_2^{\mathrm{o}}}\right), \qquad \lambda_2^{\mathrm{f}}(1) = \lambda_2^{\mathrm{o}}(0), \tag{7.7b}$$

Due to λ-contractivity of $\mathcal{S}_2$ an even stricter choice $\lambda_2^{\mathrm{f}}(1) = \lambda\lambda_2^{\mathrm{o}}(0), \lambda_1^{\mathrm{f}}(1) = 1 - \lambda_2^{\mathrm{f}}(1) > \lambda_1^{\mathrm{o}}(0)$ can be made. Due to the cost function that is chosen, it is therefore

guaranteed that $\lambda_2^0(1) \leq \lambda_2^{\mathrm{f}}(1) = \lambda\lambda_2^{\mathrm{o}}(0)$. By applying this argument recursively it can be seen that $\lambda_2^0(k) \leq \lambda^k\lambda_2^{\mathrm{o}}(0)$ and hence that $\|\hat{x}_2^{\mathrm{o}}(k)\| \to 0$ as $k \to \infty$. Due to the fact that $\kappa_1(\cdot)$ is asymptotically stabilizing by assumption, we also have that $\|\hat{x}_1^{\mathrm{o}}(k)\| \to 0$ as $k \to \infty$. As a result $\|x(k)\| \to 0$ as $k \to \infty$, which proves asymptotic stability of Algorithm 7.2 under the specified conditions. □

To conclude, we can state the Algorithm 7.2 is able to increase the feasible region of existing MPC algorithms while retaining the two most important properties of the MPC algorithms that is used as controller $\kappa_2(\cdot)$: recursive feasibility and asymptotic stability. The additional computational cost is small, with only 2 additional LP's of moderate size that have to be solved at every time instant k. Finally, due to the cost function that is chosen, it can be seen that if the state of the system is situated within the feasible region of the local MPC controller, $\lambda_1 = 1 - \lambda_2 = 1$ is obtained, which shows that local optimality is also conserved.

7.3.4 Interpretation

The new Algorithm 7.2, which is based on non-linear general interpolation and control invariant sets, has a natural interpretation in terms of optimality and constraint handling. Figure 7.2 depicts the different operational regions of Algorithm 7.2, within which different trade-offs are made between optimal control behavior and constraint handling. As a result, als the computational complexity will vary in each region, since some regions allow significant simplifications to be made with respect to optimization problem (7.3) and the corresponding expression for the control action (7.4). The following regions can be discerned:

1. **The outer constrained region: $\mathcal{S}_2 \setminus \mathcal{S}_1$.**
   - **Action:** Solve optimization problem (7.3) and apply input (7.4).
   - **Optimality:** The control objective is only partly taken into account, depending on λ_1. The main goal is to drive the state towards $\mathcal{S}_1$ as fast as possible.
   - **Constraint handling:** Constraints are typically active for a longer period, after which the state enters $\mathcal{S}_1$, where typically still some constraints remain active.
   - **Computational complexity:** 2 LPs ((7.3) and (7.6)) and the QP associated with $\kappa_1(\cdot)$ have to be solved.
2. **The inner constrained region: $\mathcal{S}_1 \setminus \mathcal{S}$.**
   - **Action:** Since in this region $\lambda_1 \equiv 1$, we can directly apply the local MPC controller $u(k) = \kappa_1(x)$.
   - **Optimality:** The control behavior is optimized subject to the imposed constraints.
   - **Constraint handling:** Constraints are typically only active for a few time steps, after which the state enters $\mathcal{S}$. In case $\mathcal{P}$-RMPC is used as $\kappa_1(\cdot)$, the

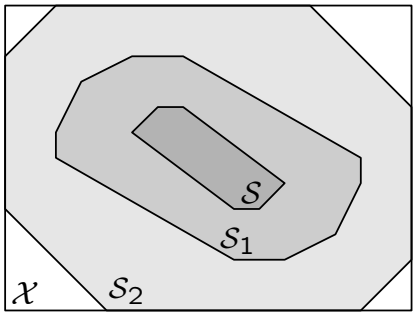


Figure 7.2: Schematic depiction in state space of the different regions of operation of Algorithm 7.2. $\mathcal{S}$ represents the MAS of the locally optimal linear controller used in the local MPC controller. $\mathcal{S}_1$ is the feasible region of the local MPC controller. $\mathcal{S}_2$ is the control invariant set used for interpolation. $\mathcal{X}$ is the imposed state constraint set. Darker shades of grey indicate more optimal behavior according to the control objective.

state is driven inside $\mathcal{S}$ in at most N time steps. Straightforward application of Algorithm 7.2 would yield $\lambda_1 = 1 - \lambda_2 = 1$, which allows us to skip this step and directly evaluate $\kappa_1(x(k))$.

- **Computational complexity:** Only the QP associated with $\kappa_1(\cdot)$ has to be solved.

3. **The unconstrained region: $\mathcal{S}$.**

   - **Action:** In this region no constraints are active and the local MPC controller $u(k) = \kappa_1(x(k))$ should result in identical behavior as the local controller $u(k) = -Kx(k)$ used in its design. Therefore this latter control action can directly be applied to the system.
   - **Optimality:** The control behavior is optimal if K is chosen to be the locally optimal controller.
   - **Constraint handling:** No constraints are active.
   - **Computational complexity:** No optimization has to be performed, only the locally optimal controller $u(k) = -Kx(k)$ has to be evaluated.

Finally, there exists one additional region, which however can be argued not to be part of the proper operation of the algorithm:

4. **The twilight zone: $\mathcal{X} \setminus \mathcal{S}_2$.**

   - **Action:** In this region it is not guaranteed that in all future time steps all constraints will be satisfied. Whether or not this happens can depend on the actual values of $A(k), B(k)$. In order to reduce the odds and/or severity of the possible constraint violations, one might still apply $u(k) = \kappa_2(x(k))$, which drives the state as close as possible to $\mathcal{S}_2$.

- **Optimality:** The control objective is not taken into account. The only concern is constraint handling.
- **Constraint handling:** Some constraints are active and the possibility exists that some state constraints might be violated in future time steps. By definition $\kappa_2(\cdot)$ always guarantees input constraint satisfaction.
- **Computational complexity:** Only LP (7.6) has to be solved.

The following general conclusions can be drawn:

- The further away one moves from the origin, the more constraints become active and during longer periods of time. This results in control behavior that increasingly deviates from the locally optimal behavior, which in turn necessitates different control strategies. Hence, the further one moves away from the origin, the more the emphasis has to be put on constraint handling and the less one can take the control objectives into account.
- The more constraints are active or the longer they are active, the more computational resources one has to employ in order to perform proper constraint handling without losing stability of the closed-loop system.
- The ability to enlarge the feasible region to the size of a control invariant set, incurs an additional cost in the form of two additional LP's that have to be solved. However, the ability to obtain this enlarged feasible region reduces the need for the local MPC controller to have a large feasible region, which in turn potentially reduces its computational complexity.

7.4 Control-invariant sets in tracking problems

While up to the previous section only stabilization problems were considered, we here consider the problem of steering a system such that its output or its states *track* a prescribed trajectory, i.e. a tracking problem. The classical stability framework introduced in Chapter 1 and further extended to the robust case in Chapter 3 is only valid for stabilization problems and hence it cannot be applied to stabilization problems in a theoretically sound manner.

The main aim of this section is to show that control invariant sets can be used in order to guarantee input and state constraint satisfaction in tracking problems. This section is conceived as a *proof-of-concept* and hence, for simplicity reasons, only *tracking without preview* is considered, meaning that no information on future values of the reference trajectory is available to the controller. The next section first formulates the problem, after which Section 7.4.2 will present an algorithm for tracking problems, that is able to guarantee robust constraint satisfaction by means of a control invariant set.

7.4.1 Problem formulation

As in previous sections we still consider LPV systems of the form (3.1)-(3.2) subject to constraints (2.10)-(2.11). While in all previous sections and chapters a control objective

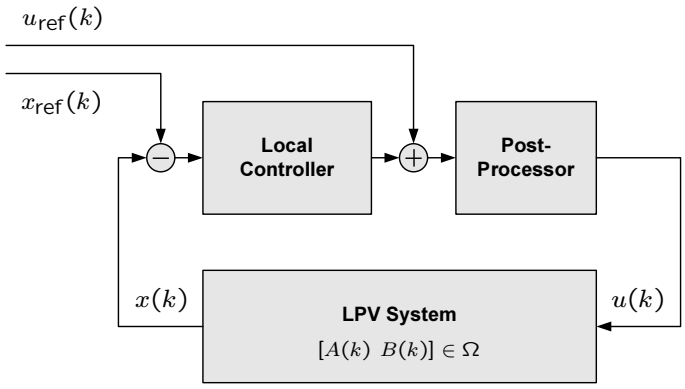


Figure 7.3: A schematic representation of the tracking setup discussed in this section. A local MPC controller designed for stabilization problems is employed by performing static reference insertion in a straightforward way. Despite the simplicity of the setup, the post-processor can still guarantee constraint satisfaction and recursive feasibility.

(1.3) with $x_{\text{ref}}(k) \equiv 0, u_{\text{ref}}(k) \equiv 0, \forall k \in \mathbb{N}$ is assumed we now explicitly consider the case when $x_{\text{ref}}(k) \neq 0, u_{\text{ref}}(k) \neq 0$.

The only purpose of this section is to obtain recursive feasibility in this setting, the reference trajectory is not required to satisfy any smoothness condition , any asymptotic conditions or any relationship (e.g., steady state conditions) with the dynamical model. It is not even required that the reference trajectory satisfies the state or input constraints.

We consider the simplified setting of tracking without preview, which means that at time k, the future part $x_{\text{ref}}(k+i), u_{\text{ref}}(k+i), i > 0$ is not available to the controller. The problem can now be formalized as follows:

Problem 7.1 (Trajectory tracking without preview for constrained LPV systems). *Consider an LPV system of the form* (3.1)-(3.2) *subject to constraints* (2.10)-(2.11) *and consider arbitrary reference trajectories* $x_{\text{ref}}(k) \in \mathbb{R}^{n_x}, u_{\text{ref}}(k) \in \mathbb{R}^{n_u}$. *Design a static state feedback control law* $u(k) = \kappa_{\text{tracking}}(x(k), x_{\text{ref}}(k), u_{\text{ref}}(k))$ *with the aim of minimizing* (1.3) *and such that constraints* (2.10)-(2.11) *are satisfied at all times.*

The following section discusses, as a proof-of-concept for the successful application of control invariant sets in a tracking setting, a possible way for solving this problem.

7.4.2 Algorithm synthesis

Similar to Section 7.3 we consider a local controller, denoted as $\kappa_{\text{local}}(\cdot)$, which can be either an MPC controller or a linear feedback law. No restrictions regarding recursive feasibility or guaranteed constraint satisfaction apply. Furthermore, we consider a control invariant set $\overline{\mathcal{S}} = \{x \in \mathbb{R}^{n_x} | A_{\overline{\mathcal{S}}} x \leq \mathbf{1}\}$ that is used to guarantee recursive feasibility.

As is indicated in Figure 7.3 we essentially apply at every time instant k the local

controller $\kappa_{\rm local}(\cdot)$:

$$\tilde{u}(k) = \kappa_{\rm local}(\tilde{x}(k)), \tag{7.8}$$

with $\tilde{u}(k) \triangleq u(k) - u_{\rm ref}(k)$ and $\tilde{x}(k) \triangleq x(k) - x_{\rm ref}(k)$.

$\kappa_{\rm local}$ can be any controller designed for e.g. disturbance rejection, optimal tracking of step signals, etc ..., but for the purpose of guaranteeing constraint satisfaction, the actual choice of $\kappa_{\rm local}$ is irrelevant. The following algorithm post-processes the input suggested by the local control law in order to obtain a control action that lies as close as possible to this suggested control action, while guaranteeing constraint satisfaction and recursive feasibility:

Algorithm 7.3 (Trajectory tracking using control invariant sets). *Consider an LPV system of the form* (3.1)-(3.2) *subject to constraints* (2.10)-(2.11) *and reference state and input trajectories* $x_{\rm ref}(k) \in \mathbb{R}^{n_x}, u_{\rm ref}(k) \in \mathbb{R}^{n_u}$. *Furthermore, consider a feasible control invariant set* $\overline{\mathcal{S}}$ *and a state feedback controller* $\kappa_{\rm local}$. *At every time instant* k, *given the current state* $x(k)$, *perform the following steps:*

- *Calculate* $u_{\rm f}(k) := u_{\rm ref} + \kappa_{\rm local}(x(k) - x_{\rm ref}(k))$ *and solve the following optimization problem:*

$$\min_{u(k)} \quad \|u(k) - u_{\rm f}(k)\|^2, \tag{7.9a}$$

$$\text{s.t.} \quad [x(k); u(k)] \in \overline{\mathcal{S}}_{xu}, \tag{7.9b}$$

  with $\overline{\mathcal{S}}_{xu}$ *defined as*

$$\overline{\mathcal{S}}_{xu} \triangleq \{[x; u] \in \mathbb{R}^{n_x+n_u} | u \in \mathcal{U}, A_j x + B_j u \in \overline{\mathcal{S}}, j = 1, \ldots, r\}. \tag{7.10}$$

- *Apply input* $u(k)$ *to the system.*

This simple control setup, regardless of $\kappa_{\rm local}$ guarantees constraint satisfaction if the initial state $x(0)$ lies within the control invariant set $\overline{\mathcal{S}}$.

Lemma 7.6 (Robust recursive feasibility). *Algorithm 7.3 is recursively feasible.*

Proof: One can see that optimization problem (7.9) is feasible if $\exists u(k) : [x(k); u(k)] \in \overline{\mathcal{S}}_{xu}$. Due to control-invariance of $\overline{\mathcal{S}}$ and the definition of $\overline{\mathcal{S}}_{xu}$, this condition is satisfied if $x(k) \in \overline{\mathcal{S}}$. Consequently, due to the definition of $\overline{\mathcal{S}}_{xu}$, it is guaranteed that $x(k+1) \in \overline{\mathcal{S}}$, which then guarantees that $\exists u(k+1) : [x(k+1); u(k+1)] \in \overline{\mathcal{S}}_{xu}$, which in turn guarantees that optimization problem (7.9) is feasible at time $k+1$. □

This lemma shows that control invariant sets can indeed be used to guarantee constraint satisfaction in control settings where this is normally not guaranteed. Extensions towards tracking *with* preview (see [136]) or off-set free setpoint tracking are straightforward, by appropriately designing $\kappa_{\rm local}$. However, the post-processing step (7.9) can remain unchanged, which is the strength of this approach.

7.5 Example

In this section we illustrate the algorithms introduced in this chapter by retaking the same numerical example as used in previous chapters. Section 7.5.1 first illustrates Algorithm 7.2, after which a tracking problem is solved in Section 7.5.2 by means of Algorithm 7.3.

7.5.1 Stabilization problem

In this section we still consider the same LPV system (4.56)-(4.57) as discussed in Sections 4.2.7, 4.3.5 and 5.4.2. The aim is to improve the feasible region of the algorithms discussed there by means of Algorithm 7.2. The following straightforward design choices are made:

- The local MPC controller κ_1 is chosen as the $\mathcal{P}$-RMPC controller with $N = 6$ designed in Section 4.3.5. Based on the results described in Section 5.4.2, $\gamma = 0.3$ is chosen. The resulting invariant set for the augmented system is described by 91 constraints.
- $\mathcal{S}_2$ is chosen as the control invariant set obtained by applying Algorithm 6.6 with $\lambda = 1$ and $\gamma_{\mathrm{Pre}} = 0.01$. The resulting set is described by 22 constraints.

The different regions (cfr. Figure 7.2) of operation thus obtained are depicted in figure 7.4. The control invariant set clearly is significantly larger than the feasible region of the local MPC controller.

In order to assess feasibility and optimality, we first compute the trajectories resulting from applying the set-induced controller $\kappa_{\mathcal{S}_2}$ to the system, starting from initial states near the boundary of $\mathcal{S}_2$. These trajectories are depicted in Figure 7.5, while the

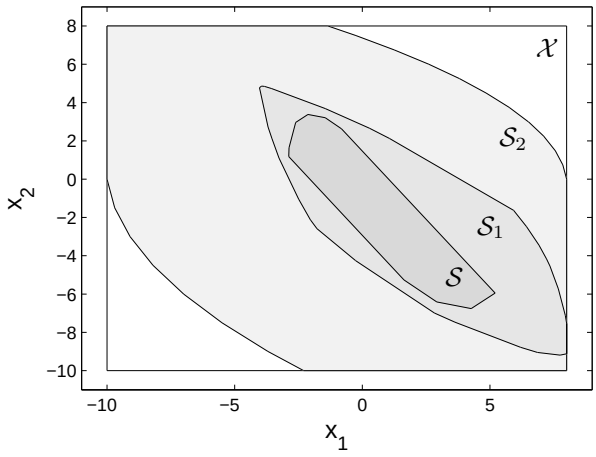


Figure 7.4: The different regions of operation of Algorithm 7.2 (as explained in Figure 7.2) for the numerical example under consideration.

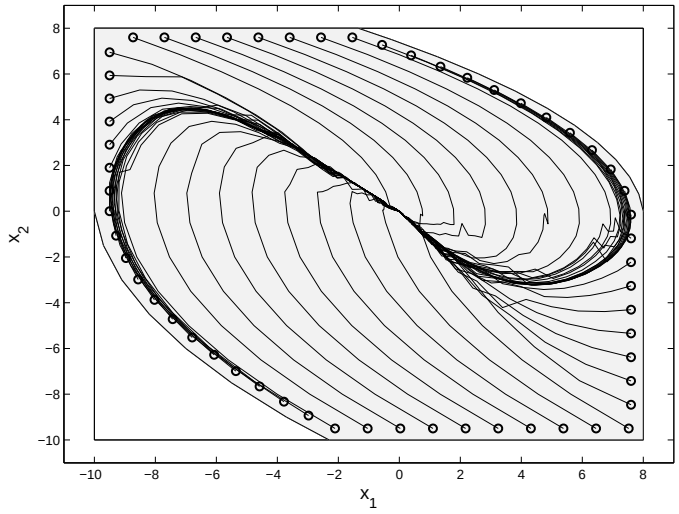


Figure 7.5: Trajectories generated by the set-induced controller $\kappa_{\mathcal{S}_2}$ for initial states near the boundary of $\mathcal{S}_2$. The set $\mathcal{S}_2$ is calculated by applying Algorithm 6.6 to system (4.56)-(4.57) using $\lambda = 1$ and $\gamma_{\mathrm{Pre}} = 0.01$. The real system behavior is chosen as the LTI system defined by A_2, B_2.

corresponding input sequences are depicted in 7.6. Real system behavior was taken as the LTI system described by A_2, B_2. One can see that feasibility is guaranteed for all states lying inside $\mathcal{S}_2$, which shows that (7.2) is satisfied for $i = 2$. However, as Figure 7.6 indicates, the control behavior is extremely nervous and non-smooth, which is due to the LP formulation (it is known that optimal solutions of parameterized LPs are non-continuous functions of the involved parameters (see [21] and references therein) and the fact that the control objective is not taken into account.

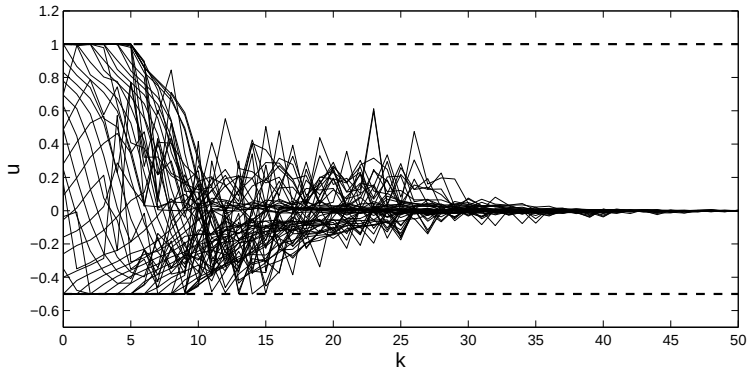


Figure 7.6: Input sequences corresponding to the trajectories shown in Figure 7.5.

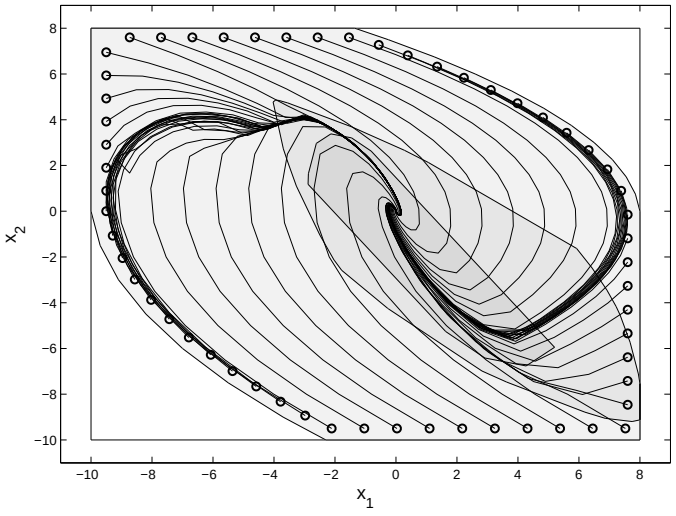


Figure 7.7: Trajectories generated by Algorithm 7.2 for initial states near the boundary of $\mathcal{S}_2$. The set $\mathcal{S}_2$ is calculated by applying Algorithm 6.6 to system (4.56)-(4.57) using $\lambda = 1$ and $\gamma_{\mathrm{Pre}} = 0.01$. The inner controller was chosen as the $\mathcal{P}$-RMPC controller with $N = 6$ described in Section 4.3.5, but with $\gamma = 0.3$. The real system behavior is chosen as the LTI system defined by A_2, B_2.

Figures 7.7 and 7.8 respectively show state and input trajectories resulting from Algorithm 7.2. One can see that the input trajectories are qualitatively better and that the trajectories are also markedly different than those resulting from $\kappa_{\mathcal{S}_2}$. The average control cost per trajectory is 448.61 when using Algorithm 7.2 compared to 477.6 when only using $\kappa_{\mathcal{S}_2}$. This shows that Algorithm 7.2 significantly improves local optimality, because local behavior only has a relatively small contribution to the total control cost

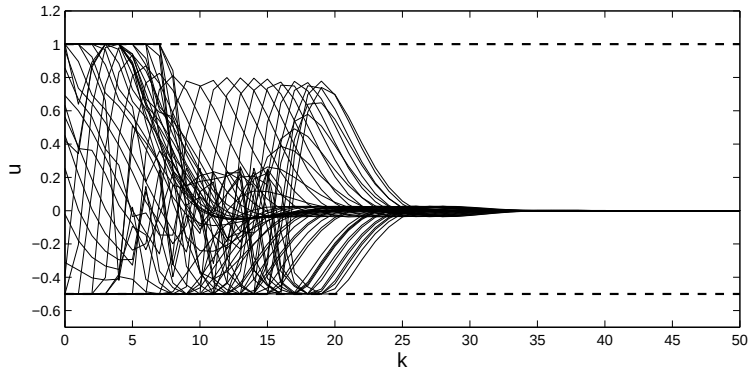


Figure 7.8: Input sequences corresponding to the trajectories shown in Figure 7.7.

and the total control cost still differs noticeably.

Average computation time per iteration was < 0.1 seconds (2.4GHz x86 CPU) for both Algorithm 7.2 and controller $\kappa_{\mathcal{S}_2}$, which is too small compared to the overhead incurred by calling external optimization solvers and hence no clear comparison can be made, besides the observation that both algorithms seem to have a computational complexity that is in the same order of magnitude.

7.5.2 Tracking problem

In this section we show that the same control invariant set used in the previous section can also be used for guaranteeing constraint satisfaction for tracking control problems, by means of Algorithm 7.3.

We consider the same system (4.56)-(4.57) and want to steer the system towards the following reference trajectory:

$$x_{\rm ref}(k) = [a(k); 0], \quad a(k) \in (-10, 8), \qquad k \in \mathbb{N}, \tag{7.11}$$

$$u_{\rm ref}(k) = 0, \qquad k \in \mathbb{N}. \tag{7.12}$$

The sequence $a(k), k \in \mathbb{N}$ is chosen as a piecewise constant function, with switching between different values occurring every 20 time instants. The reference trajectory is depicted in red in Figure 7.10. In order to illustrate the efficiency of Algorithm 7.3 even if the local controller $\kappa_{\rm local}$ is chosen in a naive way, we choose $u(k) = -K_1 x(k)$, with K_1 given the same value as in (4.58). We compare Algorithm 7.3 with the following two controllers:

$$u(k) = u_{\rm ref}(k) - K_1(x(k) - x_{\rm ref}(k)), \tag{7.13}$$

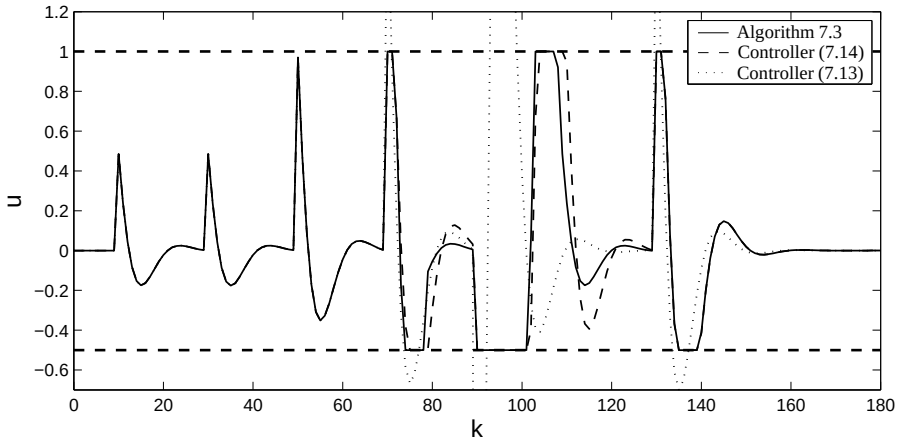


Figure 7.9: Inputs generated by the three different controllers under consideration, when applied to system (4.56)-(4.57) with the aim of tracking the trajectory shown in Figure 7.10.

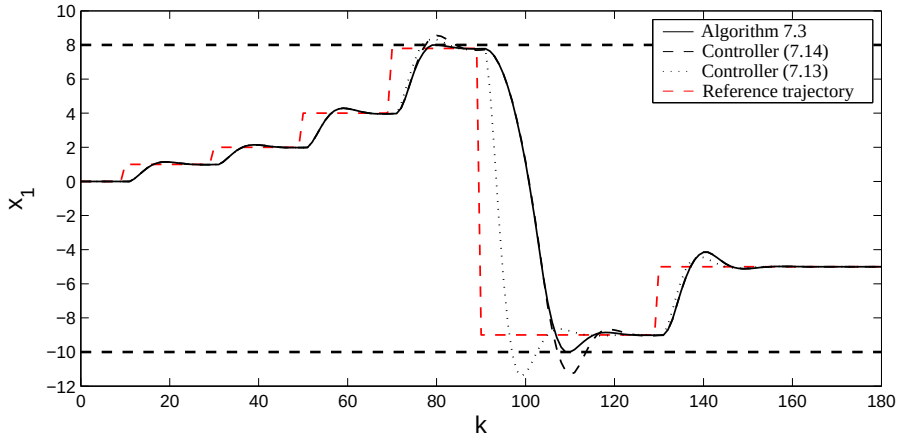


Figure 7.10: The first state component of the closed-loop systems obtained by combining system (4.56)-(4.57) with the three different controllers under consideration.

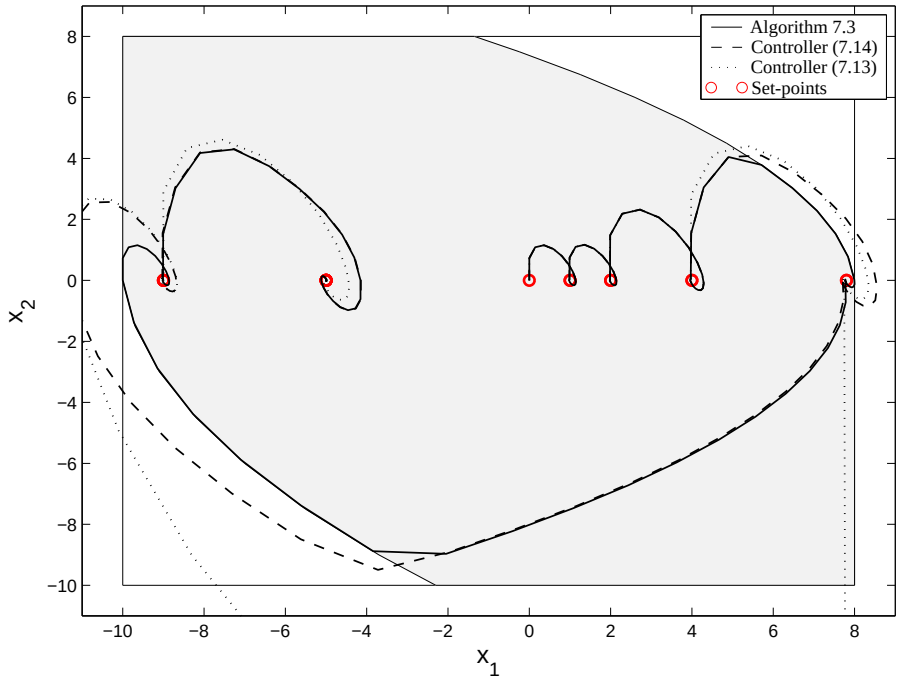


Figure 7.11: Phase plot of the closed-loop systems obtained by combining system (4.56)-(4.57) with the three different controllers under consideration.

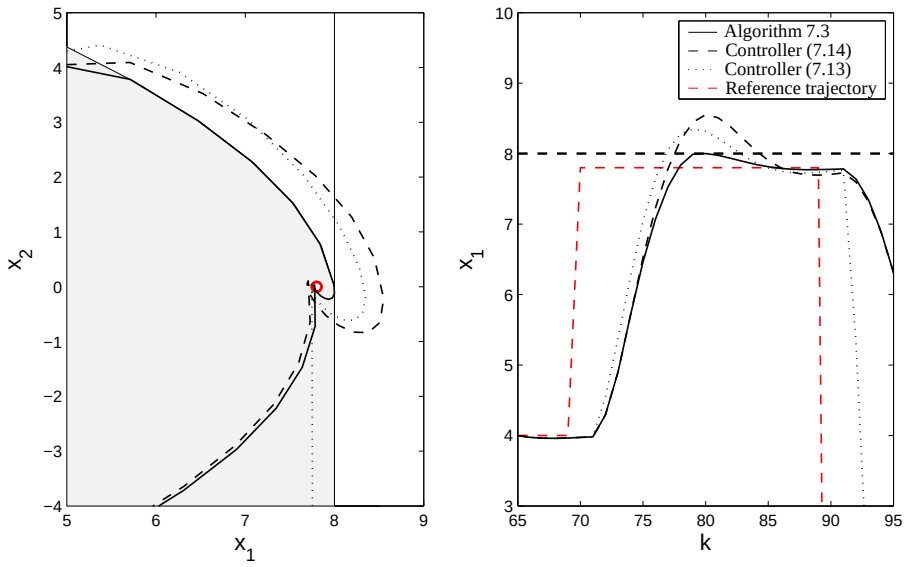


Figure 7.12: Close-up of Figures 7.11 and 7.10 in order to compare constraint handling of the three different controllers under consideration, when operating near the state constraints.

and

$$u(k) = \mathrm{trim}_{[-0.5,1]}\left(u_{\mathrm{ref}}(k) - K_1(x(k) - x_{\mathrm{ref}}(k))\right), \tag{7.14}$$

with $\mathrm{trim}_{[a,b]}(u)$ defined as

$$\mathrm{trim}_{[a,b]}(u) \triangleq \begin{cases} b, & u > b, \\ u, & u \in [a,b], \\ a, & u < a. \end{cases} \tag{7.15}$$

Figure 7.9 shows the inputs generated by the three controllers under consideration. As expected, controller (7.14) and Algorithm 7.3 respect the imposed input constraints at all times, whereas controller (7.13) violates the input constraint at several points in time.

Figure 7.10 shows that controller (7.14), while guaranteeing input constraint satisfaction, causes state constraint violations when the reference trajectory closely approaches these constraints or when large reference steps are applied. However, Algorithm 7.2 is able to also guarantee satisfaction of the state constraints. This is accomplished by adjusting the applied input such that the overshoot, after applying a reference step, is kept small enough in order to stay within the imposed constraints.

Figure 7.11 shows phase portraits of the three different closed-loop systems, together with a depiction of the control invariant set used in Algorithm 7.3. It is clear that this algorithm makes sure that at all times the state is kept within the control invariant set,

such that it is always guaranteed that there exists a feasible input sequence that keeps the state within the state constraints.

Figure 7.11 finally shows a close-up of Figures 7.11 and 7.10 to show in detail the difference in control behavior of the three controllers when changing the reference state from $[4; 0]$ to $[7.8; 0]$.

It should be noted that off-set free setpoint tracking is obtained in this example because $x_{\mathrm{ref}}(k), u_{\mathrm{ref}}(k)$ are chosen such that $x_{\mathrm{ref}}(k) = Ax_{\mathrm{ref}}(k) + Bu_{\mathrm{ref}}(k), \forall [A\ B] \in \Omega$. Still, this example clearly shows that useful results can already be obtained when appropriately combining a simple linear controller with the knowledge of allowable control behavior contained in a feasible control invariant set. Further extensions are straightforward and form the subject of current research.

7.6 Conclusions

This chapter discussed how control invariant sets can be used in MPC algorithms to further improve constraint handling. Two different settings are discussed: stabilization and tracking.

First of all it is shown that, by extending general interpolation towards non-linear control laws, control invariant sets and their induced controllers can be used to extend the feasible region of the robustly stabilizing MPC controllers discussed in Chapter 4. In this way the feasible region of the resulting control law is the largest feasible region theoretically possible, if the control invariant set is taken as the MCAS. This significant advantage comes only at the cost of a moderate amount of additional computational complexity.

Secondly, it is shown that also in the context of tracking control problems, control invariant sets can be used to guarantee robust constraint satisfaction of any (linear or non-linear) state feedback controller. This is shown as a proof-of-concept for the case of tracking without preview, but extensions towards more complex and useful settings is relatively straightforward.

On a general note, it can be said that control invariant sets have a large potential for improved constraint handling in stabilization and tracking control problems. As shown in this chapter, this potential can be put to use by means of standard techniques such as general interpolation and simple convex optimization problems, but many possibilities are probably yet to be discovered.

Chapter 8

Case Studies

"In theory, there is no difference between theory and practice.
In practice, there is."

– Chuck Reid –

In this chapter the algorithms discussed in the previous chapters are applied to two practical examples. The first case study discusses a control problem encountered in steel rolling mills. The second case study consists of a chemical process for the production of a copolymer. Both examples are constrained MIMO systems of relatively high order compared to the numerical examples considered up till now and are hence well suited to illustrate the practical efficacy of the algorithms presented in this thesis.

8.1 Steel rolling mill

This section considers the control of a steel rolling process. More specifically, the problem of controlling the tension in hot strips that pass through a finishing mill is tackled. Section 8.1.1 starts with a general overview of the process, after which Section 8.1.2 defines the control problem more precisely. Sections 8.1.4 to 8.1.6 then present results obtained with the different control strategies introduced in this thesis. Section 8.1.7 finally discusses the control performance of the obtained controllers.

8.1.1 Process description

For reasons of confidentiality not all details of the process can be given here. Therefore, all physical variables are omitted or rescaled and reported without units. For more information we refer to [36]. In what follows the general context of the control problem is sketched, in order to illustrate the practical relevance.

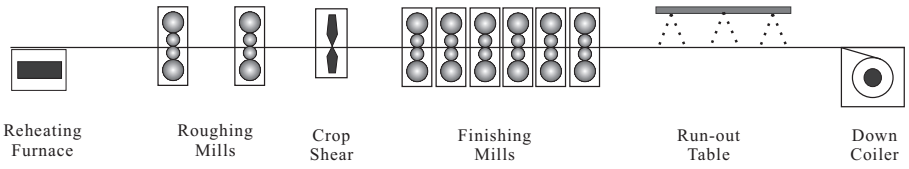


Figure 8.1: Typical layout of a hot strip mill. Slabs are heated in a furnace after which roughing and finishing mills reduce the thickness of the strips. Finished strips are cooled down and coiled for later transportations. Image taken from [36].

The main aim of a hot strip mill process is to produce steel strips with a specified thickness and width. To this aim steel slabs are heated after which their thickness is reduced by roughing and finishing mills. Figure 8.1 gives a typical layout of such a process. The reverse roughing mills result in an initial thickness reduction of the slabs, after which the finishing mills reduce the thickness of the strips to the specified value.

Figure 8.2 depicts a typical control loop present in between two finishing mills. The main aim of this looper-tension control loop is to control the tension of the strip passing through the finishing mills. Excessive tension can result in width and thickness reductions and can hence have a detrimental effect on the dimensional quality of the end product. On the other hand, when the strip tension is too low, the mass flow through the finishing mills can become unstable. Therefore the tension should be kept around a fixed value. This can be achieved by means of motors controlling the speed of the mills.

Another actuator present between two finishing mill stands is the looper, whose angle can be changed in order to compensate for sudden mass flow irregularities. However,

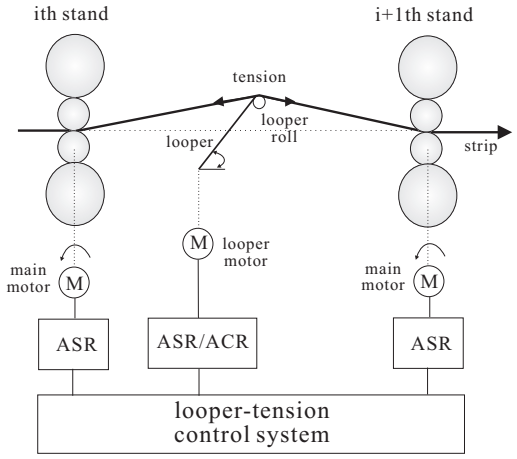


Figure 8.2: Looper and tension control system between two finishing mill stands. The strip tension and the looper angle are controlled by means of electrical motors. Image taken from [36].

	$\gamma = 0.10$	$\gamma = 0.05$	$\gamma = 0.00$
$\mathbf{N = 5}$	202 (285s)	272 (444s)	1370 (7789s)
$\mathbf{N = 10}$	362 (950s)	484 (1696s)	>4000 (>50000s)[1]
$\mathbf{N = 15}$	528 (2820s)	716 (5458s)	/
$\mathbf{N = 20}$	674 (6712s)	946 (12607s)	/
$\mathbf{N = 25}$	828 (12253s)	1158 (25405s)	/

Table 8.1: Total number of constraints and computation times (between brackets) for invariant sets of the augmented system (4.63)-(4.64) for different values of N. The invariant sets are computed using Algorithm 5.2 with different values of γ. See also Figure 8.4. ([1]conservative lower bounds based on extrapolation of partial results)

in order to maintain maximal flexibility, the looper angle should also be kept around a fixed value. The looper angle can be controlled by means of an electrical motor.

8.1.2 Problem formulation

The looper angle and finishing mill speed mechanism are modeled by means of non-linear first-principles model, which is then linearized around the desired operating point. The model has 2 inputs (controlling the two electrical motors), 6 states and 2 outputs (looper angle deviation and strip tension deviation). In order to gain robustness against non-linearities present in the real process, a linear uncertain model is used for controller design. A polytopic uncertainty set with $r = 2$ is constructed based on two linear models, linearized around two different looper angles. We hence have a robust control problem with dimensions $n_x = 6, n_u = 2, r = 2$. States and inputs are subject to component-wise upper and lower bounds. More specifically, the looper angle deviation is restricted to $[-0.2\ 0.2]$, while the strip tension deviation is restricted to $[-2\ 2]$. A quadratic control objective (1.3) is imposed with

$$Q = C^{\mathrm{T}} \begin{bmatrix} 0.25 & 0 \\ 0 & 64 \end{bmatrix} C, \qquad R = 0.001I, \tag{8.1}$$

where the scaling factors in Q compensate for the differences in units.

8.1.3 Design Specifications

The aim of this example is to provide a comparative case study in a specification driven design process of the robust MPC algorithms discussed in this thesis. We will apply the design procedures of the different MPC algorithms with the aim of satisfying the following specifications:

- The controller should be locally optimal according to the above criteria.
- The controller should have a region of operation that spans the entire range of allowed strip tension deviation and that spans a looper angle range that is as large as possible.

- Due to the fast sampling time of the system, the computation time per second should not exceed 0.02 seconds.
- The controller should be able to cope with sudden system perturbations of sufficiently large amplitudes.

Obviously, the above specifications are conflicting (e.g., operation range vs. computational complexity), so possibly no clearly optimal winner will be found.

8.1.4 Quasi-infinite horizon MPC

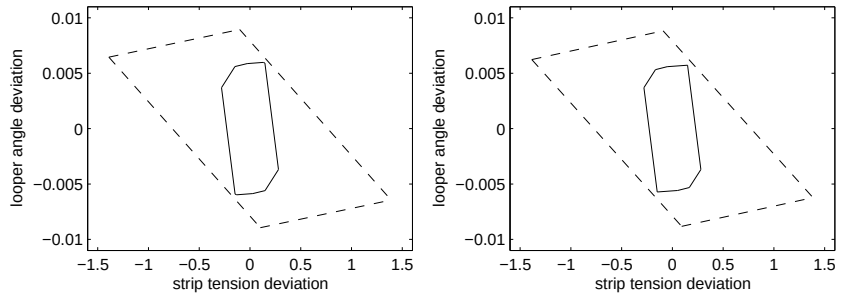
A first step into the design of a controller based on Algorithm 4.7 is the construction of a locally optimal linear feedback controller. To this end, we use Algorithm 4.2 with $\bar{x}$ chosen close to the origin. A value of $\bar{x} = 0.001e_3$ was chosen, which corresponds to a slight deviation of the looper angle from the operating point. In further sections we refer to this controller as κ_{local}. Figure 8.3 shows invariant sets for this local controller, computed using Algorithm 5.2 for different values of γ. It is clear that choosing $\gamma = 0.05$ results in a significant complexity reduction without any apparent volume loss and hence we will also use this value for later computations.

We can now construct a $\mathcal{P}$-RMPC controller (Algorithm 4.7, Section 4.3) using this locally optimal controller. Since both inputs can be considered to be equally important, we choose $E = I$. Figure 8.4 shows feasible regions for different values of N. Table 8.1 reports the resulting number of constraints and the computation times for the different horizon lengths.

Choosing $\gamma = 0$ is obviously not a practical choice due to the rapid increase of the number of constraints as a function of N. This leads to a prohibitively large computation time for constructing the invariant sets as well as a large on-line computational load. The two other values of γ (0.05 and 0.1) lead to a significantly reduced number of constraints. The increase is almost exactly linear as a function of N, which shows that Algorithm 5.2 enables the use of long prediction horizons in robust MPC algorithms. Figure 8.4 shows the resulting feasible regions corresponding to different values for N and γ. It is clear that the operating region increases as a function of N and is significantly larger than the invariant set for the locally optimal controller depicted in Figure 8.3. However, looper angle deviation allowed by the feasible region is still fairly small and increases very slowly as a function of N. Therefore the next section aims to construct a controller with a larger feasible region by means of general interpolation.

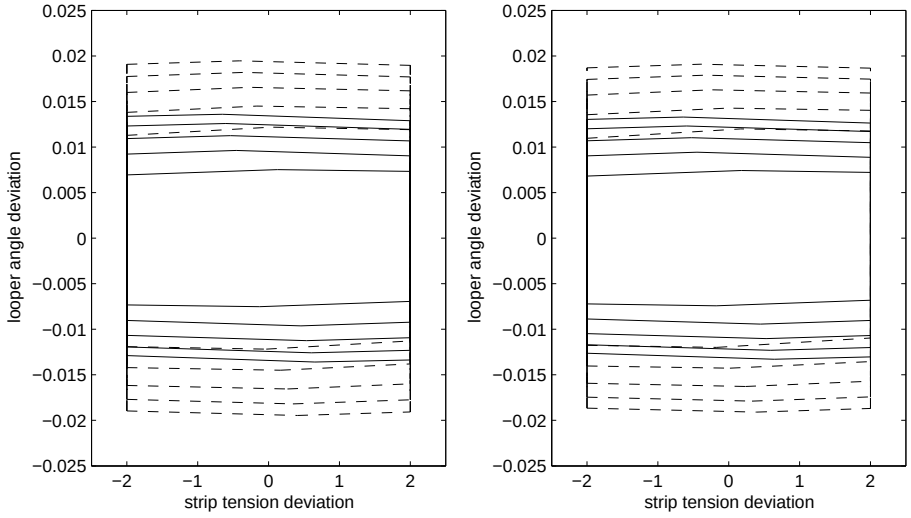
8.1.5 Interpolation based MPC

An alternative for obtaining an MPC controller with a large feasible region is to use general interpolation as discussed in Section 4.2. In order to construct an interpolation based controller we first compute additional linear control laws with enlarged corresponding invariant sets. To this end we apply Algorithm 4.2 and place $\bar{x}$ farther away from the origin. Since we aim to increase the size of the feasible region in the 3rd dimension, which corresponds to the looper angle, we try $\bar{x} = 0.01e_3$ and $\bar{x} = 0.02e_3$. These values lie outside the invariant set of the locally optimal controller.



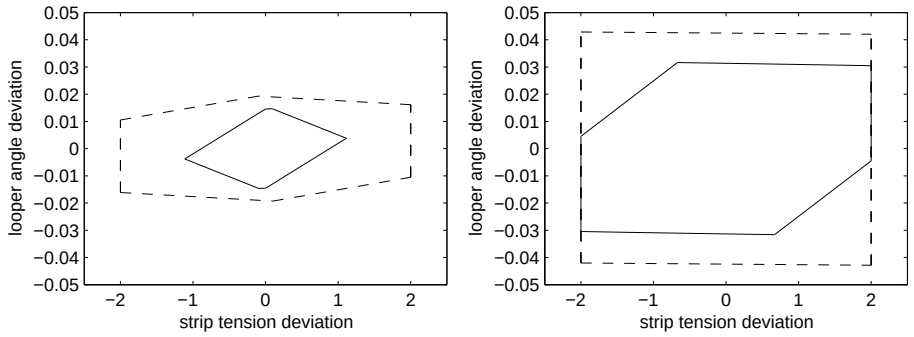
(a) $\gamma = 0$ (b) $\gamma = 0.05$

Figure 8.3: Invariant sets for the local controller designed for the hot strip mill model, computed using Algorithm 5.2. Solid lines represent intersections, dashed lines represent projections onto the two most important state dimensions. The resulting number of constraints and computation times were a) 198 constraints, 278 seconds, b) 60 constraints, 87 seconds. No significant volume differences can be observed.



(a) $\gamma = 0.05$ (b) $\gamma = 0.1$

Figure 8.4: Feasible region for $\mathcal{P}$-RMPC controller for different values of N. The invariant sets were computed using Algorithm 5.2 using different values of γ. Solid lines represent intersections, dashed lines represent projections onto the two most important state dimensions.



(a) Invariant set of κ_1 (b) Invariant set of κ_2

Figure 8.5: Invariant set for two linear control laws with enlarged feasible region for use in GIMPC(2). The controller is computed using Algorithm 4.2, while the invariant set is computed using Algorithm 5.2 with $\gamma = 0.05$. Solid lines represent intersections, dashed lines represent projections onto the two most important state dimensions.

In further sections, the thus obtained linear control laws are referred to as κ_1 and κ_2. The invariant sets of κ_1 and κ_2 are depicted in Figure 8.5 and are clearly larger than the invariant set of κ_{local}.

We now construct an interpolation based controller based on κ_{local} and either κ_1 or κ_2. In order to maximize the feasible region, we choose to construct a $\mathcal{P}$-GIMPC2 controller using method 2 described in Section 4.2.5.2. The obtained feasible regions are shown in Figure 8.6.

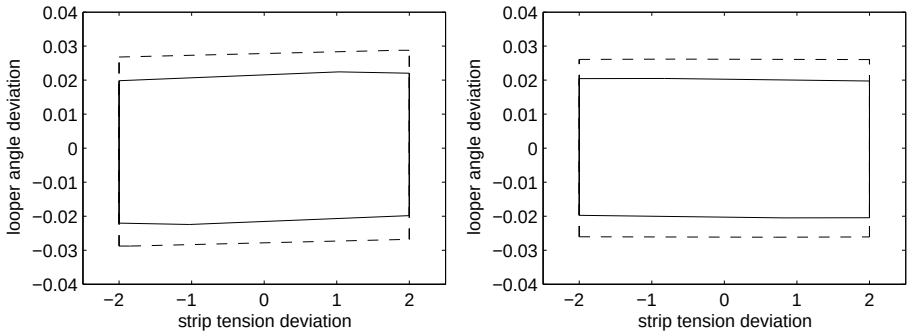
Against expectations, the feasible regions of the two different $\mathcal{P}$-GIMPC2 controllers are more or less identical in size. After comparison of Figures 8.5 and 8.6, it is found that the feasible region of the controller based on κ_{local} and κ_2 is smaller than the invariant set of κ_2. We therefore conclude that the relatively small feasible region of the latter GIMPC2 controller is due to the fact that $\gamma = 0.1$ is chosen too large. However, smaller values of γ resulted in an excessive number of constraints. Therefore we choose the controller based on κ_{local} and κ_1, with $\gamma = 0.1$ to compute the invariant set of the associated augmented system.

8.1.6 MPC using control invariant sets

Finally, we try to increase the size of the feasible region even further by means of Algorithm 7.2. This algorithm employs general interpolation between non-linear control laws in order to make the feasible region equal to the MCAS of the system. Therefore we use Algorithm 6.6 (with $\lambda = 1, \lambda' = 0.95$ and different values of γ_{Pre} and γ_{proj}) in order to compute the maximal control admissible set for the given system. Figure 8.7 shows the resulting control invariant sets. Disappointingly these sets turn out to be smaller than the feasible regions of the $\mathcal{P}$-RMPC and $\mathcal{P}$-GIMPC2 controllers. Therefore this design route is not explored any further.

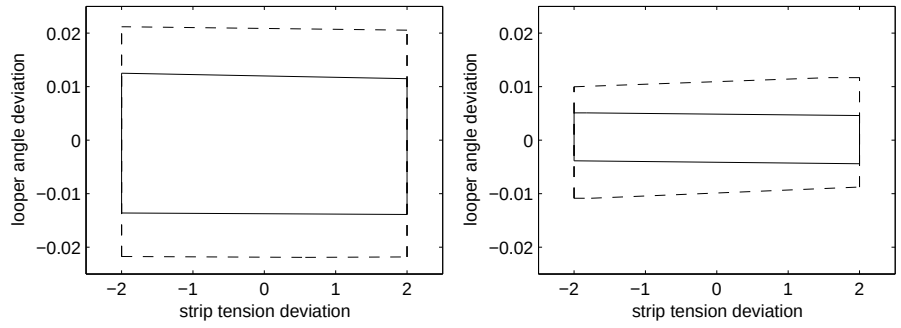
	Algorithm	Parameters	Constraints	Variables
MPC 1	$\mathcal{P}$-RMPC	$N = 5, E = I, \gamma = 0.1$	202	10
MPC 2	$\mathcal{P}$-RMPC	$N = 25, E = I, \gamma = 0.1$	828	50
MPC 3	$\mathcal{P}$-GIMPC2	$\kappa_{\text{local}}, \kappa_1, \gamma = 0.1$	750	6

Table 8.2: Overview of the different controllers compared in this section.



(a) $\mathcal{P}$-GIMPC2 controller based on κ_{local} and κ_1. (b) $\mathcal{P}$-GIMPC2 controller based on κ_{local} and κ_2.

Figure 8.6: Feasible regions for $\mathcal{P}$-GIMPC2 controllers based on different linear control laws. The invariant sets were computed using Algorithm 5.2 with $\gamma = 0.05$. Solid lines represent intersections, dashed lines represent projections onto the two most important state dimensions.



(a) $\gamma_{\text{Pre}} = 0.02, \gamma_{\text{proj}} = 0.02$ (b) $\gamma_{\text{Pre}} = 0.05, \gamma_{\text{proj}} = 0.05$

Figure 8.7: Control invariant sets computed using Algorithm 6.6 using different values of γ_{Pre} and γ_{proj}.

	Infeasible at time step	CPU-t./iter.	Control cost
MPC 1	401	0.005s	0.2709
MPC 2	1001	0.029s	0.2709
MPC 3	1601	0.009s	0.2803

Table 8.3: Comparison of the 3 MPC controllers with respect to feasibility, computational complexity and optimality. The best scores of each category are underlined.

8.1.7 Simulation results

We now assess the control performance of the different controllers constructed in the previous sections by means of a simulation. An overview of the different controllers compared here is given in Table 8.2

In order to assess the control performance of the controller we perform a simulation with all three controllers. The initial state is chosen as $x(0) = 0$ and disturbance signals are added to the states corresponding to the looper angle and the strip tension. These disturbance signals are depicted in Figure 8.8. These signals consist of Gaussian noise and spikes with increasing magnitudes. This setting allows the following three aspects to be tested:

- **Feasibility:** Due to the fact that the disturbance signals have increasing amplitudes, it is expected that infeasibilities will occur sooner or later. The time steps when these infeasibilities occur give information on the size of the disturbances that can be tackled by the controllers.
- **Optimality:** For that part of the simulations where all three controllers are still

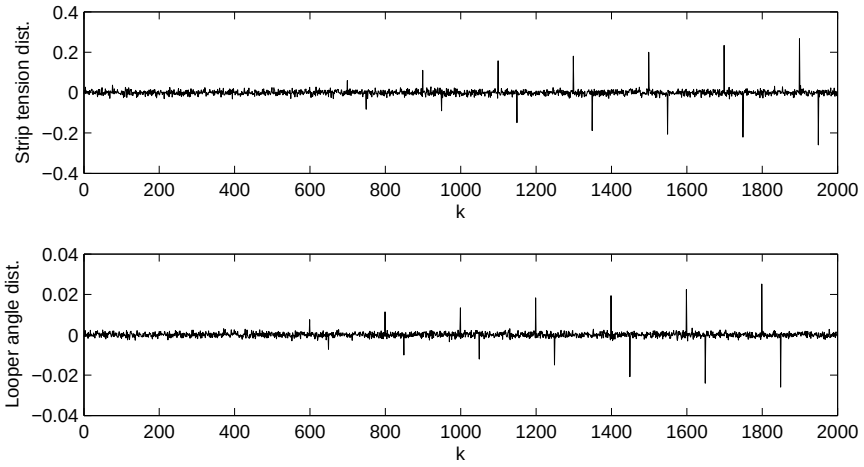


Figure 8.8: Disturbance signals used in the comparative test between the 3 MPC controllers.

feasible, one can compare the total control cost in terms of the quadratic control objective defined by the matrices Q and R. These costs give an idea of how well the controllers are able to reject the disturbances.

- **Computational efficiency:** Similarly, the computation time per iteration can be calculated for each algorithm.

Table 8.3 gives an overview of the performance of the different controllers with respect to the above aspects. The following conclusions can be drawn :

- Locally, $\mathcal{P}$-RMPC offers the most optimal control performance (according to the quadratic control objective), although the difference is not large compared to GIMPC2.
- Of the two $\mathcal{P}$-RMPC algorithms, only the short-horizon controller satisfies the imposed specifications regarding the maximum computational complexity, but it has relatively poor feasibility. This indicates that relatively small system perturbations can render the controller unusable.
- GIMPC2 has a sufficiently large marging with respect to the computational complexity specification and has the best results with respect to feasibility, indicating that the controller should be able to cope with relatively large system disturbances.

In summary, we can conclude that $GIMPC2$ seems to result in the best trade-off between the different design specifications. however, if local optimality is prioritized significantly above the ability to cope with large system disturbances, $\mathcal{P}$-RMPC should be used. However, the horizon length should be kept sufficiently small (e.g., not above 10) in order not to violate the computational complexity specification.

It should be noted that, due to the fact that all 3 controllers use QP optimization, the computational costs are very modest. If ellipsoidal invariant sets would have been used, the computational complexity would have been significantly higher due to the fact that SDP optimization would have to be used. This fact is also illustrated in the next section.

8.2 Copolymerization reactor

This section deals with the control of a copolymerization process in a continuously stirred tank reactor (CSTR). The aim of this section is not to reiterate the different design options discussed in the previous section, but to show the difference between $\mathcal{P}$-RMPC and $\mathcal{E}$-RMPC for high dimensional systems. It will become clear that, even though until recently the use of polyhedral invariant sets was thought to be limited to low dimensional systems, $\mathcal{P}$-RMPC significantly outperforms $\mathcal{E}$-RMPC in this example. These results and further information can also be found in [106] and [95] respectively.

8.2.1 Process description

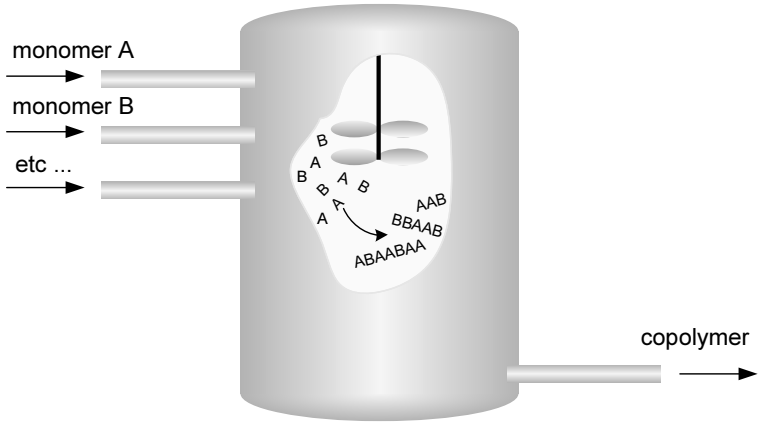


Figure 8.9: A continuously stirred tank reactor (CSTR) for a copolymerization reaction.

In this section we consider a control problem involving a CSTR reactor for a copolymerization process described in [37, 38]. Such a reactor consists of a tank to which the reagents are fed continuously. The contents of the tank are continuously stirred in order to obtain perfect mixing conditions. Simultaneously the reaction products are drained from the tank. The reaction products are separated from the rest of the drained fluid, which is fed back into the reactor through a recycle loop. However, this recycle loop is not considered in this section, since insufficient information on its dynamics was available.

The aim of a copolymerization process is to produce a copolymer of two different monomers, which will be referred to as monomer A and monomer B. The two monomers are fed into the CSTR, together with a solvent and other chemicals influencing the copolymerization reaction. The reaction product (i.e., the copolymer) consists of mixed chains of the two monomers. The two main properties of the reaction product that can be influenced are the average molar mass of the chains, which is related to the average length of the chains, and the average mass fraction of monomer A in

Input	Steady state	
monomer A mass feed rate (G_{af})	18.00	kg/h
monomer B mass feed rate (G_{bf})	89.99	kg/h
initiator mass feed rate (G_{if})	0.18	kg/h
solvent mass feed rate (G_{sf})	36.02	kg/h
chain transfer agent mass feed rate (G_{tf})	2.70	kg/h
inhibitor mass feed rate (G_{zf})	0.0003	kg/h
Output	**Steady state**	
polymer production rate (G_{p})	23.31	kg/h
monom. A mass fract. in polym. (Y_{ap})	0.56	
polymer molar mass (M_{p})	35003.48	g/mole
reactor temperature (T_{r})	353.00	K

Table 8.4: Overview of the input and output variables of the reactor model and their steady state values.

the copolymer, which is a measure for the relative number of monomer A molecules present in each chain.

The control problem considered here has the aim of maintaining a fixed average molar mass and a fixed average mass fraction of monomer A in the copolymer. To this end the feed rates of 6 different reagents can be manipulated by the controller. Two other important variables that also have to be kept constant are the reactor temperature and the production rate of the reactor.

8.2.2 Problem formulation

In order to control the process, the model described in [38] is used. The model is based on first principles and is represented as a set of non-linear coupled differential equations. The model has 6 inputs 12 states and 4 outputs. Table 8.4 gives an overview of the input and outputs of the model and their steady state values. This model is linearized around the operating point given in Table 8.4 and discretized in time with a sample time $T_s = 30$min. Input, state and output variables of the model are translated and normalized with respect to their steady state values, in order to improve numerical stability.

In order to obtain robustness with respect to the different dynamics in the neighborhood of the operating point, an LPV model is constructed based on two linear models, resulting from linearization around 90% and 110% of the steady state value of the monomer A concentration in the reactor.

Constraints are imposed on the inputs and states in order to ensure that a) the applied inputs do not deviate more than -20% and $+10\%$ from their steady state values, b) the outputs do not deviate more than $\pm 50\%$ from their steady state values and c) the states do not deviate more than $\pm 100\%$ from their steady state values.

A quadratic control objective (1.3) is imposed with $Q = 10C^{\mathrm{T}}C$ and $R = 0.001I$.

8.2.3 Design Specifications

The system under consideration here is markedly different from the system discussed in the first case study. This chemical system has a much smaller sampling frequency (1500 times smaller) and hence computational complexity is no important issue. The dimensionality of the system, however, is larger and the number of inputs is also significantly larger. Due to this higher dimensionality, constructing controllers with sufficiently large feasible regions is the main concern. The following design specifications are imposed in this case study:

- The controller should be locally optimal with respect to the above mentioned quadratic control objective.
- The computation time per iteration should be lower than 30 minutes.
- The controller should have a region of operation that is as large as possible.

In the following sections, robust MPC controllers will be designed in order to optimally satisfy these specifications.

8.2.4 Controller design

In this section we design $\mathcal{P}$-RMPC and $\mathcal{E}$-RMPC controllers for the given control setting. The design of both controllers is almost identical, except for the computation of the invariant set for the augmented system (4.63)-(4.64).

First we design a locally optimal controller. This is chosen as the locally optimal LQR controller at the operating point. The resulting controller was also found to be robustly stabilizing for the LPV system described in the previous section and hence can be used for designing a robust MPC controller.

Secondly, a choice has to be made regarding the E matrix in equation (4.61). Since one of the primary aims is to keep the concentration of the two monomers in the reactor at a constant level and because these two states are primarily governed by the first two inputs, we choose $E = [I; 0_{4\times 2}]$.

In the case of the $\mathcal{P}$-RMPC controller additional choices have to be made for the parameters related to the construction of the polyhedral invariant set for the augmented system. We try $\gamma = 0$ and $\gamma = 0.15$. The resulting feasible regions (for different horizon lengths N) are depicted in Figure 8.10, the number of constraints describing the obtained invariant sets are given in Table 8.5.

Ellipsoidal invariant sets for the $\mathcal{E}$-RMPC controller were computed using Algorithm 2.2. The resulting feasible region for $N = 25$ is shown in Figure 8.11.

In order to obtain MPC algorithms with further enlarged feasible region, attempts were made to compute control-invariant sets for the given system using Algorithm 6.6. However, even in the nominal case, the computation time was unacceptably large. Hence, this route was not further pursued. This also shows that more research is needed in this area.

N	# states in (4.63)	# constr. (Alg. 5.2)	# constr. (Alg. 2.4)
0	12	58	74
5	22	177	771
10	32	299	>2000
15	42	417	/
20	52	538	/
25	62	642	/

Table 8.5: Dimensionality of the augmented system (4.63) and the number of constraints of invariant sets for (4.63) for different prediction horizons N.

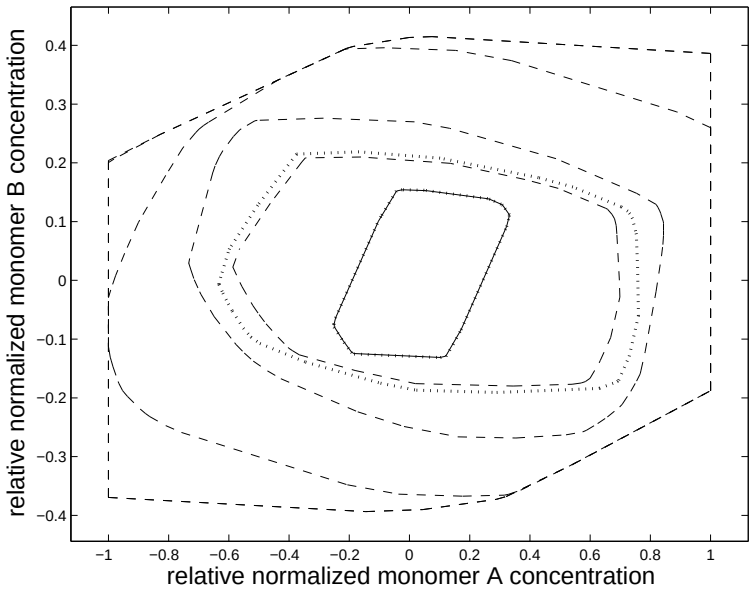


Figure 8.10: Feasible regions of $\mathcal{P}$-RMPC for $N = 0$ (solid) and $N = 5, \ldots, 25$ (dashed), computed using Algorithm 5.2 with $\gamma = 0.15$. Feasible regions for $N = 0, 5$, computed using Algorithm 2.4 are also depicted (dotted). Note that the two feasible regions for $N = 20$ and $N = 25$ only differ marginally and hence are hardly discernable. See also Table 8.5.

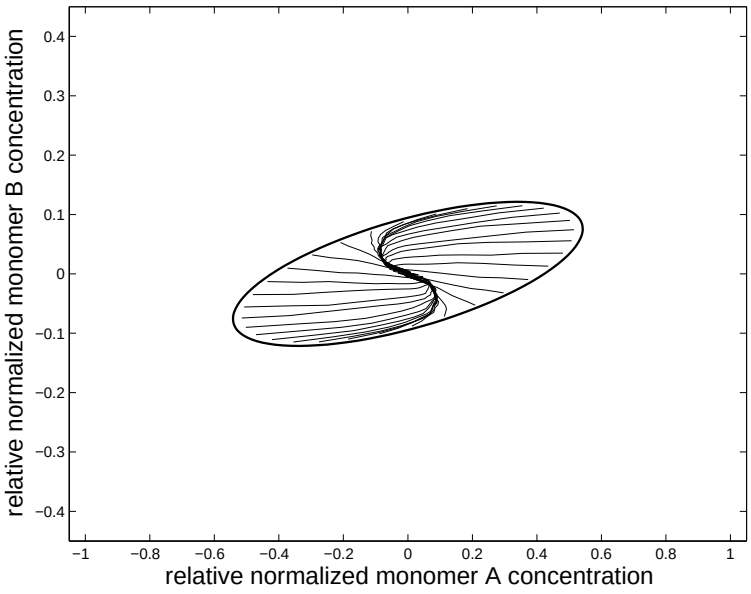


Figure 8.11: Trajectories for the $\mathcal{E}$-RMPC controller [70] for $N = 25$ starting from initial conditions starting near the boundary of the feasible region.

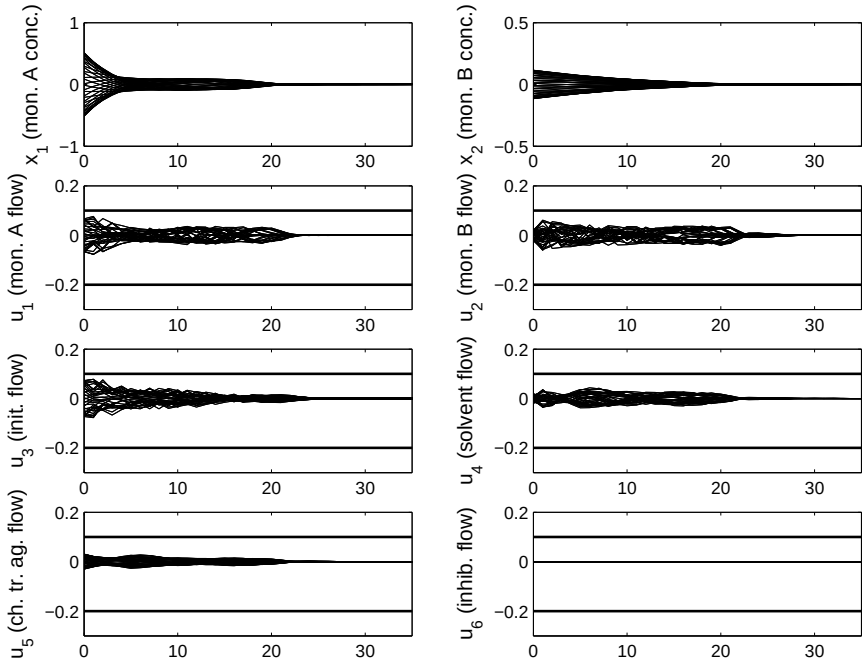


Figure 8.12: State and input sequence corresponding to the trajectories shown in Figure 8.11.

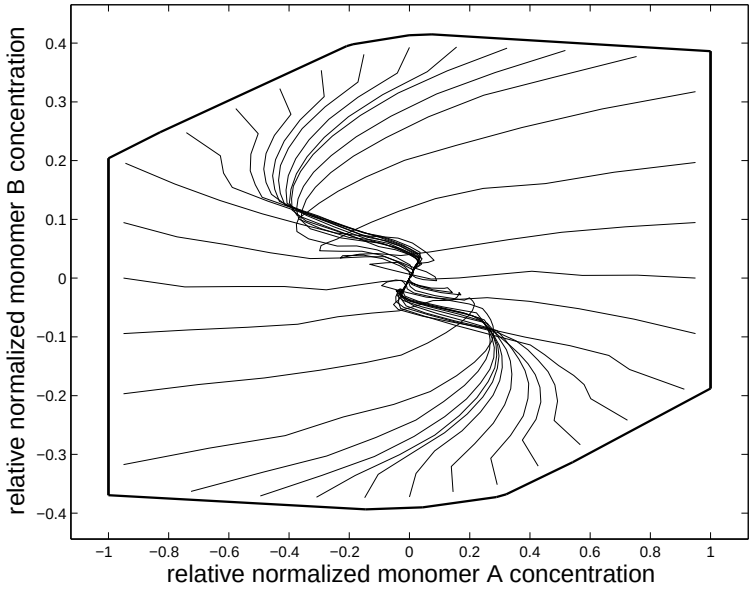


Figure 8.13: Trajectories for the $\mathcal{P}$-RMPC controller for $N = 25$ starting from initial conditions starting near the boundary of the feasible region.

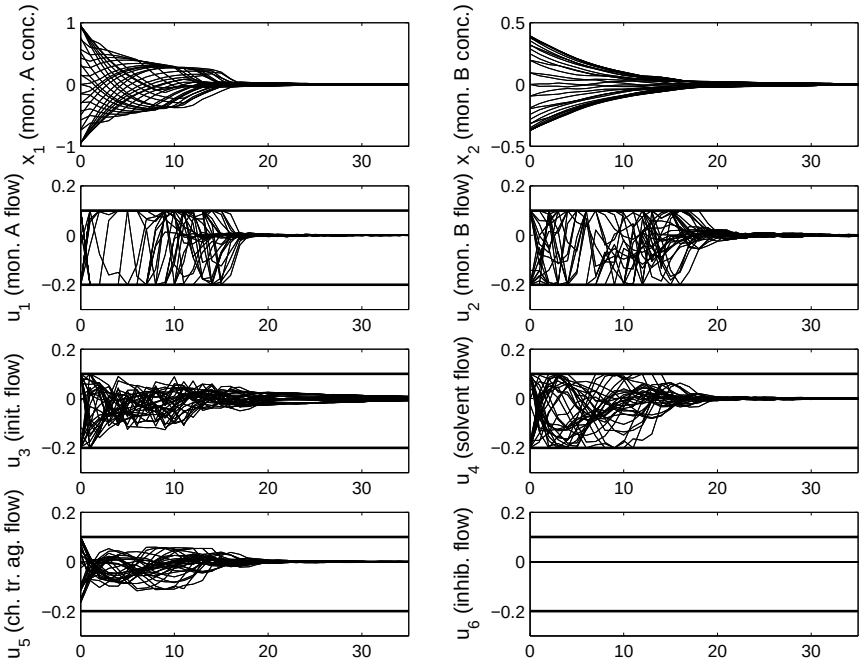


Figure 8.14: State and input sequence corresponding to the trajectories shown in Figure 8.13.

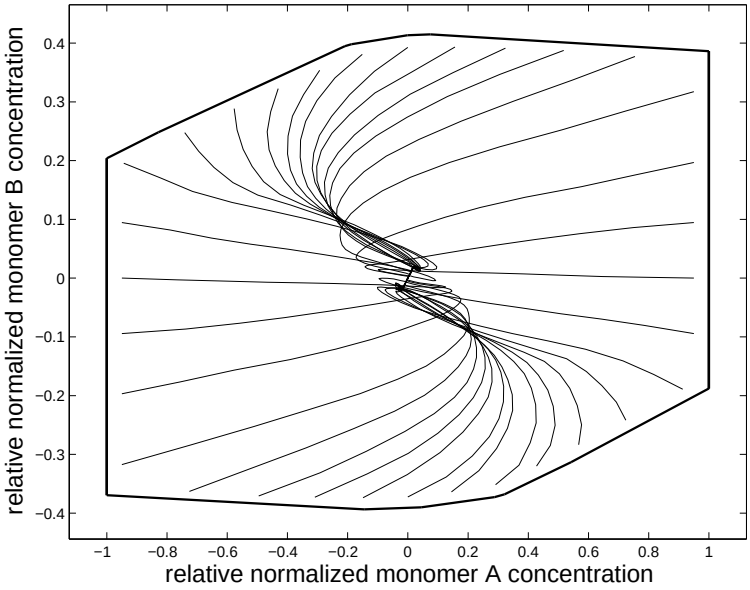


Figure 8.15: Trajectories for Algorithm A.1 [68] starting from the same initial conditions as the trajectories depicted in Figure 8.13.

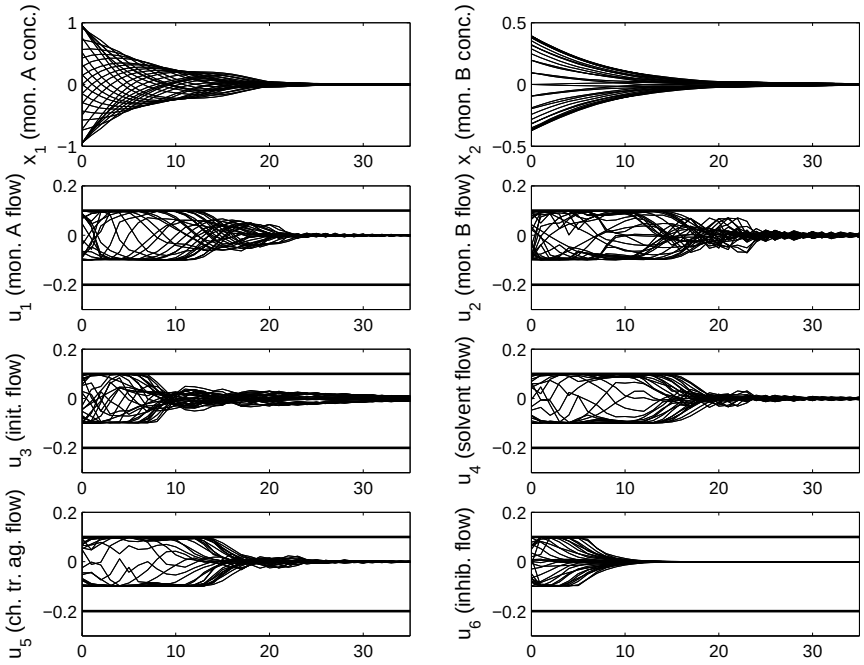


Figure 8.16: State and input sequence corresponding to the trajectories shown in Figure 8.15.

8.2.5 Simulation results

Figures 8.11-8.14 show trajectories starting from the edges of the respective feasible regions of the $\mathcal{E}$-RMPC [68] and $\mathcal{P}$-RMPC controllers with horizon length $N = 25$. It is hard to explicitly determine the feasible region of Algorithm A.1, so therefore Figures 8.15-8.16 shows simulation results for the same initial states as those used in Figures 8.13-8.14.

The following conclusions can be drawn:

- The feasible region of the $\mathcal{P}$-RMPC controller is significantly larger than the feasible region of $\mathcal{E}$-RMPC. This can be attributed to the lack of flexibility present in ellipsoidal sets compared to polyhedral sets, which becomes more apparent when the dimensionality increases. Furthermore, the fact that the feasible regions for $\mathcal{P}$-RMPC with $N = 20$ and $N = 25$ are nearly identical suggests that these feasible regions probably closely approximate the MCAS for the given system. This shows that $\mathcal{P}$-RMPC actually provides a method for *implicitely* computing the MCAS for a given system.

- Algorithm A.1 is feasible for all initial points close to the edge of the feasible region of the $\mathcal{P}$-RMPC controller. This indicates that in this example the feasible region of Algorithm A.1 is equal or larger than that of the $\mathcal{P}$-RMPC controller.

- $\mathcal{E}$-RMPC exhibits very conservative constraints handling. Even though many of the trajectories shown in Figure 8.11 start outside the invariant set of the local controller, none of the imposed constraints seem to become active for any of these trajectories. This clearly indicates that the trajectories obtained with $\mathcal{E}$-RMPC are suboptimal. Furthermore, since $\mathcal{E}$-RMPC and $\mathcal{P}$-RMPC make use of the same cost objective in their on-line optimization problems, this suboptimal behavior can entirely be attributed to the use of ellipsoidal invariant sets rather than polyhedral invariant sets. On the other hand, as shown in Figure 8.14, $\mathcal{P}$-RMPC exhibits non-conservative constraint handling and results in complex control behavior.

- Algorithm A.1 is significanlty less conservative than $\mathcal{E}$-RMPC. The applied input signals come much closer to the imposed input constraints. The asymmetry of these constraints, however, cannot be taken into account by Algorithm A.1. The convergence rate of the trajectories seems comparable to that of the $\mathcal{P}$-RMPC controller. The average (over all trajectories) total (over one trajectory) control cost was 9.63 for $\mathcal{P}$-RMPC and 10.08 for Algorithm A.1.

- The on-line computational cost of $\mathcal{P}$-RMPC was observed to be < 1 second per iteration, while the computational cost of $\mathcal{E}$-RMPC and that of Algorithm A.1 respectively were ~ 10 and ~ 20 seconds per iteration. This shows that also from a computational point of view $\mathcal{P}$-RMPC outperforms $\mathcal{E}$-RMPC in this case. However, all numbers are still well within the computational restrictions imposed by the sampling time of 30 minutes.

As a general conclusion it is not clear which algorithm is the best choice. $\mathcal{E}$-RMPC does not perform very well in this case, but $\mathcal{P}$-RMPC and Algorithm A.1 both seem to

have their advantages. $\mathcal{P}$-RMPC exhibits better (input) constraint handling and has a lower computational cost, but Algorithm A.1 has a larger feasible region and results in (slightly) more optimal control behavior.

Therefore, other factors are likely to influence the final choice. Two factors that seem to favor $\mathcal{P}$-RMPC is that QP solvers are generally speaking more robust (from a numerical point of view) than SDP solvers and that $\mathcal{P}$-RMPC has the potential of being extendable towards the inclusion of soft constraints.

On a final note, the rather restrictive choice of E that is made here, could be changed in order to give more degrees of freedom to the controller. This is possible because the computational complexity of the current controller is still well below the imposed specification.

8.3 Conclusions

In this section two case studies were investigated in order to illustrate the algorithms discussed in this thesis in a more practical setting. The first example consists of a looper-tension control loop of a hot steel mill, while the second example considers the control of a continuously stirred tank reactor for a copolymerization reaction. Both examples illustrate that the use of reduced-complexity polyhedral invariant sets leads to robust MPC controllers with enlarged operating regions, improved control performance and reduced computational complexity.

In both cases one can observe that existing methods for computing polyhedral invariant sets would not have been practically feasible due to the high dimensionality of the augmented systems encountered during the controller design. The second case study also clearly shows that the use of ellipsoidal invariant sets can lead to suboptimal control behavior and is therefore not advisable for high-dimensional systems.

Finally, the first case study shows that the construction of reduced-complexity control invariant sets is a harder problem than constructing reduced-complexity invariant sets. This issue hence needs further research.

Chapter 9

Conclusions and Future Research

"If we knew what it was we were doing, it would not be called research, would it?"

– Albert Einstein (1879-1955) –

9.1 Conclusions

General

In this thesis the author has investigated different mathematical techniques with the aim of obtaining robust MPC controllers with favorable scaling properties and non-conservative constraint handling.

The emphasis is put on the use of polyhedral constraint sets instead of ellipsoidal sets, since the latter offer little flexibility and result in expensive on-line optimization problems. Polyhedral invariant sets offer maximal flexibility but have the potential of leading to algorithms with exponential scaling behavior, eliminating the possibility of controlling large-scale systems or obtaining good control performance by using long prediction horizons.

In this thesis several methods have been described that are able to exploit this additional flexibility of polyhedral sets and result in robust MPC algorithms with improved constraint handling and improved scaling properties. These improvements were obtained using new results on two different levels: 1) on the level of the construction of invariant sets for use in MPC, and 2) on the level of algorithm synthesis in robust MPC. These two areas are hightlighted separately in the following sections.

Robust model based Predictive control

This thesis contains several algorithmic contributions to the area of robust MPC. However, all of these contributions should be considered in the light of the conceptual considerations discussed in Chapter 3. This chapter discussed the importance of incorporating the notion of feedback in the input sequence over which the on-line optimizations take place. It is shown that several robust MPC algorithms described in literature are incorrect due to the absence of this notion of feedback. This latter approach, called the closed-loop paradigm, also leads to improved control performance and, as is shown in Section 4.3, does not necessarily imply an increase of the number of on-line optimization variables. Chapter 5 goes one step further by showing that in the case of quasi-infinite horizon closed-loop MPC, reduced-complexity invariant sets can actually decrease the computational complexity, compared to open-loop robust MPC. This result is enabled by the fact that Algorithm 4.7 allows the constraints to be calculated as an invariant set of an augmented autonomous system. The following statement summarizes these observations.

> *As shown in this thesis the closed-loop MPC paradigm is of primary importance for recursive feasibility and stability of robust MPC algorithms. Contrary to common belief, the use of closed-loop predictions does not necessarily imply an increase in the on-line computational complexity but on the contrary, due to its specific structure, can lead to significant complexity reductions. The resulting algorithms can be shown to have linear scaling behavior instead of exponential scaling behavior observed in existing algorithms.*

Other algorithmic contributions are situated in several stages of the MPC design process. First of all Section 4.1 shows how polyhedral invariant sets can improve the synthesis of robust linear feedback controllers, that later can be used as a terminal or local controllers in robust MPC algorithms. Sections 4.2 and 4.3 then show how two existing MPC paradigms (quasi-infinite horizon MPC (RMPC) and interpolation based MPC (GIMPC), both of which are closed-loop paradigms) also benefit from the use of polyhedral invariant sets. On top of this GIMPC is further improved in Section 4.2.5 in order to improve constraint handling even more. The author would like to emphasize that the off-line synthesis of RMPC and GIMPC2 algorithms is almost identical and only consists of computing a positive invariant set and a quadratic Lyapunov function for a specially constructed augmented autonomous system. Both methods only differ in how this augmented system is constructed, and how the applied input depends on the augmented state.

> *The common structure present in the design process of RMPC and GIMPC2 directly suggests the existence of a more general MPC framework, where the class of candidate input sequences is parameterized by means of a linear autonomous system over whose state vector the on-line optimization actually takes place.*

This shift from FIR-like to IIR-like input sequence parameterizations would be the theoretical projection of the common practice in industry of employing piecewise

constant input sequence parameterizations (see e.g. [25]) with increasing interval lengths further down the control horizon.

The start of this paradigm shift can be traced back to [131] with the introduction of a terminal control law. The results in [123] further increased the importance of this local control law by also letting it also play a role in the within-horizon predictions. More recent contributions [29, 58, 61] further suggest the continuation of this trend.

A final contribution discussed in Chapter 7 is the extension of GIMPC towards non-linear controllers allowing interpolation between MPC controllers and controllers induced by control invariant sets. This allows the feasible region of any recursively feasible MPC algorithm to be extended to the largest region theoretically possible without a significant additional computational burden. The use of control invariant sets for ensuring constraint satisfaction in a tracking control setting is also briefly discussed, illustrating the potential of control invariant sets for a wider class of constrained control problems than positive invariant sets.

Set invariance

The main contribution of this thesis in the area of set invariance is the introduction of what might be called *"near-maximal"* polyhedral invariant sets. The two main other types of invariant sets within the class of polyhedral sets (maximal invariant sets [52] and low-complexity invariant sets [75]) each represent one extremum within the spectrum of trade-offs between maximal volume and minimal complexity. The class of near-maximal invariant sets allows the user to vary this trade-off between these two extremes. More specifically, it has been shown in this thesis that typically justifiably small volume reductions enable vast complexity reductions, with exponential scaling behavior that is reduced to linear scaling behavior in some cases.

The construction of near-maximal positive invariant sets is tackled in Chapter 5 by means of *pruning* and *trimming*, while Chapter 6 extends these techniques towards the construction of control invariant sets together with the addition of reduced-complexity polytope projections. An important aspect is that only maximality of the resulting sets is sacrificed, invariance of the resulting sets is still guaranteed and the resulting MPC algorithms can still be justified theoretically. This regularization-like method of constructing invariant sets is a novel approach and gives the user additional tuning parameters during the design process of robust MPC controllers. This can be summarized as follows.

> *The novel regularization-like approach towards the construction of polyhedral invariant sets allows the user to make a trade-off between maximality of the volume and minimality of the complexity of representation. The obtained complexity reductions enable the construction of polyhedral invariant sets for much higher-dimensional systems than previously possible, which in turn actually enables the construction of robust MPC algorithms with significantly* **larger** *feasible regions.*

Finally, we can conclude that the different directions for constructing reduced-complexity invariant sets that are explored in this thesis, should only be considered

initial steps into the unknown. Undoubtedly many other, better and more elegant solutions exist within the framework laid out in Chapters 5 and 6. Only in the last decade set invariance has been studied to the extent that is currently the case and many insights remain to be obtained. The following general statement summarizes these observations.

> *The framework for the construction of reduced-complexity (control) invariant sets laid out in Chapters 5 and 6 leaves room for many degrees of freedom, many of which are only partially explored in this thesis. Also, many properties of the algorithms presented in this thesis remain to be investigated in detail. Bringing in additional insights from computational geometry could potentially lead to vast improvements in these two areas.*

An example of this is the fact that the *combinatorial structure* of polytopes (defined by the *face lattice*) or the concept of *polar* polytopes [146] are not used in any of the algorithms presented in this thesis, while it is not unimaginable that considerations based on these concepts can lead to important new insights in set invariance theory.

9.2 Future research

Several potentially interesting future research topics can be identified in the area of robust model based predictive control as well as set invariance. In this section a few of the most interesting research directions are highlighted.

Robust model based predictive control

1. As already indicated in the conclusions, a trend towards the use of IRR-type parameterizations of the input sequence over which the MPC controller optimization can be observed. It also has become clear that two algorithms described in this thesis (RMPC and GIMPC2) fit in this framework, where the input sequence is governed by the dynamics of a linear autonomous system. A straightforward and potentially extremely interesting future research topic therefore is the development of a theoretical framework for such algorithms. This framework preferably would include robustness issues, results with respect to output-feedback, input-to-state stability, computation delays, etc ...

2. Chapter 7 already contains some preliminary results regarding tracking in order to illustrate the usefulness of control invariant sets for ensuring recursive feasibility in this setting. However, robust stability, guaranteed tracking performance and other important aspects have not been covered. A possible future subject is the development of a stability framework for tracking control problems using control invariant sets. Also potentially interesting is the combination of this framework with the research direction explained above.

Set invariance

1. One of the problems discussed in this thesis is the computation of the largest positive invariant set that lies within a given set of constraints (i.e. the construction of the MAS). Extensions towards the inclusion of bounded disturbances are mentioned briefly but not discussed in detail. A new concept arising in the presence of bounded disturbances is that of the smallest positive invariant set [114]. The size of a positive invariant set is hence upper bounded by the imposed constraint set and lower bounded by the bounded disturbances. These sets are useful for the construction of MPC algorithms (e.g. [72, 83]) for systems subject to bounded disturbances. However, when employing set duality in order to construct the polar sets of these three sets, the inclusions are inverted. This suggests many possible interesting properties and relations that can be exploited for more efficiently computing maximal and minimal admissible sets. Identifying these relations and formulating new algorithms for invariant set synthesis, exploiting these insights is one potentially interesting area of future research.

2. In this thesis the methods for constructing reduced-complexity invariant sets are further extended towards the construction of reduced-complexity control invariant sets. However, no results concerning convergence of the new algorithms are presented. Also, the expected scaling behavior of the new methods is still understood poorly. Further research in this area is necessary.

3. A third interesting future research direction is the extension of the existing results towards more general classes of systems, like hybrid systems and piecewise affine systems. Also the extension towards the gain scheduled control setting is potentially interesting. Algorithms for these settings already exist, but their scaling properties for higher dimensional systems is poor in general.

Appendices

Appendix A

Constrained Controller Synthesis for LPV Systems using LMIs

A.1 Introduction

This appendix summarizes the main results of [68]. The main results presented there consisted of a new robust MPC algorithm based on the recalculation at every sample instant of a linear robustly stabilizing control law with guaranteed constraint satisfaction by means of on-line solving an SDP. In order to guarantee constraint satisfaction, at every time instant an ellipsoidal feasible positive invariant set is calculated, corresponding to the linear control law. This latter aspect is of main importance in this appendix.

A.2 Problem formulation

We consider systems of the form (3.1)-(3.2) subject to constraints (2.10)-(2.11). The aim is to find a linear control law $u(k) = -Kx(k), k \in \mathbb{N}$ that is robustly asymptotically stabilizing

$$\lim_{k\to\infty} \max_{[A(i)\ B(i)]\in\Omega, i\in\mathbb{N}} \|x(k)\| = 0,$$

for a given initial state value $x(0) = \bar{x}$ and has guaranteed constraint satisfaction

$$x(k) \in \{x | x \in \mathcal{X}, -Kx \in \mathcal{U}\}, \qquad [A(i)\ B(i)] \in \Omega,\ i = 0, \ldots, k-1, \quad k \in \mathbb{N},$$

with $x(k+1) = (A(k) - B(k)K)x(k), k \in \mathbb{N}$. The aim is to minimize the following worst-case control objective:

$$J(\bar{x}) = \max_{[A(i)\ B(i)]\in\Omega, i\in\mathbb{N}} \left(\sum_{k=0}^{\infty} \|x(k)\|_Q^2 + \|u(k)\|_R^2 \right). \tag{A.1}$$

A.3 Solution based on convex optimization

In order to minimize (A.1) a quadratic upper bound $V(x) = x^{\mathrm{T}}Px \geq J(x)$ is considered. If the following condition is satisfied the function $V(x)$ is indeed a valid upper bound to the control objective:

$$x^{\mathrm{T}}Px - x^{\mathrm{T}}(A-BK)^{\mathrm{T}}P(A-BK)x \geq x^{\mathrm{T}}Qx + u^{\mathrm{T}}K^{\mathrm{T}}RKu, \quad \forall [A\ B] \in \Omega,$$

which in turn is satisfied if

$$P - (A-BK)^{\mathrm{T}}P(A-BK) \succ Q + K^{\mathrm{T}}RK, \qquad \forall [A\ B] \in \Omega,$$

which, due to convexity of Ω and the convexity of LMIs, is equivalent with

$$P - (A_j - B_jK)^{\mathrm{T}}P(A_j - B_jK) \succ Q + K^{\mathrm{T}}RK, \qquad j = 1, \ldots, r. \tag{A.2}$$

In order to guarantee constraint satisfaction, a level set $\mathcal{E}$ of $V(x)$

$$\mathcal{E} = \{x | V(x) \leq \gamma\},$$

which is automatically positive invariant, is chosen such that $\mathcal{E} \subseteq \{x | x \in \mathcal{X}, -Kx \in \mathcal{U}\}$. Constraint satisfaction follows automatically if $\bar{x} \in \mathcal{E}$, or equivalently if $\bar{x}^{\mathrm{T}}P\bar{x} \leq \gamma$. Optimization over γ, P, K subject to the above formulated constraints leads to a robustly stabilizing control law and a corresponding ellipsoidal invariant set. In order to perform this optimization the following change of variables $Z = \gamma P^{-1}, Y = -KZ$ is performed, leading to the following reformulation of (A.2):

$$\gamma Z^{-1} - (A_j - B_jYZ^{-1})^{\mathrm{T}}\gamma Z^{-1}(A_j - B_jYZ^{-1}) \succ Q + (YZ^{-1})^{\mathrm{T}}RYZ^{-1}, \qquad j = 1, \ldots, r.$$

Pre- and post-multiplication with Z and division by γ yields

$$Z \succ (A_jZ - B_jY)^{\mathrm{T}}Z^{-1}(A_jZ - B_jY) + \gamma^{-1}Q + \gamma^{-1}Y^{\mathrm{T}}RY, \quad j = 1, \ldots, r,$$

which can be formulated as an LMI using the Schur complement. This leads to the following algorithm [68, 93]:

Algorithm A.1 (Constrained controller synthesis for LPV systems, [68]). *Given a system* (3.1)-(3.2) *subject to constraints* (2.10)-(2.11)*, an optimality criterion* (A.1) *and an initial state* $\bar{x} \in \mathbb{R}^{n_x}$*, solve the following optimization problem:*

$$\min_{\gamma\in\mathbb{R}, Y\in\mathbb{R}^{n_u\times n_x}, Z\in\mathbb{S}_{++}^{n_x}} \gamma, \tag{A.3a}$$

subject to

$$\begin{bmatrix} 1 & * \\ \bar{x} & Z \end{bmatrix} \succ 0, \tag{A.3b}$$

$$\begin{bmatrix} Z & * & * & * \\ Q^{\frac{1}{2}}Z & \gamma I & * & * \\ R^{\frac{1}{2}}Y & 0 & \gamma I & * \\ A_jZ + B_jY & 0 & 0 & Z \end{bmatrix} \succ 0, \quad j = 1, \ldots, l \tag{A.3c}$$

$$\begin{bmatrix} Z & * \\ A_u(j,:)Y & 1 \end{bmatrix} \succ 0, \quad j = 1, \ldots, m_u, \tag{A.3d}$$

$$\begin{bmatrix} Z & * \\ A_x(j,:)Z & 1 \end{bmatrix} \succ 0, \quad j = 1, \ldots, m_x. \tag{A.3e}$$

Asterisks are used to denote the corresponding transpose of the lower block part of symmetric matrices. This avoids redundant notation, since LMIs always must consist of symmetric matrix expressions. The optimal solutions to this optimization problem are denoted as $\gamma^{\mathrm{o}}, Y^{\mathrm{o}}, Z^{\mathrm{o}}$. The feedback matrix $u(k) = -Kx(k)$, the closed-loop Lyapunov function $V(x) = x^{\mathrm{T}}Px$ and an invariant ellipsoid $\mathcal{E}$ are computed as

$$K = -Y^{\mathrm{o}}(Z^{\mathrm{o}})^{-1}, \tag{A.3f}$$

$$P = \gamma^{\mathrm{o}}(Z^{\mathrm{o}})^{-1}, \tag{A.3g}$$

$$\mathcal{E} = \{x | x^{\mathrm{T}}(Z^{\mathrm{o}})^{-1}x \leq 1\}. \tag{A.3h}$$

The following lemma is given without proof. For the proof we refer to [68].

Lemma A.1. *If* (A.3) *is feasible, then the following properties hold:*

- *The controller $u(k) = -Kx(k)$ robustly stabilizes system* (3.1)-(3.2)*.*
- *The set $\mathcal{E}$ is robustly positive invariant with respect to the closed-loop dynamics.*
- *The set $\mathcal{E}$ is feasible with respect to the state constraint set $\mathcal{X}' = \{x | x \in \mathcal{X}, -Kx \in \mathcal{U}\}$.*
- *The initial state $\bar{x}$ lies inside $\mathcal{E}$.*
- *For all initial states $x \in \mathcal{E}$, the function $V(x) = x^{\mathrm{T}}Px$ is an upper bound to the worst case value of the cost* (1.3) *(with $x_{\mathrm{ref}}(k) \equiv 0, u_{\mathrm{ref}}(k) \equiv 0$) over all possible uncertainty realizations $[A(k)\ B(k)] \in \Omega, k \in \mathbb{N}$.*

If the system is quadratically stabilisable and $\bar{x}$ is sufficiently close to the origin, the optimization problem (A.3) is always feasible. The quadratic stabilisability requirement is relaxed in [41].

One can verify that the control law, the corresponding invariant set and quadratic cost function satisfy conditions (3.11) and can hence be used as terminal controller, constraint and cost in (robust) MPC algorithms, as is e.g. done in [3, 96, 103, 104, 140–142].

Apart from the use as an off-line tool to calculate robust feedback laws and corresponding invariant sets, the above algorithm can be applied on-line in order to obtain a robust MPC algorithm. In [68] recursive feasibility and robust asymptotic stability of such a setup are proven.

A.4 Convex Combinations

An interesting additional advantage of the fact that Algorithm A.1 is based on solving a single convex optimization problem, is the fact that different solutions to (A.3) can be combined by making convex combinations. Assume (A.3) is solved for n different initial states $\bar{x}_1, \ldots, \bar{x}_n$, resulting in optimal solutions $\gamma_i, Y_i, Z_i, i = 1, \ldots, n$, then the following solutions

$$\gamma = \sum_{i=1}^{n} \lambda_i \gamma_i, \qquad Y = \sum_{i=1}^{n} \lambda_i Y_i, \qquad Z = \sum_{i=1}^{n} \lambda_i Z_i,$$

are feasible solutions to (A.3) for initial state $\bar{x} = \sum_{i=1}^{n} \lambda_i \bar{x}_i$ if $\sum_{i=1}^{n} \lambda_i = 1, \lambda_{1\ldots n} \geq 0$. These solutions then induce a corresponding robustly stabilizing (but not necessarily optimal) control law with corresponding invariant set. This property was used in [96, 97, 103–105, 142, 143] to construct MPC algorithms with enlarged feasible region and reduced on-line computational complexity.

Appendix B

Projecting Polytopes using Fourier-Motzkin Elimination

B.1 Problem Formulation

In this section we consider an n-dimensional H-polytope $\mathcal{P}$, defined as the intersection of m halfspaces :

$$\mathcal{P} \triangleq \{x \in \mathbb{R}^n | Ax \leq b\}, \tag{B.1}$$

with $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$. In the case of the polytope depicted in Figure B.1, we have e.g. $n = 2$ and $m = 5$. The problem considered here is that of finding a H-polytope description of the orthogonal projection of $\mathcal{P}$ onto $\mathbb{R}^{n-1}$.

Problem B.1. *Given a polytope $\mathcal{P}$ described as* (B.1)*, compute a polytope $\mathcal{P}' \triangleq \{x \in \mathbb{R}^{n-1} | A'x \leq b'\}$ such that*

$$\mathcal{P}' = \text{proj}(\mathcal{P}). \tag{B.2}$$

In the next section we describe how Problem B.1 can be solved by means of Fourier-Motzkin elimination [146]. A different method would be Equality Set Projection (ESP, [62]), but this method does not lend itself well to the extensions proposed in Section 6.2.4. Both methods are included in the MPT toolbox [86] for MATLAB.

B.2 Fourier-Motzkin Elimination

Fourier-Motzkin can be seen as the equivalent of Gauss-Jordan elimination for sets of inequality constraints. Gauss-Jordan elimination aims to find the strictest set of linear equalities in $x_1, \ldots, x_{n-1}$ that forms necessary conditions for satisfaction of the complete system of equality constraints, while Fourier-Motzkin does exactly the same for sets of inequality constraints. While Gauss-Jordan elimination constructs this set as

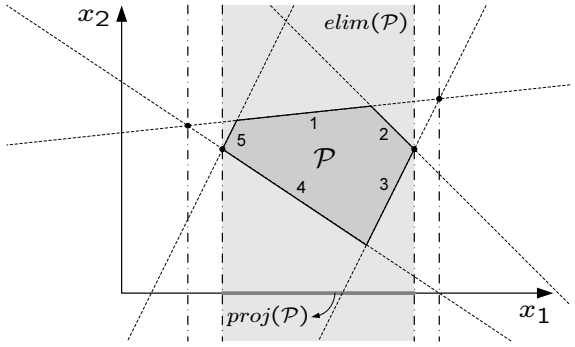


Figure B.1: Geometric interpretation of the Fourier-Motzkin algorithm (Algorithm B.1). Dash-dotted lines depict 4 out of 6 candidate constraints generated by Fourier-Motzkin elimination.

linear combinations of the existing constraints, Fourier-Motzkin elimination constructs this set as *convex* combinations of the existing constraints.

Before formulating the algorithm, some notation has to be introduced. We use shorthand notation to denote elements and rows of A and b, with $a_{ij} \triangleq A(i,j), a_i \triangleq A(i,:)$ and $b_i \triangleq b(i)$.

Algorithm B.1 (Fourier-Motzkin Elimination, [146]**).** *Given a polytope $\mathcal{P}$ defined as* (B.1)*, construct $\overline{A}, \overline{b}$ as follows :*

- *Initialize $\overline{A} := [a_{i_1}; \ldots; a_{i_{m_0}}], \overline{b} := [b_{i_1}; \ldots; b_{i_{m_0}}]$, where $i_j, j = 1, \ldots, m_0$ are all indices such that $a_{i_j n} = 0$.*
- *For every index $i_1 \in [1,m], i_2 \in [1,m]$ such that $a_{i_1 n} > 0$ and $a_{i_2 n} < 0$ set*

$$\overline{A} := \left[\overline{A}; \frac{-a_{i_2 n}}{a_{i_1 n} - a_{i_2 n}} a_{i_1} + \frac{a_{i_1 n}}{a_{i_1 n} - a_{i_2 n}} a_{i_2}\right], \tag{B.3}$$

$$\overline{b} := \left[\overline{b}; \frac{-a_{i_2 n}}{a_{i_1 n} - a_{i_2 n}} b_{i_1} + \frac{a_{i_1 n}}{a_{i_1 n} - a_{i_2 n}} b_{i_2}\right]. \tag{B.4}$$

- *Remove all redundant constraints from $\overline{A}x \leq \overline{b}$.*

In Figure B.1, we have $a_{in} > 0$ for $i = 1, 2, 5$ and $a_{in} < 0$ for $i = 3, 4$. The constraints resulting from $(i_1, i_2) = (2, 4)$ and $(i_1, i_2) = (5, 3)$ are not depicted due to space constraints.

It can be verified algebraically that all constraints in A' are independent of x_n. Furthermore, since all constraints are obtained as convex combinations of the original constraints, the original set $\mathcal{P}$ is a subset of the set defined by $\overline{A}, \overline{B}$. Since all possible convex combinations are added to $\overline{A}, \overline{b}$ it can intuitively be seen that the resulting set $\text{elim}(\mathcal{P}) = \{x | \overline{A}x \leq \overline{b}\}$. The projection can hence be found as $A' = \overline{A}(:, 1 : n-1), b' = \overline{b}$.

It should be noted that in the worst case, the number of constraints in A', b' is equal to $\lfloor \frac{m^2}{4} \rfloor$. Computing k-dimensional projections can in the worst case lead $\lfloor \frac{m}{2} \rfloor^{2^k}$ constraints, with an accordingly high computational cost.

Another disturbing observation is that often one sees that, when projecting across multiple dimensions, the sets at intermediate dimensions have the largest number of non-redundant constraints[1], which indicates that even having an upper bound on the number of constraints describing the final projection (e.g., based on knowledge about the problem structure) does not guarantee that this projection can be computed efficiently.

[1]This increased number of constraints at intermediate dimensions is sometimes referred to as *the bulge*.

Appendix C

Joint Spectral Radius

The Joint Spectral Radius (JSR) is a mathematical concept that has gained interest in recent years, because of its usefulness in a variety of applications, among which the stability analysis of (uncertain) linear autonomous dynamical systems is of the most importance in this thesis. We first define the spectral radius of a matrix and then define the joint spectral radius of a set of matrices.

Definition C.1 (Spectral Radius). *Given a matrix norm $\|\cdot\|$, the spectral radius of a matrix $A \in \mathbb{R}^{n\times n}$ is defined as*

$$\rho(A) \triangleq \lim_{k\to+\infty} \|A^k\|^{\frac{1}{k}}. \tag{C.1}$$

The spectral radius of a matrix can be interpreted as the asymptotic growth rate of the series A^k for increasing values of k. The value of the spectral radius is independent of the matrix norm that is used (see [57]) and can be shown to be equal to:

$$\rho(A) = \{|\lambda| : \lambda \text{ is an eigenvalue of } A\}. \tag{C.2}$$

This equivalence already shows that if the spectral radius of a matrix A is smaller than 1, the autonomous systems $x(k+1) = Ax(k)$ is stable.

The joint spectral radius is an extension of the spectral radius to multiple matrices:

Definition C.2 (Joint Spectral Radius (JSR, [126])). *Given a matrix norm $\|\cdot\|$, the joint spectral radius of a set of matrices $\mathcal{M} \triangleq \{A_1, \ldots, A_r\}$ is defined as*

$$\hat{\rho}(\mathcal{M}) \triangleq \limsup_{k\to+\infty} \max_{A(1\ldots k)\in\mathcal{M}} \|A(1)\cdot\ldots\cdot A(k)\|^{\frac{1}{k}}. \tag{C.3}$$

It should be noted that the value of the JSR does not change if also all convex combinations of the matrices A_i are considered when taking the maximum, i.e. if we define $\mathcal{M} = \mathrm{Co}\{A_1, \ldots, A_r\}$.

Similar to the spectral radius, the JSR can be used to check the stability of linear uncertain or linear time-varying systems. The condition $\hat{\rho}(\mathcal{M}) < 1$ is a sufficient and necessary condition for asymptotic stability of $x(k+1) = A(k)x(k),\ A(k) \in \mathcal{M}$,

which makes the JSR a powerful tool for stability analysis of such autonomous LPV systems. However, the computation of the JSR is an NP-hard problem. However, some methods for approximately computing the JSR have been proposed recently [18]. The following bounds on the JSR are proven in [71]:

$$\max_{\sigma\in\{1,\dots,m\}^k} \rho(A_\sigma)^{\frac{1}{k}} \leq \hat{\rho}(\{A_1,\dots,A_m\}) \leq \max_{\sigma\in\{1,\dots,m\}^k} \|A_\sigma\|^{\frac{1}{k}}, \qquad \text{(C.4)}$$

where A_σ denotes $A_{\sigma_1}A_{\sigma_2}\dots A_{\sigma_k}$. Throughout this thesis the above expression with $k = 15$ is used to compute upper bounds to the JSR. See [18, 55] for more computational results.

Appendix D

Proofs

D.1 Proof of Lemma 5.3

Proof: We consider the invocation of Lemma 5.2 during an arbitrary iteration i of Algorithm 5.2 when applied to (4.63)-(4.64). The constraint tree of the intermediate set $\bigcap_i \mathcal{X}_i$ is depicted in Figure D.1. This proof is restricted to the case $r = 2$ and $E = I$, but similar derivations can be made for $r > 2$ and $E \neq I$.

We now consider the application of Lemma 5.2 for tightening constraint $a_1^{\mathrm{T}} x \leq 1$ in order to make constraint $a_2^{\mathrm{T}} x \leq 1$ redundant, both of which are children of constraint $b^{\mathrm{T}} x \leq 1$ as depicted in Figure D.1. In order to calculate the necessary tightening factor, optimization problem (5.8) is solved. The matrix A of (5.8) represents all constraints of the set under consideration (Figure D.1) except the two constraints defined by a_1 and a_2. In order to obtain an upper bound on the tightening factor η, we construct a lower bound on the optimal solution ξ^{o} of (5.8). In order to do so, we first rewrite (5.8) in the following equivalent form:

$$\min_x \; a_1^{\mathrm{T}} x, \quad \text{s.t.} \quad \begin{cases} Ax \leq \mathbf{1}, \\ a_2^{\mathrm{T}} x = 1. \end{cases} \tag{D.1}$$

We now calculate the Lagrange dual optimization problem (see [24, Chapter 5]) of this LP:

$$\begin{aligned} \max_{\lambda,\nu} \quad & -\mathbf{1}^{\mathrm{T}} \lambda - \nu, \\ \text{s.t.} \quad & \lambda \geq 0, \\ & a_1 + A^{\mathrm{T}} \lambda + \nu a_2 = 0. \end{aligned} \tag{D.2}$$

It is known [24] that any feasible solution to the Lagrange dual problem is a lower bound to the optimal solution of the original (primal) problem. We now construct a feasible solution $(\lambda^{\mathrm{f}}, \nu^{\mathrm{f}})$ to (D.2) in order to obtain such a lower bound. First we construct a feasible solution to the equality constraints of (D.2), after which we make sure the solution also satisfies $\lambda^{\mathrm{f}} \geq 0$. We first choose $\nu^{\mathrm{f}} = -1$, after which we need to choose λ^{f} such that

$$A^{\mathrm{T}} \lambda^{\mathrm{f}} = a_2 - a_1. \tag{D.3}$$

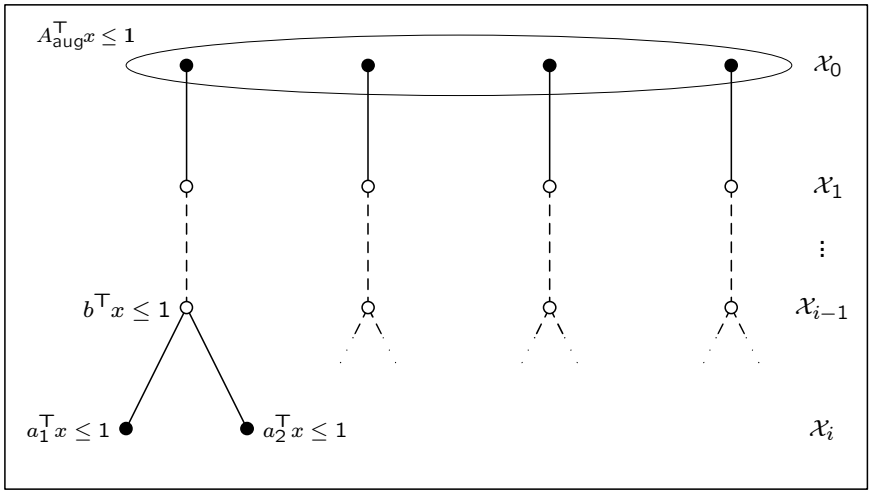


Figure D.1: Constraint tree under consideration in Lemma 5.3. Circles represent individual constraints. Solid circles represent those constraints that are explicitly taken into consideration in the derivation of the upper bound on the tightening factor η.

By making use of the structure present in (4.63), and the fact that $a_1 = \Psi_{\text{aug},1}^{\text{T}} b$ and $a_2 = \Psi_{\text{aug},2}^{\text{T}} b$, we can rewrite (D.3) as

$$A^{\text{T}} \lambda^{\text{f}} = \begin{bmatrix} (\Phi_2 - \Phi_1)^{\text{T}} \\ (B_2 - B_1)^{\text{T}} \\ 0_{(N-1)\cdot n_u \times 1} \end{bmatrix} \overline{b}, \tag{D.4}$$

with $b = [\overline{b};\ \underline{b}], \overline{b} \in \mathbb{R}^{n_x}, \underline{b} \in \mathbb{R}^{N \cdot n_u}$ and $\Phi_i = a_i - B_i K$. This shows that, due to the structure of (4.63), the vectors a_1 and a_2 only differ in their first $n_x + n_u$ components, which gives additional degrees of freedom in how to choose λ^{f}. We choose to set those components of λ^{f} to 0 that correspond to constraints of $\mathcal{X}_j, j > 0$. This is justified, since the constraints of $\mathcal{X}_0 \equiv \mathcal{X}_{\text{aug}}$ form a closed set in $\mathbb{R}^{n_x+n_u} \times \{0\}^{(N-1).n_u}$ and therefore the rows of A_{aug} form a basis for this space, which is also exactly the space in which the right-hand side of (D.4) lies. By denoting the non-zero components of λ as λ', we obtain the simplified system of equalities:

$$\underbrace{\begin{bmatrix} A_x^{\text{T}} & -K^{\text{T}} A_u^{\text{T}} \\ 0 & A_u^{\text{T}} \end{bmatrix}}_{C} \lambda'^{\text{f}} = \underbrace{\begin{bmatrix} (\Phi_2 - \Phi_1)^{\text{T}} \\ (B_2 - B_1)^{\text{T}} \end{bmatrix}}_{U} \overline{b}. \tag{D.5}$$

We now construct a non-negative solution $\lambda'^{\text{f}} \triangleq \lambda^{(+-)} + \epsilon \lambda^{(+)}$, with

$$\lambda^{(+-)} \triangleq C^{\dagger} U \overline{b}, \tag{D.6}$$

a vector that can have both positive and negative components and $\lambda^{(+)}$ a strictly positive homogeneous solution to (D.5). Since C has full row rank (for reasons given above) $\lambda^{(+-)}$ is guaranteed [9] to be an exact solution to (D.5).

The existence of a strictly positive homogeneous solution $\lambda^{(+)}$ to (D.5) is guaranteed due to the fact that $\mathcal{X}_{\text{aug}}$ is a strictly bounded polytope inside its affine hull, and therefore 0 lies strictly inside the relative interior of the dual (polar) polytope (see [146, Section 2.3]). This latter observation guarantees the existence of a strictly positive homogeneous solution $\lambda^{(+)}$ to (D.5). ϵ can now be chosen large enough, such that $\lambda'^{\text{f}} \geq 0$, while satisfying (D.5). It is clear that if we choose ϵ as follows:

$$\epsilon = \frac{\|\lambda^{(+-)}\|}{\min \lambda^{(+)}}, \tag{D.7}$$

where the minimum is taken over all components, the resulting λ'^{f} effectively satisfies $\lambda'^{\text{f}} \geq 0$. The corresponding variable λ^{f} is now constructed by inserting the appropriate number of zeros. We now compute (a lower bound to) the corresponding value of the objective function of (D.2). First we compute an upper bound to $\|\lambda'^{\text{f}}\|$:

$$\begin{aligned}
\|\lambda'^{\text{f}}\| &= \left\| \lambda^{(+-)} + \frac{\|\lambda^{(+-)}\|}{\min \lambda^{(+)}} \lambda^{(+)} \right\|, \\
&\leq \left(1 + \frac{\|\lambda^{(+)}\|}{\min \lambda^{(+)}} \right) \|\lambda^{(+-)}\|, \\
&\leq \left(1 + \frac{\|\lambda^{(+)}\|}{\min \lambda^{(+)}} \right) \|C^{\dagger}\| \|U\| \|\overline{b}\|.
\end{aligned} \tag{D.8}$$

The first two factors of the right-hand side depend solely on the constraints A_{aug} imposed on the augmented system. while the third factor only depends on the amount of uncertainty present in the system to be controlled. Only the last factor depends on the (first n_x components of the) specific constraint whose children are considered in Lemma 5.2. It is now possible to construct a lower bound on the objective function of (D.2):

$$\begin{aligned}
-\mathbf{1}^{\text{T}} \lambda^{\text{f}} - \nu^{\text{f}} &= 1 - \mathbf{1}^{\text{T}} \lambda'^{\text{f}}, \\
&\geq 1 - \sqrt{m} \|\lambda'^{\text{f}}\|, \\
&\geq 1 - \sqrt{m} \left(1 + \frac{\|\lambda^{(+)}\|}{\min \lambda^{(+)}} \right) \|C^{\dagger}\| \|U\| \|\overline{b}\|,
\end{aligned} \tag{D.9}$$

where m is the number of rows in C. Because the left-hand side of (D.9) is a lower bound to the optimal objective value of (D.1) and hence to the (identical) optimal objective value ξ^{o} of (5.8), we have

$$\begin{aligned}
\xi^{\text{o}} &\geq -\mathbf{1}^{\text{T}} \lambda^{\text{f}} - \nu^{\text{f}}, \\
&\geq 1 - \sqrt{m} \left(1 + \frac{\|\lambda^{(+)}\|}{\min \lambda^{(+)}} \right) \|C^{\dagger}\| \|U\| \|\overline{b}\|,
\end{aligned} \tag{D.10}$$

and hence, if $\sqrt{m} \left(1 + \frac{\|\lambda^{(+)}\|}{\min \lambda^{(+)}} \right) \|C^{\dagger}\| \|U\| \|\overline{b}\| < 1$, we have

$$\eta \leq \frac{1}{1 - \sqrt{m} \left(1 + \frac{\|\lambda^{(+)}\|}{\min \lambda^{(+)}} \right) \|C^{\dagger}\| \|U\| \|\overline{b}\|}. \tag{D.11}$$

This concludes the proof. □

D.2 Proof of Theorem 5.5

We first prove that under the conditions of this theorem, all tightening attempts of Algorithm 5.2 are successful. Due to Corollary 5.2, which implies that the depth of the constraint tree is upper bounded by a linear function of N, we can then conclude that the number of constraints describing the resulting invariant set is $O(N)$.

In order to show that all tightening attempts are successful, it has to be proven that both conditions of step 2(b) of Algorithm 5.2 are satisfied. It is straightforward to see that the first condition is satisfied due to the lower bound imposed on γ by this theorem, while the second condition is satisfied due to the upper bound imposed on γ by this theorem.

As a result no branching ever occurs in the constraint tree. The maximal depth of the constraint tree can be computed to be

$$k^{\star}_{\max} = N - 1 + \left\lfloor -\frac{\ln a + \ln b + \ln c}{\ln d_1} \right\rfloor, \tag{D.12}$$

using similar arguments as those used in Corollary 2.1. Since d_1 can be chosen independent of N, it is now shown that the total number of constraints describing the invariant set increase as a linear function of N. □

Bibliography

[1] F. Alizadeh and D. Goldfarb. Second-order cone programming. *Mathematical Programming Series B*, (95):3–51, 2003.

[2] J. C. Allwright and G. C. Papavasiliou. On linear programming and robust model-predictive control using impulse-responses. *Systems and control letters*, 18:159–164, 1992.

[3] M. Bacic, M. Cannon, Y. I. Lee, and B. Kouvaritakis. General interpolation in MPC and its advantages. *IEEE Transactions on Automatic Control*, 48(6):1092–1096, June 2003.

[4] A. Bemporad, F. Borrelli, and M. Morari. Model predictive control based on linear programming - the explicit solution. *IEEE Transactions on Automatic Control*, 47(12):1974–1985, 2002.

[5] A. Bemporad, F. Borrelli, and M. Morari. Min-max control of constrained uncertain discrete-time linear systems. *IEEE Transactions on Automatic Control*, 48(9):1600–1606, 2003.

[6] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit solution of model predictive control via multiparametric quadratic programming. In *Proceedings of the American Control Conference*, 2000.

[7] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

[8] A. Bemporad and A. Mosca. Fulfilling hard constraints in uncertain linear systems by reference managing. *Automatica*, 34(4):451–461, 1998.

[9] S. Bernstein. *Matrix Mathematics*. Princeton University Press, 2005.

[10] R. R. Bitmead, M. Gevers, and V. Wertz. *Adaptive Optimal Control - The Thinking Man's GPC*. Englewood Cliffs, NJ: Prentice Hall, 1990.

[11] G. Bitsoris. On the positive invariance of polyhedral sets for discrete-time systems. *Systems and Control Letters*, 11:243–248, 1988.

[12] G. Bitsoris. Positively invariant polyhedral sets of discrete-time linear systems. *International Journal of Control*, 47(6):1713–1726, 1988.

[13] F. Blanchini. Ultimate boundedness control for uncertain discrete-time systems via set-induced lyapunov functions. *IEEE Transactions on Automatic Control*, 39(2):428–433, 1994.

[14] F. Blanchini. Set invariance in control – a survey. *Automatica*, 35:1747–1767, 1999.

[15] F. Blanchini, S. Miani, and C. Savorgnan. Polyhedral Lyapunov functions computation for robust and gain scheduled design. In *Proceedings of the Symposium on nonlinear Control Systems (NOLCOS), Stuttgart, Germany*, 2004.

[16] F. Blanchini, S. Miani, and M. Sznaier. Robust performance with fixed and worst-case signals for uncertain time-varying systems. *Automatica*, 33(12):2183–2189, 1997.

[17] H. H. J. Bloemen, T. J. J. van den Boom, and H. B. Verbruggen. Optimizing the end-point state-weighting matrix in model-based predictive control. *Automatica*, 38:1061–1068, 2002.

[18] V. D. Blondel and Y. Nesterov. Computationally efficient approximations of the joint spectral radius. *SIAM Journal of Matrix Analysis*, 27(1):256–272, 2005.

[19] V. D. Blondel, Y. Nesterov, and J. Theys. On the accuracy of the ellipsoid norm approximation of the joint spectral radius. *Linear Algebra and its Applicatons*, (394):91–107, 2005.

[20] V. D. Blondel and J. N. Tsitsiklis. The boundedness of all products of a pair of matrices is undecidable. *Systems and Control Letters*, 41:135–140, 2000.

[21] F. Borrelli, A. Bemporad, and M. Morari. Geometric algorithm for multiparametric linear programming. *Journal of Optimization Theory and Applications*, 118(3):515–540, 2003.

[22] S. Boyd and C. H. Barratt. *Linear Controller Design – Limits of Performance*. Prentice Hall, 1991.

[23] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*. SIAM Publications, 1994.

[24] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[25] R. Cagienard, P. Grieder, E. Kerrigan, and M. Morari. Move blocking strategies in receding horizon control. *Provisionally accepted for Journal of Process Control*, 2006.

[26] E.F. Camacho and C. Bordóns. *Model Predictive Control in the Process Industry*. Springer-Verlag, 1995.

[27] P. J. Campo and M. Morari. Robust model predictive control. In *Proceedings of the American Control Conference*, pages 1021–1026, 1987.

[28] M. Cannon, V. Deshmukh, and B. Kouvaritakis. Nonlinear model predictive control with polytopic invariant sets. *Automatica*, 39:487–494, 2003.

[29] M. Cannon and B. Kouvaritakis. Optimizing prediction dynamics for robust MPC. *IEEE Transactions on Automatic Control*, 50(11):1892–1897, 2005.

[30] M. Cannon, B. Kouvaritakis, and V. Deshmukh. Enlargement of polytopic terminal region in nmpc by interpolation and partial invariance. *Automatica*, 40:311–317, 2004.

[31] A. Casavola, M. Gianelli, and E. Mosca. Min-max predictive control strategies for input-saturated polytopic uncertain systems. *Automatica*, 36:125–133, 2000.

[32] H. Chen and F. Allgöwer. A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica*, 34:1205–1217, 1998.

[33] H. Chen, C. Scherer, and F. Allgöwer. A game theoretic approach to nonlinear robust receding horizon control of constrained systems. In *Proceedings of the American Control Conference*, 1997.

[34] L. Chisci, P. Falugi, and G. Zappa. Predictive control for constrained LPV systems. In *Proceedings of the European Control Conference, Porto, Portugal*, pages 3074–3079, 2001.

[35] L. Chisci and G. Zappa. Robust predictive regulation with invariance constraint. *European Journal of Control*, 8(1):2–14, 2002.

[36] I. S. Choi, J. A. Rossiter, Fleming P. Pluymers, B., and B. De Moor. A robust MPC design for hot rolling mills: a polyhedral invariant set approach. In *Proceedings of the American Control Conference, Minneapolis, USA*, 2005.

[37] J. P. Congalidis, J. R. Richards, and W. H. Ray. Modeling and control of a copolymerization reactor. In *Proceedings of the American Control Conference*, volume 3, pages 1779–1793, 1986.

[38] J.P. Congalidis, J.R. Richards, and W.H. Ray. Feedforward and feedback controler of a solution copolymerization reactor. *AIChE Journal*, 35(6), 1989.

[39] M. Corless. Stabilization of uncertain discrete time systems. In *Proceedings of the IFAC Workshop on Model Error Concepts Compensation, Boston*, 1985.

[40] M. Corless and J. Manela. Control of uncertain discrete-time systems. In *Proceedings of the American Control Conference, Seattle*, 1986.

[41] F. A. Cuzzola, J. C. Geromel, and M. Morari. An improved approach for constrained robust model predictive control. *Automatica*, 38:1183–1189, 2002.

[42] G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1963.

[43] G. De Nicolao, L. Magni, and R. Scattolini. On the robustness of receding horizon control with terminal constraints. *IEEE Transactions on Automatic Control*, 41:451–453, 1996.

[44] G. De Nicolao, L. Magni, and R. Scattolini. Robust predictive control of systems with uncertain impulse response. *Automatica*, 32(10):1475–1479, 1996.

[45] E. De Santis. On positively invariant sets for discrete-time linear systems with disturbance: An application of maximal disturbance sets. *IEEE Transactions on Automatic Control*, 39(1):245–249, 1994.

[46] C. E. T. Dorea and J. C. Hennet. $(a,\ b)$-invariant polyhedral sets of linear discrete-time systems. *Journal of Optimization Theory and Applications*, 103(3):521–542, 1999.

[47] J. C. Doyle, B. A. Francis, and A. R. Tannenbaum. *Feedback Control Theory*. Macmillan Coll. Div., 1992.

[48] R. Findeisen, L. Imsland, F. Allgöwer, and B. A. Foss. State and output feedback nonlinear model predictive control: An overview. *European Journal of Control*, 9(2-3):190–207, 2003.

[49] G. Francis, J. D. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems (5th Edition)*. Prentice Hall, 2005.

[50] H. Genceli and M. Nikolaou. Robust stability analysis of constrained L1 norm model predictive control. *AIChE Journal*, 39(12):1954–1965, 1993.

[51] E. G. Gilbert and I. Kolmanovsky. Fast reference governors for systems with state and control constraints and disturbance inputs. *International Journal of Robust and Nonlinear Control*, 9:1117–1141, 1999.

[52] E. G. Gilbert and K. T. Tan. Linear systems with state and control constraints: The theory and application of maximal output admissible sets. *IEEE Transactions on Automatic Control*, 36(9):1008–1020, 1991.

[53] J. E. Goodman and J. O'Rourke, editors. *Handbook of discrete and computational geometry, Second Edition*. CRC Press, 2004.

[54] P. Goulart, E. Kerrigan, and J. Maciejowski. Optimization over state feedback policies for robust control with constraints. *Automatica*, 42(4):523–533, 2006.

[55] G. Gripenberg. Computing the joint spectral radius. *Linear Algebra and its Applicatons*, (234):43–60, 1996.

[56] P.O. Gutman and P. Hagandar. A new design of constrained controller for linear systems. *IEEE Transactions on Automatic Control*, 30:22–33, 1985.

[57] R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, UK, 1991.

[58] L. Imsland, N. Bar, and B. Foss. More efficient predictive control. *Automatica*, 41(8):1395–1403, 2005.

[59] L. Imsland, R. Findeisen, F. Allgöwer, and B. A. Foss. Output feedback stabilization with nonlinear predictive control: Asymptotic properties. *Journal of Modeling, Identification and Control*, 24(3):169–179, 2003.

[60] L. Imsland, R. Findeisen, E. Bullinger, and F. Allgöwer. A note on stability, robustness and performance of output feedback nonlinear model predictive control. *Journal of Process Control*, 13(7):633–644, 2003.

[61] L. Imsland, J. A. Rossiter, B. Pluymers, and J. Suykens. Robust triple mode MPC. In *Proceedings of the American Control Conference 2006, Minneapolis, USA*, pages CD–ROM, 2006.

[62] C. N. Jones, E. C. Kerrigan, and J. M. Maciejowski. Equality set projection: A new algorithm for the projection of polytopes in halfspace representation. *http://www-control.eng.cam.ac.uk/˜cnj22/docs/resp_mar_04_15.pdf*, 2004.

[63] E. Kerrigan. *Robust Constraint Satisfaction: Invariant Sets and Predictive Control*. PhD thesis, Cambridge, 2000.

[64] E. Kerrigan and J. Maciejowski. Robust feasibility in model predictive control: Necessary and sufficient conditions. In *Proceedings of the IEEE Conference on Decision and Control*, 2001.

[65] E. Kerrigan and J. Maciejowski. Robustly stable feedback min-max model predictive control. In *Proceedings of the American Control Conference*, 2003.

[66] E. Kerrigan and J. Maciejowski. Feedback min-max model predictive control using a single linear program: Robust stability and the explicit solution. *International Journal of Robust and Nonlinear Control*, 4(14):395–413, 2004.

[67] I. Kolmanovsky and E. G. Gilbert. Theory and computation of disturbance invariant sets for discrete-time linear systems. *Mathematical Problems in Engineering: Theory, Methods and Applications*, 4:317–367, 1998.

[68] M. V. Kothare, V. Balakrishnan, and M. Morari. Robust constrained model predictive control using linear matrix inequalities. *Automatica*, 32:1361–1379, 1996.

[69] B. Kouvaritakis and M. Cannon. *Non-linear Predictive Control: theory and practice*. Institution of Electrical Engineers, 2001.

[70] B. Kouvaritakis, J.A. Rossiter, and J. Schuurmans. Efficient robust predictive control. *IEEE Transactions on Automatic Control*, 45(8):1545–1549, 2000.

[71] J. C. Lagarias and Y. Wang. The finiteness conjecture for the generalized spectral radius of a set of matrices. *Linear Algebra and its Applications*, (214):17–42, 1995.

[72] W. Langson, I. Chryssochoos, S. V. Raković, and D. Q. Mayne. Robust model predictive control using tubes. *Automatica*, 40(1):125–133, 2004.

[73] E.B. Lee and L. Markus. *Foundations of optimal control theory*. New York, Wiley, 1967.

[74] J.-W. Lee, W. H. Kwon, and J. Choi. On stability of constrained receding horizon control with finite terminal weighting matrix. *Automatica*, 34:1607–1612, 1998.

[75] Y. I. Lee and B. Kouvaritakis. Robust receding horizon predictive control for systems with uncertain dynamics and input saturation. *Automatica*, 36:1497–1504, 2000.

[76] W. S. Levine. *The Control Handbook*. CRC Press, 1996.

[77] M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret. Applications of second-order cone programming. *Linear Algebra and its Applications*, (284):193–228, 1998.

[78] D. G. Luenberger. *Linear and Nonlinear Programming, 2nd Edition*. Addison-Wesley, 1984.

[79] J. M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, 2001.

[80] L. Magni, G. De Nicolao, L. Magnani, and R. Scattolini. A stabilizing model-based predictive control algorithm for nonlinear systems. *Automatica*, 37:1351–1362, 2001.

[81] L. Magni and R. Sepulchre. Stability margins of nonlinear receding-horizon control via inverse optimality. *Systems and Control Letters*, 32:241–245, 1997.

[82] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36:789–814, 2000.

[83] D. Q. Mayne, M. M. Seron, and S. V. Raković. Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica*, 41:219–224, 2005.

[84] H. Michalska and D. Q. Mayne. Robust receding horizon control of constrained systems. *IEEE Transactions on Automatic Control*, 38(11):1623–1633, 1993.

[85] Mosek 3.0 MATLAB optimization toolbox user's manual. http://www.mosek.com/documentation.html.

[86] Multi-Parametric Toolbox (MPT), MATLAB toolbox form multi-parametric programming and computational geometry. http://control.ee.ethz.ch/~mpt/.

[87] D. Muñoz de la Peña, T. Alamo, and A. Bemporad. A decomposition algorithm for feedback min-max model predictive control. In *Proceedings of the IEEE Conference on Decision and Control and the European Control Conference*, pages 5126–5131, 2005.

[88] K. R. Muske and J. B. Rawlings. Model predictive control with linear models. *AIChE Journal*, 39(2):262–287, 1993.

[89] Y. E. Nesterov and A. S. Nemirovski. *Interior Point Polynomial Algorithms in Convex Programming*. SIAM Publications, Philadelphia, USA, 1994.

[90] J. O'Rourke. *Computational Geometry in C, Second Edition*. Cambridge University Press, 1998.

[91] A. Packard and J. Doyle. The complex structured singular value. *automatica*, 29:71–109, 1993.

[92] B.-G. Park, J.-W. Lee, and W. H. Kwon. Robust one-step receding horizon control for constrained systems. *International Journal of Robust and Nonlinear Control*, 9:381–395, 1999.

[93] B. Pluymers, M. V. Kothare, J. A. K. Suykens, and B. De Moor. Robust synthesis of constrained linear state feedback using LMIs and polyhedral invariant sets. In *Proceedings of the American Control Conference, Minneapolis, USA*, 2006.

[94] B. Pluymers, M.V. Kothare, J.A.K. Suykens, and B. De Moor. Comments on "min-max predictive control strategies for input-saturated polytopic uncertain systems". *Submitted for publication*, 2006, (http://www.esat.kuleuven.be/˜sistawww/cgi-bin/pub.pl).

[95] B. Pluymers, L. Roobrouck, J. Buijs, J. A. K. Suykens, and B. De Moor. An LMI-based constrained MPC-scheme with time-varying terminal cost. In *Proceedings of the IFAC Symposium on Advanced Control of Chemical Processes, Hong Kong, China*, 2003.

[96] B. Pluymers, L. Roobrouck, J. Buijs, J. A. K. Suykens, and B. De Moor. Model-predictive control with time-varying terminal cost using convex combinations. *Automatica*, 41(5):831–837, 2005.

[97] B. Pluymers, L. Roobrouck, J. Buijs, J. A. K. Suykens, and B. De Moor. Model-predictive control with time-varying terminal cost using convex combinations. *Internal Report 04-028, ESAT-SISTA, K. U. Leuven (Leuven, Belgium)*, 2005, (http://www.esat.kuleuven.be/˜sistawww/cgi-bin/pub.pl).

[98] B. Pluymers, J. A. Rossiter, J. A. K. Suykens, and B. De Moor. The efficient computation of polyhedral invariant sets for linear systems with polytopic uncertainty description. In *Proceedings of the American Control Conference (ACC), Portland, USA*, pages 804–809, 2005.

[99] B. Pluymers, J. A. Rossiter, J. A. K. Suykens, and B. De Moor. Interpolation based MPC for LPV systems using polyhedral invariant sets. In *Proceedings of the American Control Conference (ACC), Portland, USA*, pages 810–815, 2005.

[100] B. Pluymers, J. A. Rossiter, J. A. K. Suykens, and B. De Moor. A simple algorithm for robust MPC. In *Proceedings of the IFAC World Congress 2005, Prague, Czech Republic*, 2005.

[101] B. Pluymers, J. A. Rossiter, J.A.K. Suykens, and B. De Moor. Efficient computation of polyhedral invariant sets for LPV systems and application to robust MPC. *Submitted for publication*, 2005, (http://www.esat.kuleuven.be/˜sistawww/cgi-bin/pub.pl).

[102] B. Pluymers, J. Suykens, and B. De Moor. Construction of reduced complexity polyhedral invariant sets for lpv systems using linear programming. *Submitted for publication*, 2006, (http://www.esat.kuleuven.be/˜sistawww/cgi-bin/pub.pl).

[103] B. Pluymers, J. A. K. Suykens, and B. De Moor. Linear model predictive control with time-varying terminal cost using sparse convex combinations and bisection searching. In *Proceedings of the IEEE Conference on Decision and Control (CDC), Paradise Island, Bahamas*, pages 2029–2034, 2004.

[104] B. Pluymers, J. A. K. Suykens, and B. De Moor. Robust finite-horizon MPC using optimal worst-case closed-loop predictions. In *Proceedings of the IEEE Conference on Decision and Control (CDC), Paradise Island, Bahamas*, pages 2503–2508, 2004.

[105] B. Pluymers, J. A. K. Suykens, and B. De Moor. Min-max feedback MPC using a time-varying terminal constraint set and comments on efficient robust constrained model predictive control with a time-varying terminal constraint set. *Systems and Control Letters*, 54(12):1143–1148, 2005.

[106] B. Pluymers, J. A. K. Suykens, and B. De Moor. Two key ingredients enabling efficient long-horizon robust MPC. *Internal Report 05-199, ESAT-SISTA, K. U. Leuven (Leuven, Belgium)*, 2005, (http://www.esat.kuleuven.be/˜sistawww/cgi-bin/pub.pl).

[107] F. P. Preparata and M. I. Shamos. *Computational Geometry*. Springer-Verlag, New York, 1985.

[108] J. A. Primbs and V. Nevistić. Feasibility and stability of constrained finite receding horizon control. *Automatica*, 36:965–971, 2000.

[109] A.I. Propoi. Use of linear programming methods for synthesizing sampled-data automatic systems. *Automatic Remote Control*, 24(7):837–844, 1963.

[110] Mayne. D. Q. Optimization in model based control. In *Proceedings of the IFAC symposium on dynamics and control chemical reactors and batch processes*, pages 229–242, 1995.

[111] S. J. Qin and T. A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11:733–764, 2003.

[112] S.J. Qin and T.A. Badgwell. An overview of industrial model predictive control technology. *AIChE Symposium Series 316*, 93:232–256, 1996.

[113] S. Raković. *Robust control of constrained discrete time systems: Characterization and implementation*. PhD thesis, Imperial College, 2005.

[114] S. Raković, E. Kerrigan, K. Kouramas, and D. Mayne. Invariant approximations of the minimal robust positively invariant set. *IEEE Transactions on Automatic Control*, 50(3):406–410, 2005.

[115] J. B. Rawlings and K. R. Muske. Stability of constrained receding horizon control. *IEEE Transactions on Automatic Control*, 38:1512–1516, 1993.

[116] O. J. Rojas and G. C. Goodwin. A simple antiwindup strategy for state constrained linear control. In *Proceedings of the IFAC World Congress*, 2002.

[117] J. A. Rossiter. *Model Based Predictive Control*. CRC Press, 2003.

[118] J. A. Rossiter, Pluymers. B, Y. Ding, X. Lei, J. A. K. Suykens, and B. De Moor. Extending the feasibility of interpolation based mpc and application to LPV systems. *Submitted for publication*, 2005, (http://www.esat.kuleuven.be/˜sistawww/cgi-bin/pub.pl).

[119] J. A. Rossiter, Pluymers. B., J. A. K. Suykens, and B. De Moor. A multi parametric quadratic programming solution to robust predictive control. In *Proceedings of the IFAC World Congress, Prague, Czech Republic*, 2005.

[120] J. A. Rossiter, Y. Ding, B. Pluymers, J. A. K. Suykens, and B. De Moor. Interpolation based MPC with exact constraint handling: the uncertain case. In *Proceedings of the joint European Control Conference & IEEE Conference on Decision and Control, Sevilla, Spain*, 2005.

[121] J. A. Rossiter, Y. Ding, L. Xi, B. Pluymers, J. A. K. Suykens, and B. De Moor. Interpolation based MPC with exact constraint handling: the nominal case. *Internal Report 04-231, ESAT-SISTA, K. U. Leuven (Leuven, Belgium)*, 2005, (http://www.esat.kuleuven.be/˜sistawww/cgi-bin/pub.pl).

[122] J. A. Rossiter, B. Kouvaritakis, and M. Bacic. Interpolation based computationally efficient predictive control. *International Journal of Control*, 77(3):290–301, 2004.

[123] J. A. Rossiter, B. Kouvaritakis, and M. J. Rice. A numerically robust state-space approach to stable predictive control strategies. *Automatica*, 34:65–73, 1998.

[124] J. A. Rossiter and B. Pluymers. The potential of interpolation for simplifying predictive control and applicationto lpv systems. In *Proceedings of the International Workshop on Assessment and Future Directions in NMPC, Freudenstadt - Lauterbad, Germany*, 2005.

[125] J. A. Rossiter, B. Pluymers, and B. De Moor. volume Assessment and Future Directions in Nonlinear Model Predictive Control of *Lecture Notes in Control and Information Sciences*, chapter The potential of interpolation for simplifying predictive control and application to LPV systems. Springer, 2006.

[126] G.-C. Rota and W. G. Strang. A note on the joint spectral radius. *Indag. Math.*, 22:379–381, 1960.

[127] R. Routledge. *Discoveries & Inventions of the Nineteenth Century, 13th edition*. 1900.

[128] W. J. Rugh and J. S. Shamma. Research on gain scheduling. *Automatica*, 36(10):1401–1425, 2000.

[129] P. O. M. Scokaert and D. Q. Mayne. Min-max feedback model predictive control for constrained linear systems. *IEEE Transactions on Automatic Control*, 43(8):1136–1142, 1998.

[130] P. O. M. Scokaert, D. Q. Mayne, and J. B. Rawlings. Suboptimal model predictive control (feasibility implies stability). *IEEE Transactions on Automatic Control*, 44(3):648–654, 1999.

[131] P. O. M. Scokaert and J. B. Rawlings. Constrained linear quadratic regulation. *IEEE Transactions on Automatic Control*, 43(8):1163–1168, 1998.

[132] P. O. M. Scokaert and J. B. Rawlings. Feasibility issues in model predictive control. *AIChE Journal*, 45(8):1649–1659, 1999.

[133] Sedumi 1.1 MATLAB SDP optimization toolbox. http://sedumi.mcmaster.ca/.

[134] J. S. Shamma and D. Xiong. Set-valued methods for linear parameter varying systems. *Automatica*, 35:1081–1089, 1999.

[135] M. Sznaier and M. J. Damborg. Suboptimal control of linear systems with state and control inequality constraints. In *Proceedings of the IEEE Conference on Decision and Control, Los Angeles, USA*, pages 761–762, 1987.

[136] K. Takaba. Robust servomechanism with preview action for polytopic uncertain systems. *International Journal of Robust and Nonlinear Control*, 10:101–111, 2000.

[137] J. Theys. *Joint Spectral Radius: theory and approximations*. PhD thesis, Université Catholique de Louvain, 2005.

[138] M. J. Todd. The many facets of linear programming. *Mathematical Programming Series B*, (91):417–436, 2002.

[139] P. Tøndel, T. A. Johansen, and A. Bemporad. An algorithm for multi-parametric quadratic programming and explicit MPC solutions. *Automatica*, 39(3):480–497, 2003.

[140] Z. Wan and M. V. Kothare. Robust output feedback model predictive control using off-line linear matrix inequalities. *Journal of Process Control*, 12:763–774, 2002.

[141] Z. Wan and M. V. Kothare. An efficient off-line formulation of robust model predictive control using linear matrix inequalities. *Automatica*, 39(5):837–846, 2003.

[142] Z. Wan and M. V. Kothare. Efficient robust constrained model predictive control with a time varying terminal constraint set. *Systems and Control Letters*, 48:375–383, 2003.

[143] Z. Wan, B. Pluymers, M. V. Kothare, and B. De Moor. Comments on "efficient robust constrained model predictive control with a time varying terminal constraint set" [systems & control letters 48 (2003) 375-383]. *Accepted for publication in Systems & Control Letters*, 2006, (http://www.esat.kuleuven.be/˜sistawww/cgi-bin/pub.pl).

[144] Z. Q. Zheng and M. Morari. Robust stability of constrained model predictive control. In *Proceedings of the American Control Conference*, pages 379–383, San Francisco, 1993.

[145] K. Zhou and J. C. Doyle. *Essentials of Robust Control*. Prentice Hall, 1997.

[146] G. M. Ziegler. *Lectures on Polytopes*. Graduate Texts in Mathematics, Springer, 1995.

Curriculum Vitae

Bert Pluymers was born August 24, 1979 in Geel, Belgium.

He received his M.Sc. in Electrical Engineering degree (Data mining & Automation) from the Katholieke Universiteit Leuven, Belgium in July 2002. The subject of his Master's thesis was "Robust and adaptive model based controller design via neural network models" and was finished under the supervision of Prof. Johan Suykens.

In October 2002 he started a Ph.D. at the SCD-SISTA research group at the Electrical Engineering Department of the Katholieke Universiteit Leuven, under the supervision of Prof. Bart De Moor and Prof. Johan Suykens, supported by a Ph.D. grant of the IWT Flanders.

In 2005 he received the Student Best Paper Award of the American Control Conference for the paper entitled "Interpolation Based MPC for LPV Systems using Polyhedral Invariant Sets", which he co-authored with Prof. J. A. Rossiter (Sheffield University, UK) and Prof. De Moor and Prof. Suykens. In 2006 he is selected as a finalist for the same award of 2006 American Control Conference with the paper entitled "Robust Synthesis of Constrained Linear State Feedback using LMIs and Polyhedral Invariant Sets", which he co-authored with Prof. M. V. Kothare (Lehigh University, USA) and Prof. De Moor and Prof. Suykens.

Publications by the author

Book chapters

- J. A. Rossiter, B. Pluymers, and B. De Moor. volume Assessment and Future Directions in Nonlinear Model Predictive Control of *Lecture Notes in Control and Information Sciences*, chapter The potential of interpolation for simplifying predictive control and application to LPV systems. Springer, 2006.

Journal papers

- B. Pluymers, L. Roobrouck, J. Buijs, J. A. K. Suykens, and B. De Moor. Model-predictive control with time-varying terminal cost using convex combinations. *Automatica*, 41(5):831–837, 2005.

- B. Pluymers, J. A. K. Suykens, and B. De Moor. Min-max feedback MPC using a time-varying terminal constraint set and comments on efficient robust constrained model predictive control with a time-varying terminal constraint set. *Systems and Control Letters*, 54(12):1143–1148, 2005.

- Z. Wan, B. Pluymers, M. V. Kothare, and B. De Moor. Comments on "efficient robust constrained model predictive control with a time varying terminal constraint set" [systems & control letters 48 (2003) 375-383]. *Accepted for publication in Systems & Control Letters*, 2006.

- B. Pluymers, J. A. Rossiter, J.A.K. Suykens, and B. De Moor. Efficient computation of polyhedral invariant sets for LPV systems and application to robust MPC. *Submitted for publication*, 2006.

- B. Pluymers, J. Suykens, and B. De Moor. Construction of reduced complexity polyhedral invariant sets for lpv systems using linear programming. *Submitted for publication*, 2006.

- B. Pluymers, M.V. Kothare, J.A.K. Suykens, and B. De Moor. Comments on "min-max predictive control strategies for input-saturated polytopic uncertain systems". *Submitted for publication*, 2006.

Conference papers

- B. Pluymers, L. Roobrouck, J. Buijs, J. A. K. Suykens, and B. De Moor. An LMI-based constrained MPC-scheme with time-varying terminal cost. In *Proceedings of the IFAC Symposium on Advanced Control of Chemical Processes, Hong Kong, China*, 2003.

- B. Pluymers, J. A. K. Suykens, and B. De Moor. Linear model predictive control with time-varying terminal cost using sparse convex combinations and bisection searching. In *Proceedings of the IEEE Conference on Decision and Control (CDC), Paradise Island, Bahamas*, pages 2029–2034, 2004.

- B. Pluymers, J. A. K. Suykens, and B. De Moor. Robust finite-horizon MPC using optimal worst-case closed-loop predictions. In *Proceedings of the IEEE Conference on Decision and Control (CDC), Paradise Island, Bahamas*, pages 2503–2508, 2004.

- B. Pluymers, J. A. Rossiter, J. A. K. Suykens, and B. De Moor. The efficient computation of polyhedral invariant sets for linear systems with polytopic uncertainty description. In *Proceedings of the American Control Conference (ACC), Portland, USA*, pages 804–809, 2005.

- B. Pluymers, J. A. Rossiter, J. A. K. Suykens, and B. De Moor. Interpolation based MPC for LPV systems using polyhedral invariant sets. In *Proceedings of the American Control Conference (ACC), Portland, USA*, pages 810–815, 2005.[1]

- B. Pluymers, J. A. Rossiter, J. A. K. Suykens, and B. De Moor. A simple algorithm for robust MPC. In *Proceedings of the IFAC World Congress 2005, Prague, Czech Republic*, 2005.

- J. A. Rossiter, Pluymers. B., J. A. K. Suykens, and B. De Moor. A multi parametric quadratic programming solution to robust predictive control. In *Proceedings of the IFAC World Congress, Prague, Czech Republic*, 2005.

- J. A. Rossiter and B. Pluymers. The potential of interpolation for simplifying predictive control and applicationto lpv systems. In *Proceedings of the International Workshop on Assessment and Future Directions in NMPC, Freudenstadt - Lauterbad, Germany*, 2005.

- J. A. Rossiter, Y. Ding, B. Pluymers, J. A. K. Suykens, and B. De Moor. Interpolation based MPC with exact constraint handling: the uncertain case. In *Proceedings of the joint European Control Conference & IEEE Conference on Decision and Control, Sevilla, Spain*, 2005.

- B. Pluymers, M. V. Kothare, J. A. K. Suykens, and B. De Moor. Robust synthesis of constrained linear state feedback using LMIs and polyhedral invariant sets. In *Proceedings of the American Control Conference, Minneapolis, USA*, 2005.[2]

[1] **Winner of the Student Best Paper Award of the American Control Conference 2005**
[2] **Finalist for the Student Best Paper Award of the American Control Conference 2006**

- L. Imsland, J. A. Rossiter, B. Pluymers, and J. Suykens. Robust triple mode MPC. In *Proceedings of the American Control Conference 2006, Minneapolis, USA*, pages CD–ROM, 2006.

- I. S. Choi, J. A. Rossiter, Fleming P. Pluymers, B., and B. De Moor. A robust MPC design for hot rolling mills: a polyhedral invariant set approach. In *Proceedings of the American Control Conference, Minneapolis, USA*, 2005.

Technical reports

- B. Pluymers, L. Roobrouck, J. Buijs, J. A. K. Suykens, and B. De Moor. Model-predictive control with time-varying terminal cost using convex combinations. *Internal Report 04-028, ESAT-SISTA, K. U. Leuven (Leuven, Belgium)*, 2005.

- B. Pluymers, J. A. K. Suykens, and B. De Moor. Two key ingredients enabling efficient long-horizon robust MPC. *Internal Report 05-199, ESAT-SISTA, K. U. Leuven (Leuven, Belgium)*, 2005.

Other publications

This section contains other publications (journal, conference, ...) by the author that are not directly related to this thesis, but that were written in the framework of other projects in which the author was involved.

- European Patent WO03080157 (pending in US, AU). G. Van Den Berghe, D. Berckmans, J.-M. Aerts, B. De Moor, B. Pluymers, F. De Smet. Automatic infusion system based on an adaptive patient model, 2003.

- E. Haesen, M. Espinoza, B. Pluymers, I. Goethals, V. Thong, J. Driesen, R. Belmans, and B. De Moor. Optimal placement and sizing of distributed generator units using genetic optimization algorithms. *Electrical Power Quality and Utilisation Journal*, 11(1):97–104, 2005.

- T. Van Herpe, M. Espinoza, B. Pluymers, I. Goethals, P. Wouters, G. Van den Berghe, and B. De Moor. The development of an adaptive input-output model to predict glycemia of critically ill patients. *Submitted for publication*, 2006.

- T. Van Herpe, I. Goethals, B. Pluymers, F. De Smet, P. Wouters, G. Van den Berghe, and B. De Moor. Challenges in data-based patient modeling for glycemia control in ICU-patients. In *3rd IASTED International Conference on Biomedical Control Engineering, Innsbruck, Austria*, pages 685–690, 2005.

- T. Van Herpe, M. Espinoza, B. Pluymers, P. Wouters, F. De Smet, G. Van den Berghe, and B. De Moor. Development of a critically ill patient input-output model. In *Proceedings of the IFAC symposium of system identification*, 2006.

- T. Van Herpe, B. Pluymers, M. Espinoza, G. Van den Berghe, and B. De Moor. A minimal model for the glycemia control in critically ill patients. *Submitted for publication*, 2006.