Robust model predictive control for deployment and reconfiguration of multi-agent systems

Thomas Chevet

université
PARIS-SACLAY

Robust model predictive control for deployment and reconfiguration of multi-agent systems

Thèse de doctorat de l'Université Paris-Saclay

École doctorale n° 580, Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat : Automatique
Unité de recherche : Université Paris-Saclay, CNRS, CentraleSupélec, Laboratoire des signaux et systèmes, 91190, Gif-sur-Yvette, France
Référent : Faculté des sciences d'Orsay

Thèse présentée et soutenue à Gif-sur-Yvette, le 25 septembre 2020, par

Thomas CHEVET

Composition du jury :

Saïd MAMMAR Professeur, Université d'Évry Val-d'Essonne/IBISC	Président
Pedro CASTILLO Chargé de recherche, HDR, Université de Technologie de Compiègne/Heudiasyc	Rapporteur & Examinateur
Nicolas LANGLOIS Professeur, ESIGELEC/IRSEEM	Rapporteur & Examinateur
Mirko FIACCHINI Chargé de recherche, HDR, Université Grenoble-Alpes/GIPSA-lab	Examinateur
Didier THEILLIOL Professeur, Université de Lorraine/CRAN	Examinateur
Cristina STOICA MANIU Professeur, CentraleSupélec/L2S	Directrice de thèse
Cristina VLAD Maître de conférences, CentraleSupélec/L2S	Coencadrante & Examinatrice
Youmin ZHANG Professeur, Université Concordia	Coencadrant & Examinateur
Eduardo F. CAMACHO Professeur, Université de Séville	Invité
José M. MAESTRE Professeur associé, Université de Séville	Invité

Thèse de doctorat

NNT: 2020UPASG007

NNT : 2020UPASG007

université PARIS-SACLAY

Robust model predictive control for deployment and reconfiguration of multi-agent systems

Thèse de doctorat de l'Université Paris-Saclay

École doctorale n°580 STIC
Sciences et Technologies de l'Information et de la Communication
Spécialité de doctorat
Automatique
Unité de recherche
Université Paris-Saclay, CNRS, CentraleSupélec, Laboratoire des signaux et systèmes, 91190, Gif-sur-Yvette, France
Référent
Faculté des sciences d'Orsay

Thèse présentée et soutenue à Gif-sur-Yvette, le 25 septembre 2020, par

Thomas Chevet

Composition du jury :

Saïd Mammar Professeur, Université d'Évry Val-d'Essonne/IBISC	Président
Pedro Castillo Chargé de recherche, HDR, Université de Technologie de Compiègne/Heudiasyc	Rapporteur & Examinateur
Nicolas Langlois Professeur, ESIGELEC/IRSEEM	Rapporteur & Examinateur
Mirko Fiacchini Chargé de recherche, HDR, Université Grenoble-Alpes/GIPSA-lab	Examinateur
Didier Theilliol Professeur, Université de Lorraine/CRAN	Examinateur
Cristina Stoica Maniu Professeur, CentraleSupélec/L2S	Directrice de thèse
Cristina Vlad Maître de conférences, CentraleSupélec/L2S	Coencadrante & Examinatrice
Youmin Zhang Professeur, Université Concordia	Coencadrant & Examinateur
Eduardo F. Camacho Professeur, Université de Séville	Invité
José M. Maestre Professeur associé, Université de Séville	Invité

Tomorrow soon turns into yesterday.
Everything we see just fades away.
There's sky and sand where mountains used to be.
Time drops by a second to eternity.

It doesn't matter if we turn to dust;
Turn and turn and turn we must!
I guess I'll see you dancin' in the ruins tonight!

Dancin' in the Ruins, Blue Öyster Cult

Acknowledgements

The preparation of this thesis was a long and exciting journey. This journey, which would not have been possible without the help of many great people, was marked by the meeting of many others. The following few words are an attempt to thank them for their time, help and support.

I would like to start by expressing my deepest gratitude to my supervising team. I thank Cristina Stoica Maniu and Cristina Vlad for their kindness, their trust, since even before the beginning of my thesis, their insightful advice, their constant support of my work, and for always encouraging me in moments when I needed it. Thanks to them, I was also able to participate in many other academic and administrative activities that reinforced the idea that I wanted to continue working in academia. I also thank Professor Youmin Zhang for his constant support of my work and his precious advice to further it, both on theoretical and practical aspects, as well as for welcoming me in his lab in Montréal to start working on the real-time implementation of the algorithms presented in this thesis. To be able to work under their guidance has been a great privilege and no words could ever express how thankful I am for all the work they have done to help me. I look forward to working with them again.

I am also grateful to the members of my thesis committee. I would like to thank Pedro Castillo, *Chargé de Recherches* CNRS at Université Technologique de Compiègne, and Nicolas Langlois, Professor at ESIGELEC, for accepting to review my manuscript. I also thank Mirko Fiacchini, *Chargé de Recherches* CNRS at GIPSA-lab, and Didier Theilliol, Professor at Université de Lorraine, for accepting to be part of my committee and Saïd Mammar, Professor at Université d'Évry Val-d'Essonne, for presiding it. The questions and remarks they had during the reviewing phase and the defense are a valuable contribution to further the work presented in this thesis.

It has been an honor being able to work with José Maestre and Eduardo Camacho at Universidad de Sevilla. I thank them both for welcoming me in their team, for their kindness and for the time they took to discuss new ideas and improvements of my work. Their scientific insight and excellence has been of the greatest importance to help me see the relevance and potential of my work. I really hope we will be able to continue our collaboration in the future.

A special acknowledgment has to go to Alain Théron, Professor of Industrial Sciences at Lycée Pierre de Fermat in Toulouse, who, with his classes, convinced me that Automatic control was what I wanted to work with when he "brought us to the Dark Side of the Force" by introducing the Laplace transform.

This feeling became even stronger during the seven years I spent at Centrale-Supélec (which was still Supélec when I entered the school in 2013). For that, I thank all the members of the Automatic Control Department of the school, Didier Dumur,

Emmanuel Godoy, Gilles Duc, Dominique Beauvois, Guillaume Sandou, Sorin Olaru, Pedro Rodriguez, Stéphane Font, Maria Makarov, Houria Siguerdidjane, Giorgio Valmorbida, Sihem Tebbani, Antoine Chaillet and, of course, my two supervisors Cristina Stoica Maniu and Cristina Vlad, for the high quality teaching and supervision they provide to engineering students and that I could receive myself. I also thank them for the many teaching opportunities that were given to me as a teaching assistant for the Automatic course, for all the lunch break tarot games and for all the discussions, scientific or not, that we had. I would also like to extend my thanks to Israel Hinostroza, researcher at the SONDRA laboratory, for his support during the supervision, with Maria Makarov and Cristina Stoica Maniu, of my graduation project, a project that launched me into the world of research with my first scientific publication. Finally, I would like to thank Léon Marquet and Caroline Charles for their prompt and excellent technical support in all circumstances and for all the discussions we had about electronics and informatics.

I am grateful to the Laboratoire des Signaux et Systèmes and its LIA on Information, Learning and Control for the funding of my stay in Professor Zhang's laboratory in Montréal. I would also like to thank the STIC Doctoral School for the international mobility grant that allowed me to go to work in Seville.

This doctoral journey would not have been possible without the support of my friends. I would like to thank all my friends from Supélec who were there to support me during the last three years, and mainly Gwendoline, Mony, Thomas and Yaël for the board game afternoons that we had, as well as Matthias and Marine, that I have not seen as much as I would have liked. I also have to thank Brice, one of the best friend I had through our time in *prépa*, Supélec and beyond, who has always been there to help and encourage me when I needed it. This journey also allowed me to meet many companions who made the trip more pleasant with all the discussions, the lunch break Mölkky and tarot games or the board games and theater evenings that we had. I would then like to thank Vincent for our never ending discussions on any trivial subject that arose, Maxime for our philosophical debates on Automatic control and research in general, Gauthier for all our discussions on music and drones, Dory for all the work and jokes we did together and his deadly aim at Mölkky, Geoffray and Martin for their encyclopedic knowledge of history and philosophy, Dario for his unconditional defense of Genoa, Joy for her constant and communicative good mood and positivity, Matthieu for his unfortunately unsuccessful attempts to keep us in a righteous path, Fetra for his calm and serenity, Kodjo for his exuberance and the debates only he is able to provoke, Antonello for his attempts to channel Vincent, Andreea for her kindness, Nicolò for his craziness, as well as the others, Jérémy, Ugo, Baptiste, Jean, Mert, Daniel, Benjamin and Merouane, with whom I spent less time but were always there to participate in any of the discussions, debates or activities that I mentioned.

Last but not least, my thanks and love go to my family, my mother, my father, my sister, my stepfather and my grandparents who have always been there to advise and support me in my decisions.

Table of Contents

List of Figures

Chapter 3 Decentralized control for the deployment of a multi-vehicle system

Chapter 4 Deployment of a multi-vehicle system subject to perturbations

List of Definitions

List of Assumptions

List of Acronyms

BMI	Bilinear Matrix Inequality
CC	Chebyshev configuration
CCMPC	Chance Constrained Model Predictive Control
FTC	Fault-Tolerant Control
FTFC	Fault-Tolerant Formation Control
GV	Guaranteed Voronoi
GVC	Guaranteed Voronoi Cell
LMI	Linear Matrix Inequality
LTI	Linear Time Invariant
MAS	Multi-Agent System
MPC	Model Predictive Control
mPI	Minimal Positive Invariant
mRPI	Minimal Robust Positive Invariant
MVS	Multi-Vehicle System
RPI	Robust Positive Invariant
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
ZOH	Zero Order Hold

List of Symbols

Sets

$\mathbb{N}$	Set of the nonnegative integers
$\overline{n,m}$	Set of all integers i such that $n \leq i \leq m$, with $n, m \in \mathbb{N}$
$\mathbb{R}$, $\mathbb{R}^*$, $\mathbb{R}_+$	Set of the real numbers, of the non-zero real numbers and of the positive real numbers
$\mathbb{R}^n$	Set of n-dimensional real vectors
$\mathbb{B}^n$	Unitary box in $\mathbb{R}^n$
$\mathbb{B}^n(\boldsymbol{\alpha})$, $\boldsymbol{\alpha} \in \mathbb{R}^n$	Box in $\mathbb{R}^n$
$\mathbb{R}^{n \times m}$	Set of all real n-by-m matrices
$\{\mathbf{x}\}$	Singleton set, i.e. a set which only contains one element (here vector $\mathbf{x}$)
$\emptyset$	Empty set
$\partial\mathcal{A}$	Boundary of set $\mathcal{A}$
$\mathcal{A} \subseteq \mathcal{B}$ (resp. $\mathcal{A} \subset \mathcal{B}$)	$\mathcal{A}$ is a subset (resp. strict subset) of $\mathcal{B}$
$\mathcal{A} \setminus \mathcal{B}$	Set difference of $\mathcal{A}$ and $\mathcal{B}$, the elements of $\mathcal{A}$ that are not in $\mathcal{B}$
$\mathcal{A} \times \mathcal{B}$	Cartesian product of sets $\mathcal{A}$ and $\mathcal{B}$
$\mathcal{A} \cap \mathcal{B}$	Intersection of sets $\mathcal{A}$ and $\mathcal{B}$
$\mathcal{A} \cup \mathcal{B}$	Union of sets $\mathcal{A}$ and $\mathcal{B}$
$\mathcal{A} \oplus \mathcal{B}$	Minkowski sum of sets $\mathcal{A}$ and $\mathcal{B}$
$\mathcal{A} \ominus \mathcal{B}$	Pontryagin difference of sets $\mathcal{A}$ and $\mathcal{B}$
$\lvert\mathcal{A}\rvert$	Cardinality of set $\mathcal{A}$, i.e. the number of elements in $\mathcal{A}$

Algebra

$a \in \mathbb{R}$	A real scalar
$\mathbf{x} \in \mathbb{R}^n$	A real vector of n elements

$\boldsymbol{A} \in \mathbb{R}^{n \times m}$	A real matrix of n rows and m columns
$\boldsymbol{I}_n$	Identity matrix of size n-by-n
$\mathbf{1}_{n \times m}$, $\mathbf{1}_n$	Matrices filled with ones of size n-by-m and n-by-n
$\mathbf{0}_{n \times m}$, $\mathbf{0}_n$	Matrices filled with zeros of size n-by-m and n-by-n
$\boldsymbol{A}^\top$	Transpose of matrix $\boldsymbol{A}$
$\boldsymbol{A}^{-1}$	Inverse of matrix $\boldsymbol{A}$
$\boldsymbol{A} \succ 0$ (resp. $\boldsymbol{A} \succeq 0$)	Matrix $\boldsymbol{A}$ is positive definite (resp. semidefinite)
$\mathbf{x} > 0$ (resp. $\mathbf{x} \geq 0$)	Element-wise positivity (resp. nonnegativity) of vector $\mathbf{x}$
$\lvert\mathbf{x}\rvert$, $\mathbf{x} \in \mathbb{R}^n$	Element-wise absolute value of vector $\mathbf{x}$
$\lVert\mathbf{x}\rVert_2$	Euclidean norm of vector $\mathbf{x}$ such that $\lVert\mathbf{x}\rVert_2 = \sqrt{\mathbf{x}^\top \mathbf{x}}$
$\lVert\mathbf{x}\rVert_{\boldsymbol{Q}}$	Weighted norm of vector $\mathbf{x}$ such that $\lVert\mathbf{x}\rVert_{\boldsymbol{Q}} = \sqrt{\mathbf{x}^\top \boldsymbol{Q}\mathbf{x}}$
$\boldsymbol{A} \otimes \boldsymbol{B}$	Kronecker product of matrices $\boldsymbol{A}$ and $\boldsymbol{B}$
$\det(\mathbf{x}, \mathbf{y})$, $\mathbf{x}, \mathbf{y} \in \mathbb{R}^2$	Determinant of the matrix $\begin{bmatrix}\mathbf{x} & \mathbf{y}\end{bmatrix} \in \mathbb{R}^{2\times 2}$

Probabilities

$\mathbb{P}(X \leq x)$	Probability that the continuous random variable $X \in \mathbb{R}$ is less than or equal to $x \in \mathbb{R}$
$\mathbb{P}(\mathbf{X} \in \mathcal{A})$	Probability that the continuous multivariate random variable $\mathbf{X} \in \mathbb{R}^n$ belongs to the set $\mathcal{A} \subset \mathbb{R}^n$
$\mathbb{E}(X)$	Mathematical expectation of the continuous random variable $X \in \mathbb{R}$
$\boldsymbol{\mu}_X$	Mean of the continuous random variable $X \in \mathbb{R}$, alternative notation for $\mathbb{E}(X)$
$\mathbb{E}(\mathbf{X})$	Mathematical expectation of the continuous multivariate random variable $\mathbf{X} \in \mathbb{R}^n$
$\boldsymbol{\mu}_{\mathbf{X}}$	Mean of the continuous multivariate random variable $\mathbf{X} \in \mathbb{R}^n$, alternative notation for $\mathbb{E}(\mathbf{X})$
$\boldsymbol{\Sigma}_{\mathbf{X}}$	Variance matrix of the continuous multivariate random variable $\mathbf{X} \in \mathbb{R}^n$

Multi-agent system

Σ	Multi-agent system
$\mathcal{V}_i(k)$	Voronoi cell of agent i at time k
$\mathcal{V}_i^g(k)$	Guaranteed Voronoi cell of the set $\mathcal{W}_i$ at time k

$\partial\mathcal{V}_{i\,j}^{g}(k)$	Border of the guaranteed Voronoi cell of the set $\mathcal{W}_i$ induced by generator set $\mathcal{W}_j$ at time k
$\mathcal{V}_i^{p}(k)$	Pseudo-Voronoi cell of agent i at time k
$\partial\mathcal{V}_{i\,j}^{p}(k)$	Border of the pseudo-Voronoi cell of agent i induced by the generator j at time k
$\mathbf{c}_i(k)$	Chebyshev center of a set at time k

Robust control

$\mathbf{x}_i$	State of agent i subject to perturbations
$\check{\mathbf{x}}_i$	Nominal state of agent i
$\widehat{\mathbf{x}}_i$	Estimated state of agent i
$\widetilde{\mathbf{x}}_i$	Estimation error $\mathbf{x}_i - \widehat{\mathbf{x}}_i$ of agent i
$\check{\mathbf{x}}_i$	Deviation error $\widehat{\mathbf{x}}_i - \check{\mathbf{x}}_i$ of agent i
$\mathcal{S}_{\mathbf{x}}$	Approximation of the mRPI set for the dynamics of $\mathbf{x}$

Résumé en français

Sommaire

Contexte

Qu'est-ce qu'un système multi-agents ?

La notion de système multi-agents trouve ses origines dans le domaine de l'informatique. Aujourd'hui, le concept de système multi-agents est utilisé dans de nombreuses disciplines. Toutefois, comme présenté par Wooldridge and Jennings (1995), il n'est pas facile de répondre à la question « qu'est-ce qu'un agent ? » En effet, la notion d'*agent* n'a pas de définition transcendant le domaine dans lequel elle est utilisée. C'est pour cela que Wooldridge (2009) propose, sur la base d'une liste de caractéristiques présentée dans Wooldridge and Jennings (1995), une définition qui sera celle utilisée tout au long de la présente thèse.

Définition Fr.1 : Agent (Wooldridge, 2009)

Un *agent* est un système informatique se trouvant dans un *environnement* donné, capable de réaliser des actions de façon *autonome* dans cet environnement afin d'atteindre l'objectif pour lequel il a été conçu.

La liste de caractéristiques associées à cette définition dans Wooldridge and Jennings (1995) est :

(i) l'*autonomie* : agir sans l'intervention d'un opérateur humain ;

(ii) la *perception* : percevoir les changements dans son environnement par le biais de capteurs ;

(iii) l'*interaction* : communiquer avec les autres agents présents dans l'environnement ;

(iv) la *réactivité* : répondre aux changements de son environnement ;

(v) la *proactivité* : prendre des initiatives afin de remplir sa mission.

Ces caractéristiques sont rendues possibles en pratique du fait des grandes tendances de la recherche en informatique des soixante dernières années. Ainsi, un système multi-agents peut être défini comme suit.

Définition Fr.2 : Système multi-agents

Un *système multi-agents* est un ensemble d'agents possédant les caractéristiques (i)–(v).

Depuis leur introduction dans les années cinquante, les systèmes multi-agents ont été utilisés dans de nombreux domaines tels que l'économie (Čech et al., 2013, Malakhov et al., 2017, Herzog et al., 2017, Han et al., 2017, Gao et al., 2018), la sociologie (Davidsson, 2002, Li et al., 2008, Serrano and Iglesias, 2016, Ramírez et al., 2020), la biologie (Couzin et al., 2005, Roche et al., 2008, Ren et al., 2008, Colosimo, 2018), l'informatique (Ferber and Gutknecht, 1998, DeLoach et al., 2001, Bellifemine et al., 2007, Calvaresi et al., 2019) ou encore, l'automatique (Hespanha et al., 2007, Bullo et al., 2009, Dimarogonas et al., 2011, Maestre and Negenborn, 2014). C'est dans le cadre de cette dernière discipline que s'inscrit le travail présenté dans cette thèse. En effet, dans de nombreuses applications, un système complexe peut se décomposer en sous-systèmes possédant les caractéristiques (i)–(v) évoquées précédemment. Chaque sous-système peut alors être appelé agent tandis que le système complexe est appelé système multi-agents au sens de la Définition Fr.2. La différence principale avec l'informatique vient du fait que l'évolution de chaque agent est régie par un système d'équations différentielles (pour un système à temps continu) ou aux différences (pour un système à temps discret). Les agents sont alors dits *dynamiques*. Dans la suite, la dénomination « système multi-agents » est alors utilisée pour désigner un système composé de plusieurs agents dynamiques en interaction.

Une grande variété de systèmes multi-agents sont étudiés en automatique comme les *smart grids* (Logenthiran et al., 2012, Radhakrishnan and Srinivasan, 2016, Singh et al., 2017) et les *microgrids* (Dimeas and Hatziargyriou, 2005, Minchala-Avila et al., 2015), les réseaux de distribution d'eau (Wang et al., 2017, Shahdany et al., 2019), de circulation (Lin et al., 2012, Chanfreut et al., 2020), de fret (Negenborn et al., 2008, Larsen et al., 2020) ou de capteurs mobiles (Cortés et al., 2004) ou bien encore les formations multi-robots et multi-véhicules tant dans leur mouvement général (D'Andrea and Dullerud, 2003, Wurman et al., 2008, Bullo et al., 2009, Prodan et al., 2011, Alonso-Mora et al., 2015, Kamel et al., 2020) que pour leur groupement en pelotons (Ploeg et al., 2013, Turri et al., 2016, Van De Hoef et al., 2017) ou leur déploiement (Schwager et al., 2011, Nguyen and Stoica Maniu, 2016, Papatheodorou et al., 2017). Pour toutes ces applications, l'automatique va se concentrer sur l'étude des interactions entre les agents, ainsi que sur le développement de lois de commande pour permettre aux agents d'atteindre un but commun.

Toutefois, une telle diversité d'applications mène à différentes voies de recherche liées aux caractéristiques du système multi-agent étudié. En effet, la définition d'un système comme celui-ci dispose de plusieurs degrés de liberté comme la nature

de sa dynamique (linéaire variant dans le temps, par exemple), la nature de la communication entre les agents ou les contraintes physiques s'appliquant sur le système. Ainsi, la suite de ce résumé présente quelques exemples de stratégies de commande liées à certains des critères susmentionnés.

Aperçu des stratégies de commande des systèmes multi-agents

Afin d'asservir un système multi-agents, il est nécessaire de concevoir un algorithme de commande fournissant un signal d'entrée à chacun des agents pour atteindre un objectif commun. Comme cela a été évoqué précédemment, il existe différentes classes d'algorithmes de commande pour des systèmes multi-agents. De nombreuses classifications existent sur la base de paramètres divers mais ici sont seulement présentées des stratégies différenciées en fonction d'abord de la topologie de communication entre les agents, puis de la façon dont le signal d'entrée est calculé. Cette différenciation permet de donner un aperçu des techniques de commande existant pour les systèmes multi-agents.

Topologies de communication

La communication entre les agents est un élément essentiel de la commande des systèmes multi-agents. À partir de la définition du réseau de communication entre les agents, les stratégies de commande peuvent être séparées en trois catégories : les stratégies *centralisées*, *décentralisées* et *distribuées* (Tanenbaum and Van Steen, 2007).

Lorsque la commande est dite *centralisée*, chaque agent a connaissance de l'état et du signal de commande des autres agents pour calculer son propre signal d'entrée. Pour ce faire, un correcteur central collecte les informations utiles relatives à tous les agents du système et calcule le signal de commande de chaque agent avant de le lui envoyer. Les algorithmes de commande centralisés ont été abondamment étudiés au cours des dernières décennies (Xu and Hespanha, 2006, Olfati-Saber et al., 2007, Prodan et al., 2011, Changuel et al., 2014, Wang et al., 2015, Sujil et al., 2018). Ils sont efficaces en ce sens que la commande de chaque agent est calculée à partir de la connaissance du comportement de l'ensemble du système multi-agents. Cependant, ils sont entièrement dépendants de la robustesse du réseau de communication devant supporter de nombreux échanges entre les agents et le correcteur central. Le correcteur doit lui-même être suffisamment puissant pour calculer les signaux de commande désirés, la complexité du problème pouvant augmenter très rapidement avec le nombre d'agents composant le système. Du fait des limitations des stratégies centralisées, des stratégies *décentralisées* et *distribuées* ont vu le jour.

Dans le cas d'un algorithme de commande *décentralisé*, chaque agent calcule son propre signal de commande à partir de la connaissance de son état et d'informations partielles sur l'état du système multi-agents obtenues auprès d'une entité centrale communiquant avec tous les agents. Ces stratégies sont utilisées pour de nombreuses applications telles que la commande de *microgrids* (Liu et al., 2014), de *smart grids* (Lu et al., 2011, Ayar et al., 2017), le suivi de trajectoire pour des robots mobiles (Prodan, 2012, Angelini et al., 2018) ou l'évitement de collisions (Verginis and Dimarogonas, 2019), cette liste n'étant évidemment pas exhaustive. Ce type de stratégie fournit des lois de commande relativement extensible permettant d'asservir

de nombreux agents. Toutefois, du fait des échanges limités entre les agents, atteindre un objectif coopératif se trouve être plus complexe qu'avec une solution centralisée.

Un algorithme de commande *distribué* est similaire à un algorithme *décentralisé* en ce sens que chaque agent calcule son propre signal de commande. Toutefois, un algorithme *distribué* verra les agents échanger des informations entre eux, participant au calcul du signal d'entrée, sans passer par une entité centrale. Une telle stratégie de commande se trouve à mi-chemin entre une stratégie de commande centralisée et une stratégie décentralisée. En effet, la charge de travail (en termes de calcul) de chaque agent se trouve augmentée par rapport à une architecture décentralisée, mais toujours inférieure à la charge de travail du correcteur central dans le cas d'une architecture centralisée. De plus, chaque agent dispose de plus d'informations sur l'état du système multi-agents que dans le cas décentralisé, lui permettant d'atteindre plus facilement un objectif coopératif. En revanche, il n'a toujours pas la connaissance complète du système que l'on peut atteindre avec une architecture centralisée. Enfin, les stratégies décentralisées sont plus robustes que les deux autres à la perte d'une partie des communications. Plusieurs productions scientifiques de la dernière décennie proposent un état de l'art des stratégies de commande distribuée existantes (Scattolini, 2009, Cao et al., 2012, Maestre and Negenborn, 2014, Rossi et al., 2018).

Mode de calcul du signal de commande

Une autre façon de classifier les méthodes de commande de systèmes multi-agents se fonde sur le type de problème résolu pour obtenir le signal d'entrée appliqué au système. Les classes obtenues sont nombreuses et l'objectif ici n'est pas d'en produire une liste exhaustive mais seulement de présenter quelques grandes catégories utilisées pour la commande de systèmes multi-agents.

L'une des typologies de méthodes de commande les plus communes pour les applications multi-agents est le *consensus* (Olfati-Saber and Murray, 2004, Ren and Beard, 2008), fondé sur la *théorie des graphes*. En effet, le réseau de communication inter-agents peut être vu comme un graphe dont les agents sont les sommets. Le signal de commande de chaque agent sera alors calculé sur la base des connexions le reliant au reste du réseau (Sorensen and Ren, 2006, Flores-Palmeros et al., 2019).

En ce qui concerne les problèmes de navigation, une approche trouvée fréquemment dans la littérature (Hagelbäck and Johansson, 2008, Prodan, 2012, Ivić et al., 2016, Baillard et al., 2018) utilise des *champs de potentiel*. L'objectif du système multi-agents sera alors représenté par un potentiel attractif alors que les obstacles seront représentés par des potentiels répulsifs. Ainsi, les agents se déplaceront en direction de leur objectif tout en évitant les obstacles.

Pour des applications telles que les réseaux de circulation (Chanfreut et al., 2020), les réseaux d'irrigation (Fele et al., 2014) ou les systèmes de grande échelle plus généraux (Fele et al., 2018), la stratégie de commande peut être fondée sur la *théorie des jeux*. Dans ce cas, les agents sont des joueurs, soit des entités prenant des décisions intelligentes et rationnelles, évoluant dans un système de règles régissant leur comportement (Osborne and Rubinstein, 1994). L'ensemble des règles correspond alors aux contraintes s'appliquant sur le système et chaque joueur cherche à maximiser ses gains par les décisions qu'il prend.

Enfin, l'une des classes de stratégies de commande les plus populaires des deux dernières décennies est fondée sur l'*optimisation*. Le signal de commande est obtenu en

résolvant un problème d'optimisation, sous contraintes ou non. Ce type de stratégies peut prendre plusieurs formes telle que la commande optimale (Ji et al., 2006, Movric and Lewis, 2013, Yuan et al., 2018), la commande par réseaux de neurones (Hou et al., 2009, Wen et al., 2016, Yu et al., 2020) ou la commande prédictive qui a connu un essor considérable au cours des vingt dernières années. La popularité de la commande prédictive vient de sa capacité à résoudre des problèmes de commande sous contraintes (Mayne et al., 2000). Les propriétés de cette stratégie de commande ont été abondamment étudiées et documentées dans la littérature (Maciejowski, 2002, Camacho and Bordons, 2007, Rawlings and Mayne, 2009, Kouvaritakis and Cannon, 2016) et appliquées dans de nombreuses applications multi-agents (Maestre and Negenborn, 2014, Olaru et al., 2015). De plus, les stratégies de commande prédictive classiques présentées par Maciejowski (2002) ou Rawlings and Mayne (2009) ont été étendues pour traiter des problèmes complexes. Ces extensions ont par exemple donné naissance à la *commande prédictive explicite* (Bemporad et al., 2002, Tøndel et al., 2003) ou à la *commande prédictive robuste* lorsque le système est soumis à des perturbations déterministes bornées (Mayne et al., 2005, 2006, Kouvaritakis and Cannon, 2016) ou à des perturbations stochastiques (Cannon et al., 2010, 2012, Kouvaritakis and Cannon, 2016).

Commande pour le déploiement

Lorsqu'un système multi-agents sera composé de véhicules autonomes (drones, robots mobiles, véhicules de surface autonomes, etc.), le système est dit multi-véhicules. Ce type de système multi-agents est utilisé pour de nombreuses applications telles que le contrôle de feux de forêt (Merino et al., 2012, Yuan et al., 2019) ou de ressources (Laliberte and Rango, 2009, Jin and Tang, 2010, d'Oleire Oltmanns et al., 2012), la cartographie (Nex and Remondino, 2014, Han and Chen, 2014, Torres et al., 2016) ou la surveillance (Li et al., 2019, Trujillo et al., 2019). Dans la plupart de ces applications, le système multi-véhicules a pour objectif de maximiser la couverture d'une zone donnée en étant potentiellement soumis à des contraintes opérationnelles. Afin d'obtenir une telle couverture, une solution pour le système est de permettre aux véhicules de se répartir en une configuration fixe rendant possible leur mission dans l'environnement à l'intérieur duquel ils évoluent (Schwager et al., 2011).

Au cours des deux dernières décennies, plusieurs algorithmes pour le déploiement autonome d'un système multi-véhicules ont été proposés en se fondant sur des algorithmes de planification de mouvement (Choset, 2001), à base de champs de potentiel (Howard et al., 2002) ou à base de partitions de Voronoï (Cortés et al., 2004, Nguyen, 2016, Hatleskog, 2018). La partition de Voronoï est un objet mathématique introduit par Dirichlet (1850) et approfondi par Voronoï (1908). Il permet de découper un espace métrique muni d'une distance en un ensemble de cellules ne se chevauchant pas et appelées *cellules de Voronoï*. Les cellules sont générées à partir d'un ensemble de points de l'espace, appelés *germes*, et une cellule est obtenue comme la portion de l'espace se trouvant plus près d'un germe que de tous les autres. La *partition de Voronoï classique* est obtenue en utilisant la norme euclidienne comme distance, les germes étant des points de l'espace de travail, mais d'autres distances (Aurenhammer, 1991) ainsi que des germes non ponctuels (Sugihara, 1993, Choset and Burdick, 1995) peuvent être utilisés pour générer une partition de Voronoï.

Dans le cas du déploiement d'un système multi-véhicules, à un instant donné, les germes sont les positions des véhicules. L'objectif du système est alors de se déployer

en une configuration statique où la position de chaque véhicule est confondue avec un point remarquable de la cellule de Voronoï dont le véhicule est le germe. L'une des principales difficultés vient du fait que la partition de Voronoï varie dans le temps, les germes étant mobiles, et que la configuration finale n'est pas connue *a priori*. En se basant sur les travaux de Lloyd (1982), Cortés et al. (2004) proposent un algorithme de commande optimale décentralisé pour le déploiement d'agents ayant une dynamique simple intégrateur. Dans ces travaux, les agents cherchent à rejoindre le centre de masse de leur cellule de Voronoï. Plusieurs algorithmes proposés dans la littérature (Schwager et al., 2011, Song et al., 2013, Moarref and Rodrigues, 2014) se fondent également sur une configuration où les agents reposent sur le centre de masse de leur cellule de Voronoï. Toutefois, ces dernières années, des travaux se fondant sur un autre point remarquable de la cellule, le *centre de Tchebychev*, ont émergé (Nguyen, 2016, Hatleskog, 2018). L'avantage du centre de Tchebychev est qu'il est en général plus simple à obtenir que le centre de masse puisqu'il est solution d'un problème d'optimisation linéaire, là où le centre de masse est obtenu comme un rapport d'intégrales de surface ou de volume. Enfin, Papatheodorou et al. (2016, 2017), Tzes et al. (2018) et Turanli and Temeltas (2020) étudient des algorithmes pour le déploiement de systèmes multi-véhicules lorsque la position de chaque véhicule est incertaine.

Contributions

Sur la base de ce qui a été introduit plus haut, la présente thèse propose différents algorithmes de commande prédictive décentralisés pour le déploiement d'un système multi-agents dans une zone convexe et bornée du plan. Ces algorithmes se basent sur une partition de Voronoï de la zone à couvrir et utilisent le centre de Tchebychev de ces cellules comme objectif de déploiement pour les agents. Les paragraphes ci-dessous détaillent les contributions de la thèse pour chacun des algorithmes proposés.

Déploiement dans le cas nominal

Nguyen (2016) introduit un correcteur prédictif décentralisé pour le déploiement d'un système multi-agents dans une zone convexe et bornée du plan dans le cas nominal (c'est-à-dire lorsque les agents ne sont sujet à aucun défaut, incertitude ou perturbation) avec lequel chaque agent est conduit vers le centre de Tchebychev de sa cellule de Voronoï. Hatleskog (2018), quant à lui, donne une preuve de convergence d'un algorithme de déploiement similaire pour des agents obéissant à une dynamique simple intégrateur, les agents étant asservis par un correcteur par retour d'état sans contraintes.

Suite à ces travaux, la présente thèse propose un correcteur prédictif centralisé à base de cellules de Voronoï pour le déploiement d'un système multi-agents dans une zone convexe et bornée du plan, chaque agent se dirigeant vers le centre de Tchebychev de sa cellule. Après avoir formulé le pendant décentralisé de l'algorithme de déploiement susmentionné, une preuve de faisabilité du problème d'optimisation utilisé par le correcteur prédictif dans le cas où les agents obéissent à une dynamique simple intégrateur est détaillée. La preuve de faisabilité mène à une discussion sur la convergence de l'algorithme de déploiement décentralisé. Pour montrer l'efficacité dudit algorithme, le correcteur prédictif est utilisé comme correcteur de position pour

une flotte de drones quadrirotors, pour lesquels un modèle d'évolution non linéaire est utilisé. Des pistes pour la preuve de convergence de l'algorithme de déploiement pour une flotte de drones quadrirotors sont ensuite introduites.

Ces contributions ont mené à la publication des articles Chevet et al. (2018) et Chevet et al. (2020b).

Déploiement sous perturbations déterministes bornées

Papatheodorou et al. (2016) et Papatheodorou et al. (2017) étudient le déploiement d'un système où les agents sont soumis à une incertitude bornée sur la mesure de position. Ces travaux se fondent sur la définition d'une partition de Voronoï garantie à base d'ellipsoïdes pour laquelle les germes ne sont plus des points mais des disques.

Dans la même idée, la présente thèse introduit une nouvelle partition de Voronoï garantie à base de boîtes où les germes de la partition de Voronoï ne sont plus des points mais des boîtes (ou des rectangles lorsque l'espace est ramené au plan). Une telle partition garantie est utilisée dans le cas où les agents sont soumis à des perturbations bornées sur leur mesure de position.

Pour traiter le cas de perturbations déterministes bornées agissant sur un système, une stratégie de commande prédictive à base de tubes (Mayne et al., 2005) peut être utilisée. L'idée régissant un tel algorithme de commande est la séparation du signal d'entrée appliqué au système en deux parties : l'une obtenue par résolution d'un problème d'optimisation sur la dynamique nominale et l'autre par multiplication de l'écart entre l'état réel et l'état nominal du système par un gain matriciel. Il est alors garanti que l'état du système appartient à un ensemble, positivement invariant de manière robuste pour la dynamique de l'écart évoqué précédemment, centré sur l'état nominal du système. Alvarado (2007) propose un problème d'optimisation sous contraintes d'inégalités matricielles linéaires pour obtenir la matrice de gain utilisée pour un correcteur prédictif à base de tubes classique (Mayne et al., 2005).

Ainsi, la présente thèse propose un correcteur prédictif décentralisé à base de tubes avec observateur pour le déploiement d'un système où les agents sont soumis à des perturbations d'entrée et de sortie déterministes et bornées. Une nouvelle procédure d'optimisation sous contraintes d'inégalités matricielles linéaires/bilinéaires est conçue pour obtenir les gains matriciels du retour d'état et de l'observateur nécessaires pour la stratégie de commande à base de tubes avec observateur. Cette procédure minimise la taille des ensembles positivement invariants de manière robuste pour les dynamiques d'erreur auxquels l'état du système appartient. Cela permet de laisser suffisamment de degrés de liberté au correcteur prédictif pour optimiser la performance du système. L'algorithme de déploiement décentralisé est ensuite appliqué, dans un premier temps, à un système composé d'agents obéissant à une dynamique simple intégrateur, puis pour la commande en position d'une flotte de drones quadrirotors.

Ces contributions ont mené à la publication de l'article Chevet et al. (2019).

Déploiement sous perturbations stochastiques non bornées

Lorsqu'un système est soumis à des perturbations stochastiques, bornées ou non, des contraintes probabilistes apparaissent dans le problème d'optimisation résolu par le correcteur prédictif. Gavilan et al. (2012) proposent un correcteur prédictif sous contraintes probabilistes pour un système pour lequel les contraintes doivent être

satisfaites avec une probabilité égale à 1. À partir des propriétés stochastiques de la perturbation, Gavilan et al. (2012) introduisent une méthode pour transformer les contraintes probabilistes en contraintes algébriques.

Ainsi, la présente thèse introduit un nouveau correcteur prédictif décentralisé sous contraintes probabilistes avec observateur pour le déploiement d'un système où les agents sont soumis à des perturbations d'entrée et de sortie stochastiques et non bornées. Toutefois, la perturbation stochastique, dont l'évolution est inconnue sur l'horizon de prédiction, apparaît dans les contraintes du correcteur prédictif. Ces contraintes doivent être respectées avec une probabilité égale à 1. Une procédure pour transformer ces contraintes probabilistes en contraintes algébriques, pour pouvoir résoudre le problème d'optimisation du correcteur, est conçue, accompagnée d'une preuve de faisabilité. Cette procédure se fonde sur les propriétés stochastiques des signaux de perturbation et cherche à trouver une borne permettant aux contraintes d'être respectées pour presque toutes les perturbations agissant sur le système. L'algorithme de déploiement décentralisé est ensuite appliqué, dans un premier temps, à un système composé d'agents obéissant à une dynamique simple intégrateur, puis, pour la commande en position d'une flotte de drones quadrirotors.

Ces contributions ont mené à la publication de l'article Chevet et al. (2020a).

Reconfiguration dans le cas d'un système multi-véhicules variant dans le temps

Une autre contribution de cette thèse porte sur la reconfiguration d'un système multi-agents se déployant pendant que des agents rejoignent ou quittent la zone de déploiement. Ce problème se rapproche du domaine de la commande de formation tolérante aux défauts étant donné que, bien souvent, les agents devant quitter la zone de déploiement le font parce qu'ils sont défectueux.

Dans le cas d'agents rejoignant la zone de déploiement, l'algorithme proposé dans cette thèse est une extension naturelle de l'algorithme de déploiement dans le cas nominal. En effet, cet algorithme étant décentralisé, il est aisément extensible et permet, par construction, d'ajouter de nouveaux agents à l'objectif de déploiement.

Dans le cas d'agents quittant la zone de déploiement, la présente thèse introduit deux nouvelles stratégies se fondant sur un objectif transitoire, différent du centre de Tchebychev de la cellule de Voronoï, pour les agents restant dans la zone de déploiement, leur permettant d'éviter la trajectoire des agents quittant la zone. Dans la première stratégie, l'objectif transitoire d'un agent est le barycentre des positions pondérées de ses voisins. Avec la seconde stratégie, l'objectif transitoire est obtenu par résolution d'un problème d'optimisation contraignant cet objectif à l'intérieur d'une région de la zone de déploiement dans laquelle l'agent peut évoluer de manière sûre, sans risquer de collisions avec les agents quittant l'espace de travail. Cette stratégie a pour avantage de permettre une reconfiguration du système multi-agents lorsque plusieurs agents quittent la zone de déploiement en même temps.

Pour le cas où des agents intègrent la zone de déploiement, des résultats de simulation sont présentés pour un système où les agents obéissent à une dynamique simple intégrateur. Pour le cas d'agents quittant la zone de déploiement, les deux stratégies de reconfiguration sont comparées à l'aide d'un système composé d'agents obéissant à une dynamique simple intégrateur, un agent quittant la zone au cours du déploiement. Enfin, des résultats de simulation sont présentés pour la reconfiguration

d'un système multi-agents lorsque deux agents quittent la zone de déploiement, d'abord avec un système composé d'agents obéissant à une dynamique simple intégrateur, puis avec une flotte de drones quadrirotors.

Ces contributions ont mené à la publication des articles Chevet et al. (2018) et Chevet et al. (2020b).

Chapter 1

Introduction

Table of Contents

1.1 Context and motivations

1.1.1 What is a multi-agent system?

The notion of *Multi-Agent System* (MAS) takes its roots in the computer science community. In the introduction of his book, Wooldridge (2009) gives five main trends that lead the research and development in computing:

- *Ubiquity*: the capability to introduce processing power into devices where it was once thought to be impossible due to both technical and economic restrictions;
- *Interconnection*: the networking of devices empowered with processing capabilities to interact and communicate in a complex topology;
- *Intelligence*: the ability of computing devices to perform increasingly complex tasks;
- *Delegation*: the possibility to trust and have confidence in the computing devices when performing safety critical tasks;

- *Human-orientation*: the interaction between humans and computing devices rendered more natural, as it would be between two humans, by using concepts and metaphors instead of machine code.

These trends have led to the emergence of MAS. However, as stated by Wooldridge and Jennings (1995), the question "what is an agent?" is not easy to answer. Indeed, despite the concept being used nowadays in a wide spectrum of topics, there is no unified definition of the concept of *agent*. This is why, based on a list of properties that an agent has to possess, Wooldridge and Jennings (1995) propose a definition further improved in Wooldridge (2009) and that is the one accepted throughout the present thesis.

Definition 1.1: Agent (Wooldridge, 2009)

An *agent* is a computer system that is situated in some *environment*, and that is capable of *autonomous* actions in this environment in order to meet its design objectives.

In their work, Wooldridge and Jennings (1995) present four properties associated with Definition 1.1 that an agent has to obey. However, these properties can be reorganized into five intertwined requirements:

(i) *Autonomy*: an agent is able to evolve without the direct intervention of a human operator;

(ii) *Perception*: an agent is able, through sensors of very different natures, to perceive the changes in its environment;

(iii) *Interaction*: an agent is able to communicate with other agents present in the environment;

(iv) *Reactivity*: an agent is able to respond to changes in its environment when necessary;

(v) *Pro-activeness*: an agent is able to take the initiative to fulfill its mission.

These requirements are made possible thanks to the main trends that have led research and developments in computer science since its birth. Then, it is possible to formulate a definition of a MAS.

Definition 1.2: Multi-agent system

A *multi-agent system* is a set of agents meeting the requirements (i)-(v).

Apart from computer science and artificial intelligence (Weiss, 1999, Wooldridge, 2009, Wang et al., 2016b, Grzonka et al., 2018), the notion of MAS is of interest for several other scientific fields. MAS are used for example in economy (Čech et al., 2013, Malakhov et al., 2017, Herzog et al., 2017, Han et al., 2017, Gao et al., 2018), social sciences (Davidsson, 2002, Li et al., 2008, Serrano and Iglesias, 2016, Ramírez et al., 2020), biology (Couzin et al., 2005, Roche et al., 2008, Ren et al., 2008, Colosimo, 2018), computer science (Ferber and Gutknecht, 1998, DeLoach et al., 2001, Bellifemine et al., 2007, Calvaresi et al., 2019) or control engineering (Hespanha et al., 2007, Bullo et al., 2009, Dimarogonas et al., 2011, Maestre and

Negenborn, 2014). This field of engineering is of interest for the remainder of this thesis. In the last two decades, an increasing number of control applications where the system can be decomposed into several subsystems, each of them meeting the requirements (i)-(v), have appeared. The systems studied in such control applications are thus covered by the scope of Definition 1.2 and are then dubbed MAS while each subsystem is an individual agent. The main difference with computer science is that in control applications, each agent obeys a given *dynamical equation* in addition to the five previous requirements. This is why, in control engineering literature, multi-agent systems are sometimes referred to as dynamical MAS. Then, in this thesis, MAS designates a dynamical multi-agent system where each agent meets the requirements (i)-(v) and obeys a dynamical equation.

Plenty of multi-agent systems are actively studied in control engineering such as smart grids (Logenthiran et al., 2012, Radhakrishnan and Srinivasan, 2016, Singh et al., 2017) and microgrids (Dimeas and Hatziargyriou, 2005, Minchala-Avila et al., 2015), water distribution networks (Wang et al., 2017, Shahdany et al., 2019), traffic (Lin et al., 2012, Chanfreut et al., 2020) and transportation networks (Negenborn et al., 2008, Larsen et al., 2020), mobile sensor networks (Cortés et al., 2004), multi-robot or multi-vehicle formation (D'Andrea and Dullerud, 2003, Wurman et al., 2008, Bullo et al., 2009, Prodan et al., 2011, Alonso-Mora et al., 2015, Kamel et al., 2020), vehicle platooning (Ploeg et al., 2013, Turri et al., 2016, Van De Hoef et al., 2017) and deployment control (Schwager et al., 2011, Nguyen and Stoica Maniu, 2016, Papatheodorou et al., 2017). For all these kinds of MAS, the control engineering approach consists in the supervision of the interactions between the agents as well as the development of a control strategy to lead the agents towards a common goal.

As always in control engineering, as pointed out by Nguyen (2016), each application has its own characteristics that lead to different research directions. Indeed, the definition of a MAS has several degrees of freedom such as the nature of the dynamics (e.g. linearity, time-invariance, etc.), the nature of the communications between the agents or the physical constraints on the system such as actuator limitations or safety constraints. Then, the following paragraphs give several examples of MAS control strategies based on different criteria.

1.1.2 An overview of multi-agent system control strategies

In order to control the agents of the MAS, it is necessary to design control algorithms to obtain the input signal of each agent in order to achieve a common goal. There exist different classes of control algorithms for multi-agent systems. These classes depend on various parameters such as the communication topology between the agents or the way the control input is computed. In the following are described two ways to classify the control algorithms to give an overview of the existing control techniques used for MAS.

1.1.2.1 Communication topologies

As discussed at the end of Section 1.1.1, communications are one of the critical issues when dealing with MAS. Based on the communication topology between the agents, the MAS control strategies are often divided in three classes: *centralized*, *decentralized* and *distributed* strategies (Tanenbaum and Van Steen, 2007).

1.1.2.1.1 Centralized topology In *centralized control*, all the agents know the control policy of the other agents to compute its own. To do so, the information of all agents is sent to a central controller C, this central controller being either an agent of the MAS or an additional computer. Then, based on the agents' information, the central controller C computes the control input of each agent and sends it back to the corresponding agent. Figure 1.1 gives an example of centralized architecture for a MAS composed of N agents. Each agent of the MAS sends information, e.g. its state vector $\mathbf{x}_i(k)$, with $i \in \overline{1, N}$, in the case of Figure 1.1, to the controller C which computes the control inputs to apply to the agents of the MAS and sends the individual control inputs $\mathbf{u}_i(k)$, with $i \in \overline{1, N}$, back to each agent.

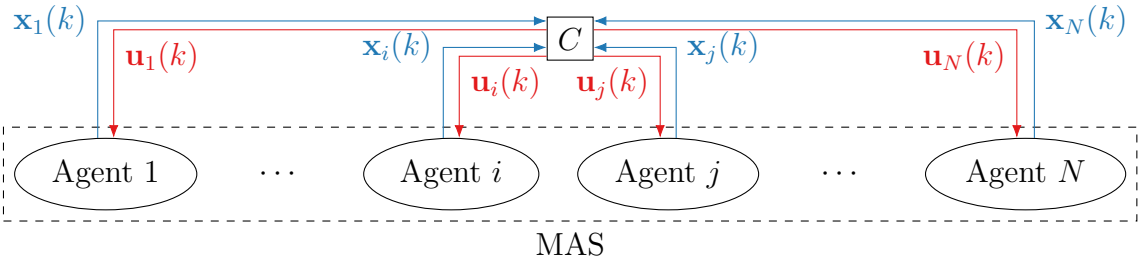


Figure 1.1: Centralized control architecture

Such centralized strategies have been extensively studied for the control of multi-agent systems in the past two decades (Xu and Hespanha, 2006, Olfati-Saber et al., 2007, Prodan et al., 2011, Changuel et al., 2014, Wang et al., 2015, Sujil et al., 2018). These strategies are efficient in the sense that the control input for each agent of the MAS is computed with full knowledge of the multi-agent system's behavior. However, to obtain the control inputs for all the agents of the MAS, the central controller C needs, at all times, full knowledge of each agent of the MAS and it also has to be able, at all times, to send the control input to each agent of the MAS. Due to this property, the MAS is heavily dependent on the communication between the agents and the central controller and is highly sensitive to any communication fault. Moreover, an increasing number of agents implies an increasing number of communications, leading potentially to network saturation, and a control problem of increasing size which can be computationally time-consuming depending on the considered strategy, and therefore not applicable in real-time systems.

Depending on the considered application, such a dependence on the central controller can be unsuitable. To overcome these limitations, *decentralized* and *distributed* control policies have been developed.

1.1.2.1.2 Decentralized topology In a *decentralized control* structure, each agent computes its own control policy with the knowledge of its own state while exchanging information with a central entity. Figure 1.2 gives an example of decentralized architecture for a multi-agent system composed of N agents. Each agent of the MAS can exchange information (e.g. an aggregated state vector $\mathbf{x}(k)$ containing all or part of the state vectors $\mathbf{x}_i(k)$ of each agent $i \in \overline{1, N}$ in the case of Figure 1.2) with a central entity C. However, each agent computes its input signal by sending information (e.g. its state vector $\mathbf{x}_i(k)$, with $i \in \overline{1, N}$, and the aggregated vector $\mathbf{x}(k)$ in the case of Figure 1.2) to a local controller C_i, with $i \in \overline{1, N}$, which sends back the control input $\mathbf{u}_i(k)$ to agent i.

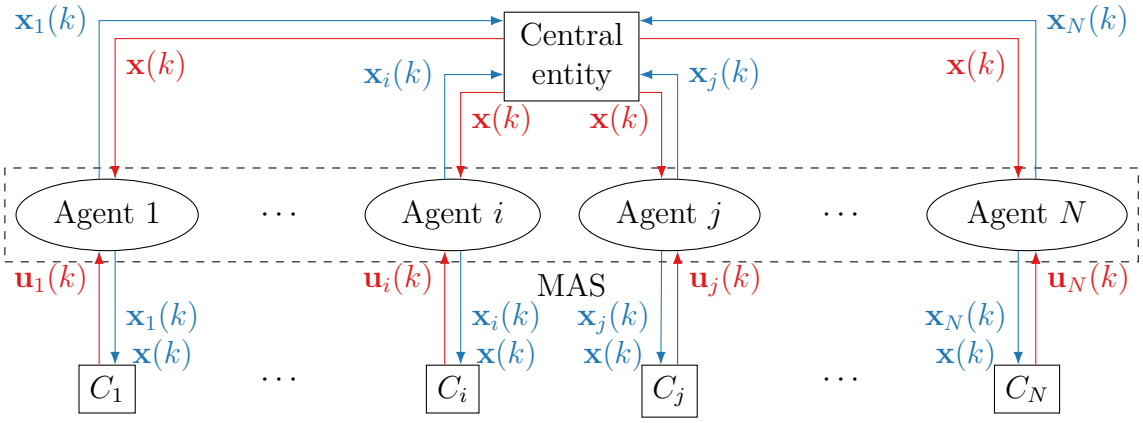


Figure 1.2: Decentralized control architecture

One of the most used decentralized strategy in control of multi-vehicle systems is the leader-follower approach where the central entity is another agent and several structures of the kind presented in Figure 1.2 can be cascaded. This approach is mainly used in formation control (Consolini et al., 2008, Liu et al., 2016, Wang et al., 2019) of MAS where the agents are mobile robots or vehicles. In a more general framework, decentralized control strategies have been developed for numerous control applications such as microgrids (Liu et al., 2014), smart grids (Lu et al., 2011, Ayar et al., 2017), trajectory tracking for mobile robots (Prodan, 2012, Angelini et al., 2018) or collision avoidance (Verginis and Dimarogonas, 2019).

A decentralized control strategy is easily scalable compared to a centralized strategy since the addition of new agents to the MAS does not increase the complexity of a central controller, each agent obtaining its own input with a local controller. A decentralized structure is also resilient to the loss of the central entity since each agent is able to either continue its mission by itself or reach a safe state by using its local controller.

However, this comes with a drawback linked to the limited communications between the agents since such communications have to be done through the central entity. Indeed, the control input of an agent being computed solely from the information of this agent, cooperative objectives can be more difficult to carry out. The example of a fleet of several vehicles following a given trajectory while maintaining a given formation is considered to illustrate this element. A common strategy is for each vehicle to follow a trajectory provided by the formation leader such that if every vehicle follows its trajectory, the formation is maintained. If a perturbation or a fault occurs on one or several vehicles, their trajectory can be modified. However, due to the absence of communications, a vehicle does not know its relative position to another vehicle. Then, without additional strategies to mitigate the perturbation or the fault, there is a potential risk of collision between two or more vehicles which is a hindrance to cooperation. Some strategies in the literature seek to guarantee a certain level of interaction between the agents to attain global coordination of the MAS such as the one proposed by Morărescu and Fiacchini (2016).

1.1.2.1.3 Distributed topology With a *distributed control* strategy, similarly to the decentralized case, each agent computes its own control policy with the

knowledge of its own state. However, the agents are able to exchange information with all or part of the agents belonging to the MAS to obtain its control input. Figure 1.3 gives an example of distributed architecture for a MAS composed of N agents. Each agent computes its control input $\mathbf{u}_i(k)$, with $i \in \overline{1, N}$, with a local controller C_i. However, contrary to the decentralized case, each controller C_i is able to exchange information with the controllers C_j of the other agents. In the example of Figure 1.3, each controller exchanges information with its neighbors, the nature of the information not being reported here. It can be noticed that such a structure is an example among others and more connections between the local controllers could be added.

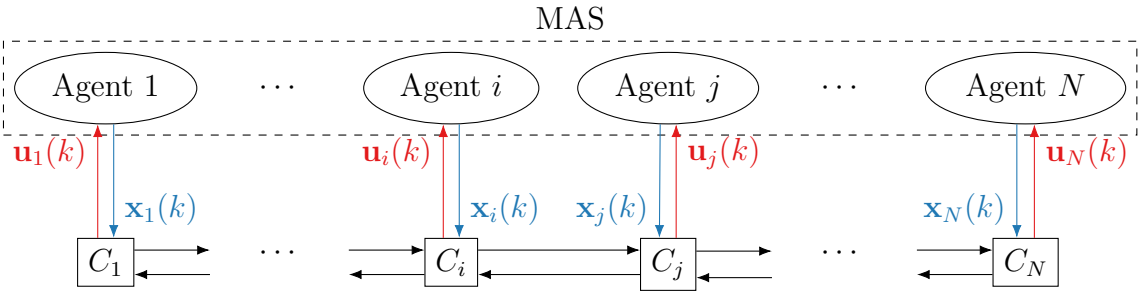


Figure 1.3: Distributed control architecture

The distributed strategies bridge the gap between centralized and decentralized strategies. Indeed, they involve more communications than a decentralized control strategy but less than a centralized one. The computational load delegated to each agent is heavier than with a decentralized strategy but the complexity of the control problem solved by each local controller is reduced compared to the complexity of the problem solved by the central controller of a centralized architecture. Moreover, a distributed strategy, as a decentralized strategy, is resilient to the loss of communication with one or several local controllers and it allows to improve the cooperation capabilities of the agents with respect to a decentralized strategy.

Due to these elements and the *ubiquity* trend presented in Section 1.1.1, distributed strategies are attracting increasing attention. Some examples include Olfati-Saber (2006), Ren and Beard (2008) or Cao and Ren (2011) who use distributed consensus strategies or Viel (2017) who studies event-triggered control for MAS. In their book, Maestre and Negenborn (2014) give a in-depth overview of existing distributed model predictive control algorithms. A state of the art of distributed control techniques can be found in either Scattolini (2009), Cao et al. (2012) or Rossi et al. (2018). Such strategies can also be applied to different types of multi-agent systems such as microgrids (Khan et al., 2016, Cominesi et al., 2017), traffic networks (Chanfreut et al., 2020), power grids (Bidram et al., 2014) or multi-vehicle systems (Turri et al., 2016, Li et al., 2017b, Belkadi et al., 2017, 2019).

1.1.2.2 Classification of the approaches for MAS control

An element that goes with the choice of a communication topology for the multi-agent system control structure is the nature of the controller itself, either central or local. Indeed, the control problem solved to obtain the control input of the agents is as important and dependent on the nature of the problem as the communication

topology. Then, the following paragraph does not aim to be exhaustive but to present some of the most widely used methods of the last twenty years to compute the control input in the central or local controllers.

1.1.2.2.1 Graph theory The communication topology used by the agents of a MAS to exchange information can be seen as a graph where the agents are the nodes of the graph and the communication between two agents is a link, directed or not, between the two nodes. Results from algebraic graph theory (Godsil and Royle, 2001) are therefore considered in combination with the dynamics of the agents to compute the control inputs.

The main graph-theoretic strategy that has been extensively applied in multi-agent system control application is the *consensus* (Olfati-Saber and Murray, 2004, Ren and Beard, 2008), also called *agreement* (Mesbahi and Egerstedt, 2010). The name consensus is self-descriptive. Indeed, in the case of consensus problems, the agents aim to synchronize to a common value. Consensus strategies are often used to address issues related to multi-vehicle systems such as rendezvous (Sorensen and Ren, 2006) or formation control (Fax and Murray, 2004, Olfati-Saber, 2006, Flores-Palmeros et al., 2019) and are related to synchronization problems (Ahmadizadeh et al., 2016, Panteley and Loría, 2017).

1.1.2.2.2 Potential field As stated in Cheng et al. (2004), the idea of potential field approaches is based on a physical analogy. Indeed, an electrically charged body produces an electrical vector field that either attracts or repulses other electrically charged bodies. This vector field is the gradient of a scalar function f of a vector variable $\mathbf{x}$. In most physics applications, this vector variable $\mathbf{x}$ is a vector of space coordinates (e.g. Cartesian or spherical coordinates). A potential field approach in control is then based on the construction of an artificial scalar function f which is the sum of an attractive and a repulsive potential. The attractive potential is meant to attract the system towards its desired objective while the repulsive potential takes the system away from a given area of the state space in which it evolves. Then, the negative gradient $-\nabla f(\mathbf{x})$ indicates the most promising direction in the state space to drive the system towards its objective.

Potential field approaches are often used in navigation problems (Hagelbäck and Johansson, 2008, Prodan, 2012, Ivić et al., 2016, Baillard et al., 2018) where the attractive potential is minimal on the objectives that the agents have to reach and maximal on the obstacles such that the agents go towards their objectives while avoiding obstacles. One of the main drawbacks of this approach is that it can drive the system into a configuration corresponding to a local minimum of the potential function. Rimon and Koditschek (1992) provide methods to compute artificial potential functions free of local minima.

1.1.2.2.3 Game theory Game theory (Osborne and Rubinstein, 1994) is a mathematical framework used to describe the behavior of interacting *players*. In such a framework, a player is an intelligent and rational decision maker, i.e. an entity that takes decisions from reasoning, which has to evolve within a set of rules guiding its behavior. Game theory is often used in social sciences to represent human interaction and social phenomena. Moreover, given its characteristics, such a framework is an obvious tool to study MAS control problems. Indeed, the agents are players which

have to reach an objective where they maximize their gain in the game, the rules of the game being the constraints applied on the problem.

Game theoretic methods for MAS control have been increasingly studied in the last decade. For example, Fele et al. (2017) present a game theoretic control framework called coalitional control to deal with various problems. This framework is based on the theory of coalitional games (Myerson, 1991), in which it is assumed that cooperation always brings benefits to the players, but with the limitation that forming new coalitions of agents might provide a lower gain than doing tasks with a larger number of coalitions. Coalitional control has been used in the last years to deal with irrigation canals (Fele et al., 2014), traffic networks (Chanfreut et al., 2020) or more general large-scale systems (Fele et al., 2018). However, coalitional control is just one of the many aspects of game theoretic control for multi-agent systems and numerous other works deal with different game theoretic strategies (Quijano et al., 2017). Such works are for example Semsar-Kazerooni and Khorasani (2009) and Zhang et al. (2014) which mix respectively cooperative and differential game theory with consensus problems or Khan et al. (2016) that use non-cooperative game theory for the control of microgrid systems.

1.1.2.2.4 Optimization Optimization-based control methods compute the control input of a system as the solution of an optimization problem minimizing a given criterion under constraints. Then, using efficient tools such as linear, quadratic, mixed-integer or even semi-definite programming solvers, it is possible to obtain the optimal input driving the agents of a MAS towards their objective. Optimization-based control have been applied to multi-agent systems in different forms such as optimal control (Ji et al., 2006, Movric and Lewis, 2013, Yuan et al., 2018) or neural-network based control (Hou et al., 2009, Wen et al., 2016, Yu et al., 2020), which have been attracting a lot of attention for the past ten years. One of the most widely used optimization-based control method is model predictive control (MPC) thanks to its effectiveness with constraint handling due to the receding horizon policy it adopts (Mayne et al., 2000). The properties of MPC have been extensively studied and are well documented in the literature (Maciejowski, 2002, Camacho and Bordons, 2007, Rawlings and Mayne, 2009, Kouvaritakis and Cannon, 2016). In the case of multi-agent systems, Maestre and Negenborn (2014) and Olaru et al. (2015) present a detailed overview of current uses and applications of MPC strategies.

In addition to the classical MPC as presented in Maciejowski (2002) or Rawlings and Mayne (2009), several different flavors have been developed over the years to deal with increasingly complex situations. For example, an explicit solution to the MPC optimization problem can be found as a piecewise affine function (Bemporad et al., 2002, Tøndel et al., 2003) obtained by exploiting the Karush-Kuhn-Tucker conditions (Boyd and Vandenberghe, 2009). Moreover, robust versions have been developed for the case of a perturbed system (Kouvaritakis and Cannon, 2016), either when the perturbations are deterministic and bounded (Mayne et al., 2005, 2006) or have a stochastic nature (Cannon et al., 2010, 2012), both having been applied on multi-agent systems (Prodan et al., 2011, Trodden and Richards, 2014, Nikou and Dimarogonas, 2019, Lyons et al., 2012, Dai et al., 2016).

1.1.3 Deployment control

Multi-vehicle systems (MVS) are a type of MAS where each agent is an autonomous vehicle. They are used for a wide spectrum of real world missions such as forest fire monitoring (Merino et al., 2012, Yuan et al., 2019), ground and resource monitoring (Laliberte and Rango, 2009, Jin and Tang, 2010, d'Oleire Oltmanns et al., 2012), mapping and modeling (Nex and Remondino, 2014, Han and Chen, 2014, Torres et al., 2016) or even surveillance missions (Li et al., 2019, Trujillo et al., 2019). For most of these applications, a MVS seeks to maximize the coverage of a given spatial area under potential operating constraints. A way to achieve such a coverage is for the MVS to self-deploy within the area it seeks to cover. The works cited above have different ways to define what is a deployment for coverage. The definition that is used for this thesis is the one from Schwager et al. (2011), which considers the deployment of a multi-vehicle system as a strategically appropriate spreading of the MVS in a given environment in order to reach a fixed configuration, the environment being the spatial area that the vehicles aim to cover.

In the last twenty years, several works have studied the self-deployment of a MVS problem such as Choset (2001), Howard et al. (2002), Cortés et al. (2004), Li and Cassandras (2005), Murray (2007), Moarref and Rodrigues (2014), Nguyen (2016), Papatheodorou et al. (2017) or Hatleskog (2018). The deployment of a MAS is meant to ensure the maximal coverage of an environment with respect to a given criterion. In that sense, several techniques have been proposed to deal with this problem such as *motion planning* (Choset, 2001), *potential field* (Howard et al., 2002), *probabilistic* (Li and Cassandras, 2005) or *Voronoi-based techniques* (Cortés et al., 2004). Schwager et al. (2011) show that the last three approaches can be unified through the use of a mixing function encoding the coverage criteria to be maximized. Many of the works cited are based on the use of a time-varying Voronoi tessellation of the deployment area. The *Voronoi tessellation* is a tool that has been introduced by Dirichlet (1850) and further developed by Voronoï (1908). It is a way to partition a metric space, equipped with a distance function, into a set of non-overlapping regions called *Voronoi cells*. These cells are generated from a finite set of elements belonging to the metric space, these elements being called generators. Each cell is associated uniquely with one of the generators. The *classical Voronoi tessellation* is obtained by using the Euclidean norm but Voronoi diagrams can also be obtained from other types of distances (Aurenhammer, 1991) and for generators that are not points but sets (Sugihara, 1993, Choset and Burdick, 1995). Figure 1.4 presents for example the Voronoi tessellation of a square with five generators. The plot on the left is obtained using the Euclidean distance and the one on the right is obtained using the Manhattan distance where, if $\mathbf{x} = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}^\top$ is a vector of $\mathbb{R}^n$, the Euclidean and the Manhattan norm are defined as:

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2} \quad \text{and} \quad \|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|,$$

respectively. Moreover, when the generators are not points but circles or ellipses, the Voronoi tessellation is often called *power Voronoi diagram* (Aurenhammer, 1991).

For the deployment over an area, the positions of the vehicles of the MVS are the generators of the Voronoi tessellation. The goal of the MVS is then to reach a static configuration in which each vehicle lies on a remarkable point of the Voronoi cell

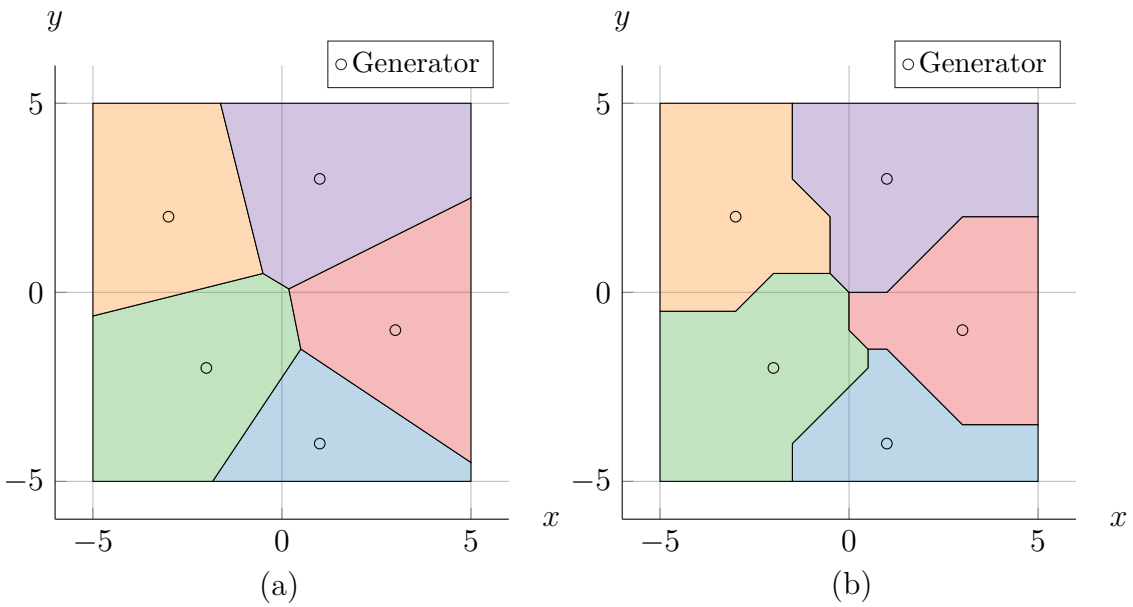


Figure 1.4: Voronoi tessellation of a square for 5 generators with the Euclidean norm (a) and the Manhattan norm (b).

generated by this vehicle. It can be noticed that the main difficulties in achieving the deployment are, first, that, since the generators of the Voronoi tessellation move over time, the Voronoi tessellation is time varying and, second, that the final configuration is not known *a priori* due to the dynamics of the Voronoi tessellation. One of the first approaches to define a Voronoi-based coverage of a given area is the Lloyd's algorithm (Lloyd, 1982), derived from the k-means clustering algorithm (MacQueen, 1967), where the goal is to obtain a maximum coverage from a set of agents without any dynamics. With Lloyd's algorithm, the considered remarkable point of a cell is the center of mass of the cell. Then, if $\mathbf{p}_i(k)$ denotes the position of the agent i at time k and $\mathbf{c}_i(k)$ the center of mass of its Voronoi cell at time k, Lloyd's algorithm runs as presented in Algorithm 1.1. Figure 1.5 presents the result of Lloyd's algorithm for five agents inside a square. The circles represent the positions of the agents at iteration k and the stars represent the centers of mass at iteration k. The first diagram presented in Figure 1.5 is obtained from the initial positions, the second is the one obtained after the first iteration of Lloyd's algorithm and the third one is the final result obtained after 26 iterations.

Algorithm 1.1: Lloyd's algorithm.

Input: The initial positions of the agents $\mathbf{p}_i(0)$, with $i \in \overline{1, N}$
while $\mathbf{c}_i(k+1) \neq \mathbf{c}_i(k)$ *for all* i **do**
  Compute the Voronoi tessellation of the area from the positions $\mathbf{p}_i(k)$;
  Compute the center of mass $\mathbf{c}_i(k)$ of each Voronoi cell;
  Set $\mathbf{p}_i(k+1) = \mathbf{c}_i(k)$;
end
Output: The static configuration of the MAS

Recent work on the deployment of a MAS starts with the deployment of a multi-sensor network in a convex two-dimensional area. In their paper, Cortés

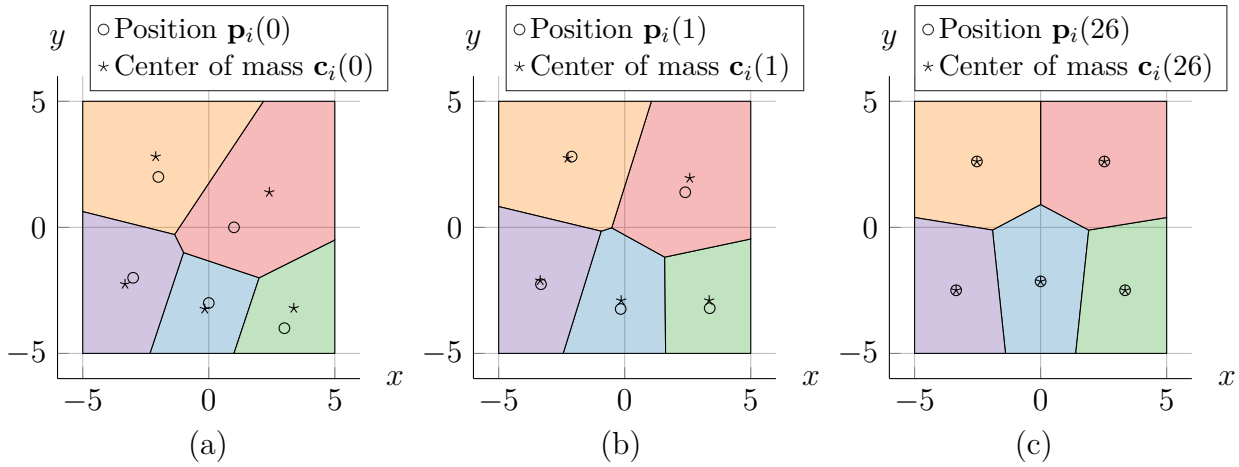


Figure 1.5: Illustration of Lloyd's algorithm at initialization, after the first iteration and after the last iteration.

et al. (2004) propose a decentralized optimal control algorithm for agents obeying single integrator dynamics such that the MAS reaches a static *centroidal Voronoi configuration*[1]. Based on this work, Sharifi et al. (2014) propose a distributed control algorithm based on feedback linearization for the deployment into a centroidal configuration in a so-called *weighted Voronoi tessellation*, where the distance between two points is the Euclidean norm of the difference between their coordinates divided by a given positive weight. Several other works propose control algorithms to drive a MAS into a static configuration under constraints such as maximization of the coverage (Schwager et al., 2011) or energy efficiency constraints (Song et al., 2013, Moarref and Rodrigues, 2014) and even control strategies for MAS subject to drifts (Bakolas and Tsiotras, 2013). In the last years, Nguyen (2016) proposed a new deployment strategy where the remarkable point considered for the Voronoi cells is not the center of mass of the cell, but the Chebyshev center (Boyd and Vandenberghe, 2009), i.e. the center of the largest ball contained in the Voronoi cell. This point is obtained as the solution of a linear optimization problem and is often easier to compute than the center of mass. Nguyen (2016) proposes state-feedback strategies, including model predictive control, for the Voronoi-based deployment of a MAS using the Chebyshev center of the Voronoi cells. This work has been further studied in Hatleskog (2018) who provides a proof of convergence of the deployment strategy in the one dimensional case and in the two dimensional case when the agents obey single integrator dynamics and are controlled with an unconstrained state-feedback controller. In addition, the problem of optimal coverage of an area when the location of the agent is uncertain has been recently studied. Papatheodorou et al. (2016) propose for example a distributed optimal control strategy for the deployment of a multi-sensor network for which the position of the agent is uncertain and contained inside a circle. These results have been further extended in Papatheodorou et al. (2017), Tzes et al. (2018) or Turanli and Temeltas (2020).

[1] *Centroidal configuration* means that the agents lie on the center of mass of their Voronoi cells.

1.1.4 Main motivations and thesis orientation: Model Predictive Control for the deployment of multi-vehicle systems

From the literature overview of Sections 1.1.1 and 1.1.2, it appears that the control of multi-agent systems is a wide and active research topic. Thus, the present thesis is limited to a given area that is overviewed in Section 1.1.3. Indeed, deployment control of a multi-vehicle system is a task that is relevant for a wide range of applications as detailed previously.

As presented in Section 1.1.3, several control strategies exist for the deployment of a MAS using a Voronoi tessellation of the area in which the MAS is deployed. However, the majority of these works are based on the center of mass of the Voronoi cells to drive the agents towards a static configuration. Then, building on the results of Nguyen and Stoica Maniu (2016) and Nguyen et al. (2017), the present thesis deals with the deployment and reconfiguration of a multi-vehicle system inside a bounded convex two-dimensional area.

The goal of the present work is to develop model predictive control algorithms for the Voronoi-based deployment of a multi-vehicle system in a bounded convex two-dimensional area and to apply the proposed control algorithms for the deployment of a fleet of quadrotor unmanned aerial vehicles (UAVs). The first step for such work is to adapt the decentralized model predictive controller proposed in Nguyen (2016) for the deployment of agents obeying simple dynamics such as single or double integrator dynamics. Thus, a decentralized deployment algorithm is proposed and applied to a fleet of quadrotor UAVs obeying continuous-time nonlinear dynamics. To this end, a cascaded control architecture is considered, where the decentralized MPC algorithm is used for the position control of the vehicles, the position control design being performed based on simplified models of the agents having double integrator dynamics. The goal of the multi-vehicle system is then to deploy into a static configuration in which each agent lies on the Chebyshev center of its associated Voronoi cell. The use of MPC is justified in this context since it allows to consider constraints on the future evolution of the system, limiting it to evolve inside its Voronoi cell, used as a safety region to avoid collision with the other vehicles, and restraining the speed and attitude of the system due to physical limitations. Moreover, such a framework is favorable towards the use of a decentralized control algorithm. Indeed, with just the knowledge of the other agents' positions by measurements or communication with a central entity, an agent is able to build its own Voronoi cell, allowing it to then carry the burden of control input computation.

Nguyen (2016) or Hatleskog (2018) study the deployment of a MAS in the nominal case. Papatheodorou et al. (2016), for example, opens the door on the deployment of MAS subject to position uncertainty. Further investigations have to be led in this direction to allow a system subject to more general perturbations to be deployed over a given area. Robust MPC techniques have been developed to account for bounded perturbations on a single system, for example in Mayne et al. (2005) and Mayne et al. (2006). Then, based on these works, the present thesis studies the deployment of a MAS when the agents are subject to *bounded deterministic perturbations* and proposes a decentralized tube-based MPC algorithm for the Voronoi-based deployment of a fleet of UAVs which drives the MAS into a static configuration, guaranteeing that the distance between the agents and their

Chebyshev center is bounded.

Nevertheless, perturbations are not always bounded and, in addition, they can be of stochastic nature, as modeling errors or measurement noises. Such signals acting on a system lead to the appearance of so-called chance constraints in the optimization problem solved by the model predictive controller for single (Cannon et al., 2012, Farina et al., 2016) or multi-agent systems (Dai et al., 2015, 2018). Then, this thesis also investigates chance-constrained MPC techniques for the deployment of MAS subject to *unbounded stochastic perturbations*.

Finally, based on the collision avoidance requirement, a question arises: what happens if one or several agents have to leave or join the area in which the deployment occurs? MPC algorithms can be used in this case, where the deploying multi-agent system needs to be reconfigured to deal with the modification in the number of agents. Indeed, it is possible to construct regions for the safe evolution of the agents during the reconfiguration phase using set-theoretic tools (Blanchini and Miani, 2015), translated into constraints to obtain a safe transient objective. For each agent, the elements needed for its safety require only the knowledge of the other agents' position, allowing for a decentralized algorithm to drive the MAS. Such problems often arise in fault-tolerant formation control and are relevant for various multi-vehicle system applications.

1.2 Contributions of the thesis

The present thesis proposes several flavors of decentralized MPC algorithms for the deployment of a MAS over a convex bounded two dimensional area. These algorithms are based on a Voronoi tessellation of the area to cover as well as on the Chebyshev center of the Voronoi cells as an objective point for the agents to be deployed. Then, the following paragraphs detail the contribution of this thesis for each of the proposed MPC techniques.

1.2.1 Deployment in the nominal case

Nguyen (2016) introduces a decentralized model predictive controller for the deployment of a nominal multi-agent system (i.e. no uncertainties, no fault) in a convex bounded area, where each agent is driven towards the Chebyshev center of its associated Voronoi cell. For its part, Hatleskog (2018) gives a proof of convergence of the Voronoi and Chebyshev-based deployment algorithm for the case of single integrator dynamics when the agents are driven with an unconstrained full state-feedback controller.

The present thesis proposes a Voronoi-based centralized MPC for the deployment of a MAS in a convex bounded area, each agent being driven towards the Chebyshev center of its Voronoi cell. After formulating the decentralized version of this deployment algorithm, it provides a proof of feasibility of the MPC optimization problem when the agents obey single integrator dynamics. This feasibility proof leads to a discussion on the convergence of the decentralized deployment algorithm. To show its efficiency, the proposed decentralized model predictive controller is used as the position controller for quadrotor unmanned aerial vehicles, modeled as nonlinear systems. Directions to prove the convergence of the deployment algorithm for a fleet of quadrotor UAVs are then discussed.

These contributions have led to the publication of the papers Chevet et al. (2018) and Chevet et al. (2020b).

1.2.2 Deployment under bounded deterministic perturbations

Papatheodorou et al. (2016) and Papatheodorou et al. (2017) study the deployment of a MAS over an area when the agents are subject to bounded position uncertainty. These works are based on the definition of an ellipsoidal-based so-called guaranteed Voronoi tessellation taking into account the fact that the generators of the cells are not points anymore but discs.

The present thesis introduces a new box-based guaranteed Voronoi tessellation where the generators of the Voronoi tessellation are not points but boxes (i.e. rectangles in a two-dimensional space). Such a guaranteed Voronoi tessellation is used to cope with the bounded output perturbations acting on the system.

To deal with the case of bounded perturbations acting on a system, a tube-based MPC strategy (Mayne et al., 2005) can be used. The idea behind such a control algorithm is to decompose the control input applied to a system into two parts, one obtained via optimization on the nominal dynamics and the other one by multiplying the error between the real state and the nominal state of the system by a gain matrix. The state of the system is then guaranteed to belong to a robust positively invariant set for the error dynamics centered on the nominal state of the system. Alvarado (2007) proposes a linear matrix inequality constrained optimization problem to obtain the gain matrix used in classical state-feedback tube-based MPC (Mayne et al., 2005).

This thesis then proposes a decentralized output-feedback tube-based MPC strategy for the deployment of a multi-agent system subject to bounded deterministic input and output perturbations. A novel linear/bilinear matrix inequality constrained optimization-based procedure is designed to obtain the gain matrices of the state-feedback part of the tube-based controller and of the observer introduced in the output-feedback strategy. This procedure aims to minimize the size of the robust positively invariant sets to which the state belongs such that the optimization part of the tube-based controller has enough degrees of freedom to optimize the performance of the system. The decentralized output-feedback tube-based MPC is firstly applied for the deployment of a MAS composed of agents obeying single integrator dynamics, and secondly for the position control of a fleet of quadrotor UAVs.

These contributions have led to the publication of the paper Chevet et al. (2019).

1.2.3 Deployment under unbounded stochastic perturbations

When a system is subject to unbounded stochastic perturbations, *probabilistic constraints*, also called *chance constraints*, appear on the optimization problem solved by the model predictive controller. Gavilan et al. (2012) propose a state-feedback chance-constrained model predictive controller for a single system where the chance constraints have to be satisfied with a probability equal to 1. Based on the stochastic properties of the perturbation, Gavilan et al. (2012) introduces a method to relax probabilistic constraints into algebraic constraints.

The present thesis introduces a new decentralized output-feedback chance-constrained MPC algorithm for the deployment of a MAS when the agents are subject to *unbounded stochastic perturbations*. However, the stochastic perturbation signal, which is unknown over the prediction horizon of the MPC, appears in the constraints of the MPC optimization problem. These constraints have to be satisfied with a probability equal to 1. A relaxation procedure of the probabilistic constraints into algebraic constraints in order to solve the MPC optimization problem is proposed along with a proof of feasibility. This relaxation uses the stochastic properties of the perturbations to find a bound allowing the constraints to be satisfied for almost all perturbations acting on the system. The decentralized output-feedback chance-constrained MPC is applied to the deployment of two MAS composed, respectively, of agents obeying single integrator dynamics and of agents obeying nonlinear quadrotor UAV dynamics.

These contributions have led to the publication of the paper Chevet et al. (2020a).

1.2.4 Reconfiguration in the case of a time-varying multi-vehicle system

Another contribution of the present thesis deals with the reconfiguration of a deploying multi-agent system when agents join or leave the area in which the deployment is occurring. This problem is related to fault-tolerant formation control since the main reasons leading an agent to leave the deployment area are related to faulty situations.

In the case of incoming agents, the proposed technique is a natural extension of the nominal deployment strategy. Indeed, the deployment algorithm being decentralized, it is easily scalable and allows intrinsically to consider additional agents.

To deal with the case of outgoing agents, the present work proposes two original strategies based on a transient objective, different from the Chebyshev center of the Voronoi cells, for the agents remaining in the deployment area allowing them to avoid the trajectory of the outgoing agents. For the first strategy, the transient objective of an agent is defined as the weighted barycenter of the agent's neighbors. For the second strategy, the transient objective of an agent is obtained by solving an optimization problem constraining the objective to belong to a region of the deployment area in which it is safe for this agent to evolve in order to avoid collisions with the outgoing agents. The second strategy has the advantage of allowing a reconfiguration of the MAS when several agents leave the deployment area simultaneously, increasing the fault-related capabilities.

For the case of agents joining the deployment area, simulation results are presented for a MAS with agents obeying single integrator dynamics. For the case of outgoing agents, the two reconfiguration strategies are compared on a MAS composed of agents obeying single integrator dynamics when one agent leaves the deployment area. Finally, simulation results are presented for the second reconfiguration strategy when two agents leave the deployment area with two MAS, the first one composed of agents obeying single integrator dynamics and the second composed of agents obeying nonlinear UAV dynamics.

These contributions have led to the publication of the papers Chevet et al. (2018) and Chevet et al. (2020b).

1.2.5 Publications

The work done during the preparation of this thesis has led to the submission and publication of several conference and journal papers.

Peer-reviewed journal paper

- **T. Chevet**, C. Vlad, C. Stoica Maniu, and Y.M. Zhang. Decentralized MPC for UAVs formation deployment and reconfiguration with multiple outgoing agents. *Journal of Intelligent & Robotic Systems*, 97(1):155-170, 2020.

Peer-reviewed conference papers

- **T. Chevet**, C. Stoica Maniu, C. Vlad, Y.M. Zhang, and E.F. Camacho. Chance-constrained MPC for Voronoi-based multi-agent system deployment. In 21st IFAC World Congress. Berlin, Germany, July 12–17, 2020.

- **T. Chevet**, C. Stoica Maniu, C. Vlad, and Y.M. Zhang. Guaranteed Voronoi-based deployment for multi-agent systems under uncertain measurements. In *18th European Control Conference*, pages 4016–4021. Naples, Italy, June 25–28, 2019.

- **T. Chevet**, C. Stoica Maniu, C. Vlad, and Y.M. Zhang. Voronoi-based UAVs formation deployment and reconfiguration using MPC techniques. In *International Conference on Unmanned Aircraft Systems*, pages 9–14. Dallas, TX, United States, June 12–15, 2018.

Other publications

Handouts

- **T. Chevet**, M.A. Lefebvre, V. Letort-Le Chevalier, D. Madhavan Brochier, C. Maniu, G. Sandou, and C. Vlad. Model Representations and Analysis. *CentraleSupélec Handouts*. 210 pages. 2020.

- **T. Chevet**, M.A. Lefebvre, V. Letort-Le Chevalier, C. Maniu, G. Sandou, and C. Vlad. Modélisation. Représentations et analyse des modèles. *CentraleSupélec Handouts* (in French). 210 pages. 2020.

Peer-reviewed journal paper

- D. Merhy, C. Stoica Maniu, T. Alamo, E.F. Camacho, S. Ben Chabane, **T. Chevet**, M. Makarov, and I. Hinostroza. Guaranteed set-membership estimation of an octorotor's position for radar applications. *International Journal of Control*. In press.

Peer-reviewed conference papers

- C. Stoica Maniu, C. Vlad, **T. Chevet**, S. Bertrand, A. Venturino, G. Rousseau, and S. Olaru. Control systems engineering made easy: Motivating students through experimentation on UAVs. In 21st IFAC World Congress, Demonstrator session. Berlin, Germany, July 12–17, 2020.

- D. Merhy, C. Stoica Maniu, T. Alamo, E.F. Camacho, **T. Chevet**, M. Makarov, and I. Hinostroza. Zonotopic set-membership state estimation applied to an octorotor model. In *12th Summer Workshop on Interval Methods*. Palaiseau, France, July 23–26, 2019.

- C. Stoica Maniu, C. Vlad, **T. Chevet**, G. Rousseau, S. Bertrand, and S. Olaru. Modernizing teaching through experimentation on UAVs formations. *12th IFAC Symposium on Advances in Control Education*, Invited Demonstration Session. Philadelphia, PA, United States, July 7–9, 2019. In *IFAC-PapersOnLine*, 52(9):144–146, 2019.

Oral presentations

- **T. Chevet**, C. Stoica Maniu, C. Vlad, Y.M. Zhang, and E.F. Camacho. Voronoi-based deployment of multi-vehicle systems. Seminar at *Departamento de Ingeniería de Sistemas y Automática*, University of Seville, Spain, December 17, 2019.

- **T. Chevet**, M. Makarov, C. Stoica Maniu, I. Hinostroza, and P. Tarascon. State estimation of an octorotor with unknown inputs. Application to radar imaging. Seminar at *Networked Autonomous Vehicles Lab*, Concordia University, Montréal, Canada, October 31, 2017.

1.3 Thesis outline

The remainder of the present thesis is organized as follows.

Chapter 2: Mathematical tools and set-theoretic elements This chapter provides the different mathematical results necessary for the development of the decentralized model predictive control algorithms presented in this thesis. It starts by recalling some properties on matrices and linear/bilinear matrix inequalities. It then introduces necessary set-theoretic elements before presenting the notion of set invariance in control theory. The general state-space representation of a MAS is described along with the necessary assumptions for the development of the MPC algorithms. Then, the construction of several types of Voronoi tessellation is detailed. The construction of the classical Voronoi tessellation is given before introducing two contributions of this thesis, i.e. the *box-based guaranteed Voronoi tessellation* and the so-called *pseudo-Voronoi tessellation*, along with a formal definition of a Chebyshev configuration. Finally, Chapter 2 presents some classical results on continuous univariate and multivariate random variables.

Chapter 3: Decentralized control for the deployment of a multi-vehicle system This chapter starts by giving a centralized and a decentralized formulation of the Voronoi-based deployment algorithm. It then gives a proof of feasibility of the MPC optimization problem when the agents obey single integrator dynamics and discusses the convergence of the deployment algorithm. After, simulation results are presented for the deployment of a MAS composed of agents obeying single integrator dynamics. The nonlinear dynamics of quadrotor unmanned aerial vehicles are introduced, as well as a cascaded control structure to allow the use of a decentralized

output-feedback MPC strategy for position control. Simulation results are exposed for the deployment of a MAS composed of agents obeying nonlinear quadrotor UAV dynamics. Chapter 3 ends on a discussion of possible directions for the convergence proof of the deployment algorithm when agents obey double integrator dynamics or more complex dynamics.

Chapter 4: Deployment of a multi-vehicle system subject to perturbations This chapter is twofold in the sense that it presents two robust decentralized MPC algorithms for the deployment of a MAS: the first one for agents subject to bounded deterministic perturbations, and the second one for agents subject to unbounded stochastic perturbations. In the first part, a decentralized output-feedback tube-based model predictive controller is introduced along with two linear/bilinear matrix inequality constrained optimization problems to obtain the observer and state-feedback gain matrices necessary for the tube-based strategy. In the second part, a decentralized output-feedback chance-constrained model predictive controller is exposed along with a method allowing probabilistic constraints relaxation into algebraic constraints. In both cases, simulation results are presented for the Voronoi-based deployment of a MAS firstly composed of agents obeying single integrator dynamics and secondly composed of agents obeying nonlinear quadrotor UAV dynamics.

Chapter 5: Extension to the deployment of a time-varying multi-vehicle system This chapter aims to present reconfiguration algorithms for a MAS in the case of agents joining or leaving the MAS during the deployment. It starts by presenting the decentralized MPC strategy for the incoming agents. Then, a first reconfiguration strategy to deal with the case of one agent leaving the deployment area is introduced. This reconfiguration strategy is based on a transient objective computed as an agent's neighbors' barycenter. Simulation results are presented in both situations for a MAS composed of agents obeying single integrator dynamics. Then, Chapter 5 introduces a new decentralized reconfiguration strategy to drive the remaining agents safely away from the outgoing agents when several agents leave the MAS simultaneously. This second strategy is based on a new transient objective obtained by solving an optimization problem constraining it to belong to a region in which it is safe for the remaining agent to evolve. This new algorithm is then compared in simulation with the first reconfiguration strategy. Finally, simulation results are presented for the deployment and reconfiguration of two MAS when two agents leave the deployment area using the improved reconfiguration algorithm. The first MAS is composed of agents obeying single integrator dynamics and the second one of agents obeying nonlinear quadrotor UAV dynamics.

Chapter 6: Concluding remarks and future work This chapter completes the present thesis by means of concluding remarks and gathers several open directions of this work.

Chapter 2

Mathematical tools and set-theoretic elements

Table of Contents

2.1 Definitions and useful properties of matrices

As stated by Blanchini and Miani (2015) in the preface of their book, sets will naturally appear with three aspects of control theory: constraints, uncertainties and design specifications. In this context, this chapter focuses on the main set properties necessary for the comprehension of the results elaborated in the present thesis. Before introducing elements about sets, it is necessary to present some useful definitions about matrices.

Definition 2.1: Positive definite matrix

A symmetric matrix $\boldsymbol{P} = \boldsymbol{P}^\top \in \mathbb{R}^{n\times n}$ is called *positive definite* (respectively *positive semidefinite*), denoted by $\boldsymbol{P} \succ 0$ (respectively by $\boldsymbol{P} \succeq 0$), if $\mathbf{x}^\top \boldsymbol{P}\mathbf{x} > 0$ (respectively $\mathbf{x}^\top \boldsymbol{P}\mathbf{x} \geq 0$) for all $\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}_{n\times 1}\}$.

Definition 2.2: Weighted quadratic norm

Let $\boldsymbol{P} \in \mathbb{R}^{n\times n}$ be a positive semidefinite matrix and $\mathbf{x} \in \mathbb{R}^n$ be a vector. The quadratic norm of $\mathbf{x}$ weighted by $\boldsymbol{P}$ is the quantity $\|\mathbf{x}\|_{\boldsymbol{P}} = \sqrt{\mathbf{x}^\top \boldsymbol{P}\mathbf{x}}$.

The two following properties of special matrices related to positive definiteness are important for the results presented in this thesis.

Property 2.1: Exponential of a matrix (Hall, 2015)

Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ be a matrix. Then, the exponential of matrix $\boldsymbol{A}$:

$$\exp(\boldsymbol{A}) = \sum_{i=0}^{+\infty} \frac{\boldsymbol{A}^i}{i!} \tag{2.1}$$

is positive definite, denoted by $\exp(\boldsymbol{A}) \succ 0$.

Property 2.2: Product of matrices (Horn and Johnson, 2013)

Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ be a positive semidefinite matrix and $\boldsymbol{B} \in \mathbb{R}^{n\times m}$ be a matrix with $m \leq n$. Then, the product $\boldsymbol{B}^\top \boldsymbol{A}\boldsymbol{B}$ is positive semidefinite. It is positive definite if and only if $\boldsymbol{A}$ is positive definite and $\boldsymbol{B}$ has rank m.

The concept of positive definite matrices is also used to introduce Linear Matrix Inequalities in control theory.

Definition 2.3: Linear Matrix Inequality (Boyd et al., 1994)

A Linear Matrix Inequality (LMI) is an expression of the following form:

$$\boldsymbol{F}(\mathbf{x}) = \boldsymbol{F}_0 + \sum_{i=1}^{n} x_i \boldsymbol{F}_i \succ 0 \tag{2.2}$$

where $\mathbf{x} = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}^\top \in \mathbb{R}^n$ is the vector of decision variables and the matrices $\boldsymbol{F}_i \in \mathbb{R}^{m\times m}$, with $i \in \overline{0, n}$, are symmetric.

Remark 2.1: Non strict LMI
Definition 2.3 can be extended to the case where the inequality is not strict. In this case, $\boldsymbol{F}(\mathbf{x})$ is considered to be positive semidefinite, denoted by $\boldsymbol{F}(\mathbf{x}) \succeq 0$. ◊

Remark 2.2: Affine matrix
A matrix $\boldsymbol{F}(\mathbf{x}) \in \mathbb{R}^{m\times m}$ is called affine in $\mathbf{x} \in \mathbb{R}^n$ if it can be written as:

$$\boldsymbol{F}(\mathbf{x}) = \boldsymbol{F}_0 + \sum_{i=1}^{n} x_i \boldsymbol{F}_i. \tag{2.3}$$

If $\boldsymbol{F}_0 = \mathbf{0}_m$, the the matrix $\boldsymbol{F}(\mathbf{x})$ is linear with respect to $\mathbf{x}$. ◊

LMI problems often arise in several domains of control engineering, such as robust control (Scherer and Weiland, 2015) or state-estimation (Le et al., 2013) for example. These problems often consist in minimizing a linear objective $\mathbf{c}^\top \mathbf{x}$, with $\mathbf{c} \in \mathbb{R}^n$

under strict or nonstrict LMI constraints such as:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} \quad & \\ & \boldsymbol{F}(\mathbf{x}) \succ 0. \end{aligned} \tag{2.4}$$

These problems can be solved numerically with appropriate solvers. For example, the Robust Control Toolbox of Matlab® provides a solver to solve the problems of the form (2.4) called `mincx`. Other openly distributed toolboxes for Matlab® such as CVX (Grant and Boyd, 2014, 2008) or YALMIP (Löfberg, 2004, 2020) allow to solve problems of the form (2.2) but also more general problems where the LMIs are nonstrict.

Property 2.3: Schur complement (Boyd et al., 1994)

Let $\boldsymbol{Q}(\mathbf{x}) \in \mathbb{R}^{n \times n}$ and $\boldsymbol{R}(\mathbf{x}) \in \mathbb{R}^{m \times m}$ be symmetric matrices affine in $\mathbf{x}$ and $\boldsymbol{S}(\mathbf{x}) \in \mathbb{R}^{n \times m}$ a matrix affine in $\mathbf{x}$. Then the LMI:

$$\begin{bmatrix} \boldsymbol{Q}(\mathbf{x}) & \boldsymbol{S}(\mathbf{x}) \\ \boldsymbol{S}^\top(\mathbf{x}) & \boldsymbol{R}(\mathbf{x}) \end{bmatrix} \succ 0 \tag{2.5}$$

is equivalent to:

$$\begin{cases} \boldsymbol{Q}(\mathbf{x}) \succ 0 \\ \boldsymbol{R}(\mathbf{x}) - \boldsymbol{S}^\top(\mathbf{x}) \boldsymbol{Q}^{-1}(\mathbf{x}) \boldsymbol{S}(\mathbf{x}) \succ 0 \end{cases} \tag{2.6}$$

or:

$$\begin{cases} \boldsymbol{R}(\mathbf{x}) \succ 0 \\ \boldsymbol{Q}(\mathbf{x}) - \boldsymbol{S}(\mathbf{x}) \boldsymbol{R}^{-1}(\mathbf{x}) \boldsymbol{S}^\top(\mathbf{x}) \succ 0. \end{cases} \tag{2.7}$$

The concept of LMI can be extended to the more general case of Bilinear Matrix Inequalities.

Definition 2.4: Bilinear Matrix Inequality

A Bilinear Matrix Inequality (BMI) is an inequality of the form:

$$\boldsymbol{F}(\mathbf{x}) = \boldsymbol{F}_0 + \sum_{i=1}^{n} x_i \boldsymbol{F}_i + \sum_{i=1}^{n} \sum_{j=1}^{n} x_i x_j \boldsymbol{F}_{ij} \succ 0 \tag{2.8}$$

where $\mathbf{x} = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}^\top \in \mathbb{R}^n$ is the vector of decision variables and $\boldsymbol{F}_0, \boldsymbol{F}_i, \boldsymbol{F}_{ij} \in \mathbb{R}^{m \times m}$, with $i \in \overline{1, n}$ and $j \in \overline{1, n}$, are symmetric matrices.

BMI problems can sometimes be reduced to LMI problems using for instance Property 2.3. When they cannot, specific optimization solvers exist such as PENBMI (Henrion et al., 2005) or its open source counterpart PENLAB (Fiala et al., 2013) that can solve optimization problems under BMI constraints. Such BMI constraints often appear in control with one specific procedure presented hereafter.

Theorem 2.1: S-procedure (Boyd et al., 1994)

Let $F_i \in \mathbb{R}$, with $i \in \overline{0, m}$ be quadratic functions of a variable $\mathbf{x} \in \mathbb{R}^n$:

$$F_i(\mathbf{x}) = \mathbf{x}^\top \boldsymbol{Q}_i \mathbf{x} + 2\mathbf{f}_i^\top \mathbf{x} + r_i \tag{2.9}$$

where $\boldsymbol{Q}_i \in \mathbb{R}^{n \times n}$ is a symmetric matrix, $\mathbf{f}_i \in \mathbb{R}^n$ and $r_i \in \mathbb{R}$, with $i \in \overline{0, m}$. If $\exists \tau_i \geq 0$, with $i \in \overline{1, m}$, such that:

$$F_0(\mathbf{x}) - \sum_{i=1}^{m} \tau_i F_i(\mathbf{x}) \geq 0 \tag{2.10}$$

then $F_0(\mathbf{x}) \geq 0$ for all $\mathbf{x}$ such that $F_i(\mathbf{x}) \geq 0$ for all $i \in \overline{1, m}$.

The link between the S-procedure defined in Theorem 2.1 and LMIs and BMIs is not immediate from what is presented in Theorem 2.1. Using (2.9), the left hand side of inequality (2.10) can be rewritten as:

$$\begin{aligned} F_0(\mathbf{x}) - \sum_{i=1}^{m} \tau_i F_i(\mathbf{x}) &= \mathbf{x}^\top \left(\boldsymbol{Q}_0 - \sum_{i=1}^{m} \tau_i \boldsymbol{Q}_i \right) \mathbf{x} + 2 \left(\mathbf{f}_0 - \sum_{i=1}^{m} \tau_i \mathbf{f}_i \right)^\top \mathbf{x} \\ &\qquad + \left(r_0 - \sum_{i=1}^{m} \tau_i r_i \right) \\ &= \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^\top \begin{bmatrix} \boldsymbol{Q}_0 - \sum_{i=1}^{m} \tau_i \boldsymbol{Q}_i & \mathbf{f}_0 - \sum_{i=1}^{m} \tau_i \mathbf{f}_i \\ \mathbf{f}_0^\top - \sum_{i=1}^{m} \tau_i \mathbf{f}_i^\top & r_0 - \sum_{i=1}^{m} \tau_i r_i \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}. \end{aligned}$$

Then, with the notation:

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{Q}_0 - \sum_{i=1}^{m} \tau_i \boldsymbol{Q}_i & \mathbf{f}_0 - \sum_{i=1}^{m} \tau_i \mathbf{f}_i \\ \mathbf{f}_0^\top - \sum_{i=1}^{m} \tau_i \mathbf{f}_i^\top & r_0 - \sum_{i=1}^{m} \tau_i r_i \end{bmatrix},$$

inequality (2.10) is equivalent to the nonstrict matrix inequality:

$$\boldsymbol{F} \succeq 0. \tag{2.11}$$

If the matrices $\boldsymbol{Q}_i$, the vectors $\mathbf{f}_i$ and the scalars r_i, with $i \in \overline{0, m}$, are *known*, then (2.11) is a LMI where the vector of decision variables is $\boldsymbol{\tau} = \begin{bmatrix} \tau_1 & \cdots & \tau_m \end{bmatrix}^\top$. However, if the terms $\boldsymbol{Q}_i$, $\mathbf{f}_i$ and r_i are also *decision variables*, then the expression (2.11) is a BMI. The S-procedure is used in Chapter 4 to design tube-based model predictive control policies.

2.2 Set-theoretic elements for control

Given their importance in control theory, this part of the present chapter focuses on definitions relative to sets and operations on sets. One of the most common class

of sets used in control is the class of convex sets (Boyd and Vandenberghe, 2009) and all the results presented in this thesis are based on such convex sets, hence the necessity of the following definition.

Definition 2.5: Convex set

A set $\mathcal{S} \in \mathbb{R}^n$ is called convex if for all $\mathbf{x}_1, \ldots, \mathbf{x}_k \in \mathcal{S}$, with $k \geq 2$:

$$\sum_{i=1}^{k} \alpha_i \mathbf{x}_i \in \mathcal{S}, \text{ with } \alpha_1, \ldots, \alpha_k \in \mathbb{R} \text{ such that } \sum_{i=1}^{k} \alpha_i = 1.$$

Another general definition which is of use in Sections 2.2.2 and 2.2.3 is that of the convex hull of a finite set of points.

Definition 2.6: Convex hull

The convex hull of a finite set of points $\mathcal{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$, with $\mathbf{v}_1, \ldots, \mathbf{v}_k \in \mathbb{R}^n$ and $k \geq 2$, is the set:

$$\text{conv}(\mathcal{V}) = \left\{ \mathbf{x} \in \mathbb{R}^n \;\middle|\; \mathbf{x} = \sum_{i=1}^{k} \lambda_i \mathbf{v}_i, \lambda_i \in \mathbb{R}_+, \sum_{i=1}^{k} \lambda_i = 1 \right\}$$

The remainder of this section is organized as follows: two types of sets, namely ellipsoidal and polyhedral sets are defined in Sections 2.2.1 and 2.2.2 respectively. Section 2.2.3 presents some classical operations on sets and Section 2.2.4 introduces the concept of set invariance.

2.2.1 Ellipsoidal sets

Ellipsoids are one of the most widespread family of convex sets in control theory (Kurzhanski and Vályi, 1997, Blanchini and Miani, 2015). They are used in numerous areas such as identification (Polyak et al., 2004), state estimation (Schweppe, 1968, Merhy, 2019) and, what is mainly relevant for this thesis, robust control (Boyd et al., 1994, Poznyak et al., 2014). Due to their relevance in such fields, some definitions are further detailed.

Definition 2.7: Ellipsoidal set

Given a symmetric positive definite matrix $\boldsymbol{P} = \boldsymbol{P}^\top \succ 0$ with $\boldsymbol{P} \in \mathbb{R}^{n \times n}$, a vector $\mathbf{c} \in \mathbb{R}^n$ and a real scalar $\rho > 0$, the ellipsoid $\mathscr{E}(\boldsymbol{P}, \mathbf{c}, \rho)$ is the set:

$$\mathscr{E}(\boldsymbol{P}, \mathbf{c}, \rho) = \left\{ \mathbf{x} \in \mathbb{R}^n \;\middle|\; (\mathbf{x} - \mathbf{c})^\top \boldsymbol{P} (\mathbf{x} - \mathbf{c}) \leq \rho \right\} \tag{2.12}$$

where $\boldsymbol{P}$ is called the *shape matrix*, $\mathbf{c}$ the *center* and ρ the so called *radius* of $\mathscr{E}(\boldsymbol{P}, \mathbf{c}, \rho)$.

Example 2.1: An ellipsoidal set in $\mathbb{R}^2$

Let $\boldsymbol{P} = \begin{bmatrix} 2 & -1 \\ -1 & 4 \end{bmatrix}$, $\mathbf{c} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$ *and* $\rho = 6$. *Then, Figure 2.1 shows, in red, the set*

$\mathscr{E}(\boldsymbol{P}, \mathbf{c}, 6)$. *Moreover, the border of* $\mathscr{E}(\boldsymbol{P}, \mathbf{c}, 6)$ *is the set:*

$$\partial\mathscr{E}(\boldsymbol{P}, \mathbf{c}, 6) = \left\{ \mathbf{x} \in \mathbb{R}^2 \;\middle|\; (\mathbf{x} - \mathbf{c})^\top \boldsymbol{P} (\mathbf{x} - \mathbf{c}) = 6 \right\}$$

which is an ellipse.

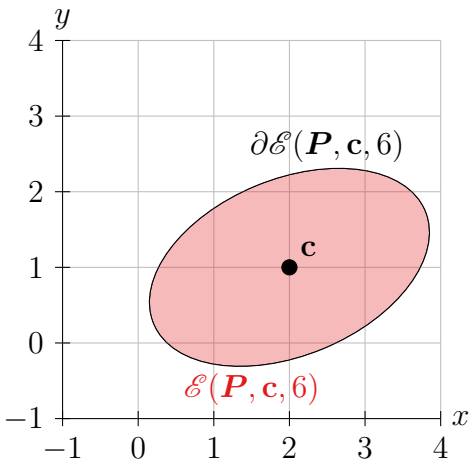


Figure 2.1: An example of ellipsoidal set in $\mathbb{R}^2$.

Definition 2.8: Normalized ellipsoidal set

A normalized ellipsoidal set is an ellipsoidal set with $\rho = 1$, leading to:

$$\mathscr{E}(\boldsymbol{Q}, \mathbf{c}, 1) = \mathscr{E}(\boldsymbol{Q}, \mathbf{c}) = \left\{ \mathbf{x} \in \mathbb{R}^n \;\middle|\; (\mathbf{x} - \mathbf{c})^\top \boldsymbol{Q} (\mathbf{x} - \mathbf{c}) \leq 1 \right\}.$$

Note that such a normalized set can be obtained from an ellipsoidal set by taking $\boldsymbol{Q} = \boldsymbol{P}/\rho$.

Example 2.2: A normalized ellipsoidal set in $\mathbb{R}^2$
By taking the same numerical values as in Example 2.1, if $\boldsymbol{Q} = \boldsymbol{P}/6$, *one has that* $\mathscr{E}(\boldsymbol{Q}, \mathbf{c}) = \mathscr{E}(\boldsymbol{P}, \mathbf{c}, 6)$. *This way, Figure 2.1 also represents the normalized ellipsoidal set* $\mathscr{E}(\boldsymbol{Q}, \mathbf{c})$ *in red and its border in black.*

The attentive reader will have noticed that, for a normalized ellipsoidal set $\mathscr{E}(\boldsymbol{Q}, \mathbf{c})$, since the radius ρ is unitary, it can be omitted from the notation. The same way, if $\mathbf{c} = \mathbf{0}_n$, the center is omitted from the notation, giving $\mathscr{E}(\boldsymbol{P}, \rho)$ for a *centered ellipsoidal set* or $\mathscr{E}(\boldsymbol{Q})$ for a *centered normalized ellipsoidal set.*

However, while simple to use and of reduced complexity, an ellipsoidal set can be more conservative than other sets such as polyhedral sets introduced in Section 2.2.2.

2.2.2 Polyhedral sets

The family of polyhedral sets is another widespread family of convex sets appearing in control theory (Blanchini and Miani, 2015). They appear in particular when linear constraints are applied to a system. Polyhedral sets have two types of representation either based on half spaces (Definition 2.9) or on vertices (Definition 2.11), making

them very flexible and allowing them to be a good approximation of any convex set (Bronstein, 2008). Another advantage of using polyhedral sets is that they are invariant by the operations presented in Section 2.2.3, i.e. the result of an operation between two polyhedral sets is a polyhedral set. Polyhedral sets are less conservative than ellipsoidal sets. However, their complexity do not depend on the dimension of the space but on the number of vertices, which can quickly increase.

Definition 2.9: Half-space or H-representation (Schrijver, 1998)

A polyhedral set or *polyhedron* $\mathcal{P} \subset \mathbb{R}^n$ is the combination of finitely many linear inequalities:

$$\mathcal{P} = \{\mathbf{x} \in \mathbb{R}^n \mid \boldsymbol{H}\mathbf{x} \leq \boldsymbol{\theta}\}, \tag{2.13}$$

with $\boldsymbol{H} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{\theta} \in \mathbb{R}^m$, where m is the number of inequalities defining $\mathcal{P}$.

From a geometrical point of view, each row of $\boldsymbol{H}$ and the associated element of $\boldsymbol{\theta}$ in (2.13) define a closed half-space $\mathcal{P}_i$, with $i \in \overline{1, m}$, such that:

$$\mathcal{P}_i = \{\mathbf{x} \in \mathbb{R}^n \mid \boldsymbol{h}_i\mathbf{x} \leq \theta_i\}$$

where $\boldsymbol{h}_i$ and θ_i are the i-th rows of $\boldsymbol{H}$ and $\boldsymbol{\theta}$, hence the name half-space representation or H-representation. The border $\partial\mathcal{P}_i$ of $\mathcal{P}_i$, is the hyperplane in $\mathbb{R}^n$:

$$\partial\mathcal{P}_i = \{\mathbf{x} \in \mathbb{R}^n \mid \boldsymbol{h}_i\mathbf{x} = \theta_i\}.$$

This way, the rows $\boldsymbol{h}_i$ of $\boldsymbol{H}$ can be interpreted as the transpose of the normal vectors to the hyperplanes $\partial\mathcal{P}_i$ and the elements θ_i of $\boldsymbol{\theta}$ as the algebraic distances of these hyperplanes to the origin $\mathbf{0}_{n\times 1}$, if $\left\|\boldsymbol{h}_i^\top\right\|_2 = 1$. Indeed, since the distance between $\partial\mathcal{P}_i$ and a vector $\mathbf{x} \in \mathbb{R}^n$ is:

$$d(\mathbf{x}, \partial\mathcal{P}_i) = \frac{|\boldsymbol{h}_i\mathbf{x} - \theta_i|}{\left\|\boldsymbol{h}_i^\top\right\|_2},$$

if $\left\|\boldsymbol{h}_i^\top\right\|_2 = 1$, $d(\mathbf{0}_{n\times 1}, \partial\mathcal{H}_i) = |\theta_i|$.

Example 2.3: A polyhedron in $\mathbb{R}^2$

Let $\boldsymbol{H} = \begin{bmatrix} 3 & 2 \\ 0 & 1 \end{bmatrix}$ *and* $\boldsymbol{\theta} = \begin{bmatrix} 8 \\ 2 \end{bmatrix}$. *Then,* $\boldsymbol{H}$ *and* $\boldsymbol{\theta}$ *induce a polyhedron* $\mathcal{P}$ *in* $\mathbb{R}^2$ *(as per Definition 2.9) represented in Figure 2.2. This figure also shows the hyperplanes* $\partial\mathcal{P}_1$ *and* $\partial\mathcal{P}_2$ *defined respectively by the first and second rows of* $\boldsymbol{H}$ *and* $\boldsymbol{\theta}$*, as well as the normal vectors* $\boldsymbol{h}_1^\top$ *and* $\boldsymbol{h}_2^\top$.

However, as it can be seen in Example 2.3, a polyhedral set can be unbounded. In control theory, a specific class of polyhedral sets is used, namely the class of bounded polyhedral sets.

Definition 2.10: Polytope

A bounded polyhedron is called a polytope.

Remark 2.3: H-polytope
A polytope defined by a H-representation is called a H-polytope. ◊

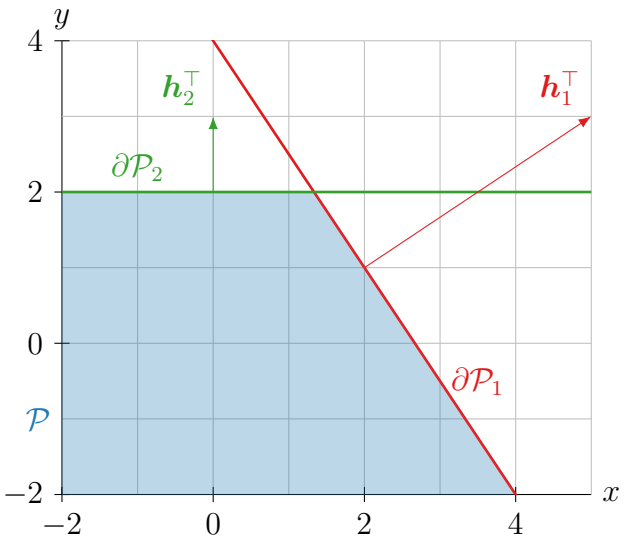


Figure 2.2: An example of polyhedral set in $\mathbb{R}^2$.

Example 2.4: A polytope in $\mathbb{R}^2$

Let $\boldsymbol{H} = \begin{bmatrix} 3 & 0 & -1 & -1 & 2 \\ 2 & 1 & 2 & -1 & -1 \end{bmatrix}^\top$ *and* $\boldsymbol{\theta} = \begin{bmatrix} 8 & 2 & 3 & 5 & 4 \end{bmatrix}^\top$. *Then* $\boldsymbol{H}$ *and* $\boldsymbol{\theta}$ *induce a polyhedron* $\mathcal{P}$ *in* $\mathbb{R}^2$ *(as per Definition 2.9) represented in Figure 2.3. It is obvious from the representation in Figure 2.3 that* $\mathcal{P}$ *is a polytope since it is bounded.*

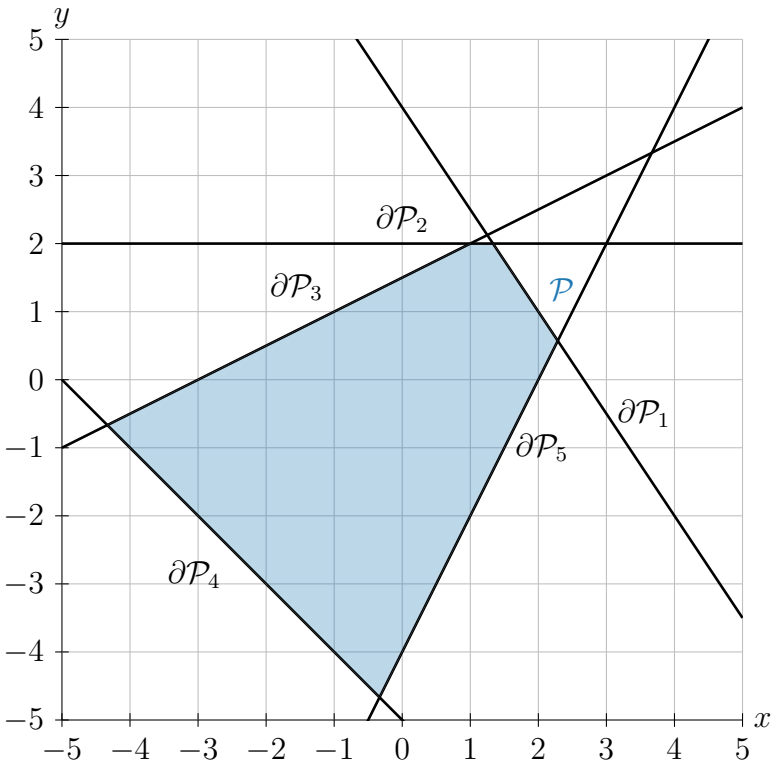


Figure 2.3: An example of polytopic set in $\mathbb{R}^2$.

Before going further in the definitions about polyhedrons and polytopes, a way to test if a polytope is empty or not is presented below.

Theorem 2.2: Farkas' lemma (Matoušek and Gärtner, 2007)

Let $\boldsymbol{H} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{\theta} \in \mathbb{R}^m$. Then exactly one of the two following statements is true:

1. There exists a vector $\mathbf{x} \in \mathbb{R}^n$ such that $\boldsymbol{H}\mathbf{x} \leq \boldsymbol{\theta}$;
2. There exists a vector $\mathbf{y} \in \mathbb{R}^m$ such that $\mathbf{y} \geq 0$, $\boldsymbol{H}^\top \mathbf{y} = \mathbf{0}_{n \times 1}$ and $\boldsymbol{\theta}^\top \mathbf{y} < 0$.

With Theorem 2.2, it is then possible to test if a polytope is empty or not given its H-representation.

Now that the half-space representation of a polyhedron has been introduced, it is time to introduce its dual representation, the vertex representation. However, this definition is limited to the V-representation of a polytope, the general definition for a polyhedron being out of the scope of this thesis.

Definition 2.11: Vertex or V-representation

A polytopic set or *polytope* $\mathcal{P} \subset \mathbb{R}^n$ is the linear combination of finitely many points called vertices $\mathbf{v}_1, \ldots, \mathbf{v}_k \in \mathbb{R}^n$, with $k \geq 2$:

$$\mathcal{P} = \left\{ \mathbf{x} \in \mathbb{R}^n \;\middle|\; \mathbf{x} = \sum_{i=1}^{k} \lambda_i \mathbf{v}_i, \lambda_i \in \mathbb{R}_+, \sum_{i=1}^{k} \lambda_i = 1 \right\}. \tag{2.14}$$

Remark 2.4: V-polytope
A polytope defined by a V-representation is called a V-polytope. ◊

Remark 2.5: Polytope and convex hull of vertices
Consider a set of points $\mathcal{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$, where $\mathbf{v}_i \in \mathbb{R}^n$, with $i \in \overline{1, k}$, $k \geq 2$. If $\mathcal{V}$ is the set of the vertices of a polytope $\mathcal{P} \subset \mathbb{R}^n$, then:

$$\mathcal{P} = \text{conv}(\mathcal{V}),$$

i.e. a polytope is the convex hull of its vertices. ◊

Two ways to represent polytopes have been introduced. However, these two representations are formally different, since one needs a set of hyperplanes and the other requires a set of vertices to induce a polytope. Intuitively, seeking all the intersection points of the hyperplanes, one can find the vertices or define hyperplanes from the vertices. While its proof is not trivial, the following theorem formalizes this idea.

Theorem 2.3: Main theorem for polytopes (Ziegler, 2007)

A subset $\mathcal{P} \in \mathbb{R}^n$ is the convex hull of a finite set of points if and only if it is a bounded intersection of half-spaces.

Corollary 2.4: Equivalence of H- and V-representation of polytopes

A polytope $\mathcal{P} \in \mathbb{R}^n$ admits a V-representation if and only if it admits a H-representation.

Example 2.5: Equivalence between H and V-representation of polytopes
Let $\mathcal{V}$ be a family of vectors of $\mathbb{R}^2$:

$$\mathcal{V} = \left\{ \mathbf{v}_1 = \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \mathbf{v}_2 = \begin{bmatrix} 1 \\ 3 \end{bmatrix}, \mathbf{v}_3 = \begin{bmatrix} -2 \\ 2 \end{bmatrix}, \mathbf{v}_4 = \begin{bmatrix} -2 \\ -2 \end{bmatrix}, \mathbf{v}_5 = \begin{bmatrix} 1 \\ -2 \end{bmatrix} \right\}.$$

The polytope $\mathcal{P} = \text{conv}(\mathcal{V})$ is presented in Figure 2.4 as well as the points $\mathbf{v}_i$, with $i \in \overline{1,5}$.

With this basic example in two dimensions, it is easy to extract the H-representation of $\mathcal{P}$ by finding the equations of the lines joining two consecutive vertices. This way:

$$\mathcal{P} = \left\{ \mathbf{x} \in \mathbb{R}^n \;\middle|\; \begin{bmatrix} 1 & 1 \\ -1 & 3 \\ -1 & 0 \\ 0 & -1 \\ 3 & -2 \end{bmatrix} \mathbf{x} \leq \begin{bmatrix} 4 \\ 8 \\ 2 \\ 2 \\ 7 \end{bmatrix} \right\}.$$

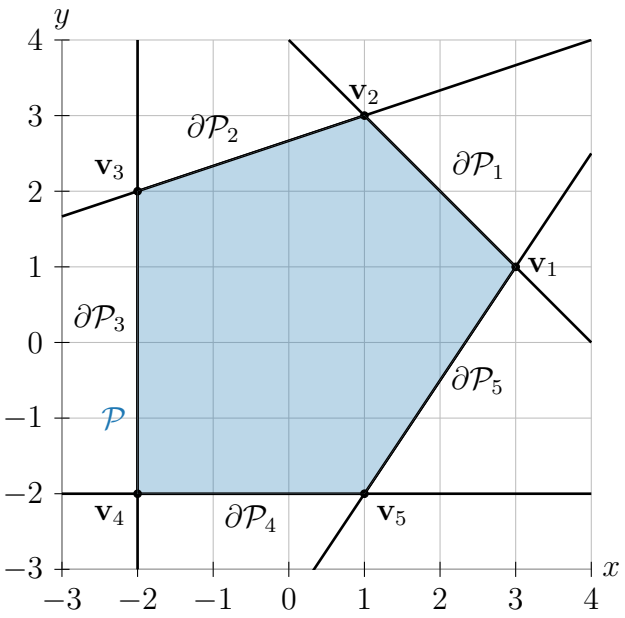


Figure 2.4: Equivalence between V-representation and H-representation for a polytope in $\mathbb{R}^2$.

Example 2.5 illustrates the equivalence between H-representation and V-representation of a polytope given in Corollary 2.4. The construction of the H-representation from a V-representation, presented in the case of a polytope in $\mathbb{R}^2$ in Example 2.5, is known as the *facet enumeration problem*. The construction of the V-representation from a H-representation is known as the *vertex enumeration problem*. Several algorithms exist to solve these problems (Fukuda and Prodon, 1995, Bremner et al., 1998) but their runtime increases greatly with the number of vertices.

Finally, a special kind of polytope that is used to define perturbation sets in Paragraph 2.4.2.1 or Section 4.1 or even to define constraint sets in the model predictive controllers of Chapters 3 to 5 is introduced.

Definition 2.12: Box

A box $\mathbb{B}^n(\boldsymbol{\alpha}) \subset \mathbb{R}^n$, with $\boldsymbol{\alpha} = \begin{bmatrix}\alpha_1 & \cdots & \alpha_n\end{bmatrix}^\top \in \mathbb{R}^n$, with $\alpha_i \geq 0$ for all $i \in \overline{1,n}$, is composed by n intervals:

$$\mathbb{B}^n(\boldsymbol{\alpha}) = \{\mathbf{x} \in \mathbb{R}^n \mid |\mathbf{x}| \leq \boldsymbol{\alpha}\}. \tag{2.15}$$

When $\boldsymbol{\alpha} \in \mathbb{R}^n$ in Definition 2.12 is reduced to $\mathbf{1}_{n\times 1}$, the box $\mathbb{B}^n(\boldsymbol{\alpha})$ is called a *unitary box*.

Definition 2.13: Unitary box

A unitary box $\mathbb{B}^n \subset \mathbb{R}^n$ is composed by n unitary intervals:

$$\mathbb{B}^n = \{\mathbf{x} \in \mathbb{R}^n \mid |\mathbf{x}| \leq \mathbf{1}_{n\times 1}\}. \tag{2.16}$$

2.2.3 Set operations

Two types of convex sets, ellipsoidal and polytopic sets, have been introduced in the previous paragraphs. This paragraph then introduces some useful operations on sets, mainly on polytopic sets since, as mentioned in Section 2.2.2, the set of polytopes is invariant by the operations that are presented in the following. Thus, the following definitions are given for general sets $\mathcal{X}, \mathcal{Y} \subset \mathbb{R}^n$, but some useful properties on polytopes coming from these definitions are introduced.

Definition 2.14: Cartesian product of two sets

The Cartesian product of two sets $\mathcal{X} \subset \mathbb{R}^n$ and $\mathcal{Y} \subset \mathbb{R}^m$ is the set:

$$\mathcal{X} \times \mathcal{Y} = \{(\mathbf{x}, \mathbf{y}) \mid \mathbf{x} \in \mathcal{X} \text{ and } \mathbf{y} \in \mathcal{Y}\}. \tag{2.17}$$

Property 2.4: Cartesian product of two polytopes

Let $\mathcal{P} \subset \mathbb{R}^n$, $\mathcal{Q} \subset \mathbb{R}^m$ be two polytopes for which a H-representation is known:

$$\mathcal{P} = \{\mathbf{x} \in \mathbb{R}^n \mid \boldsymbol{H}_\mathcal{P}\mathbf{x} \leq \boldsymbol{\theta}_\mathcal{P}\} \qquad \text{and} \qquad \mathcal{Q} = \{\mathbf{x} \in \mathbb{R}^m \mid \boldsymbol{H}_\mathcal{Q}\mathbf{x} \leq \boldsymbol{\theta}_\mathcal{Q}\}.$$

The Cartesian product of $\mathcal{P}$ and $\mathcal{Q}$ is the set:

$$\mathcal{P} \times \mathcal{Q} = \left\{ \mathbf{x} \in \mathbb{R}^{n+m} \;\middle|\; \operatorname{diag}(\boldsymbol{H}_\mathcal{P}, \boldsymbol{H}_\mathcal{Q})\mathbf{x} \leq \begin{bmatrix}\boldsymbol{\theta}_\mathcal{P} \\ \boldsymbol{\theta}_\mathcal{Q}\end{bmatrix} \right\}. \tag{2.18}$$

Definition 2.15: Intersection of two sets

The intersection of two sets $\mathcal{X}, \mathcal{Y} \subset \mathbb{R}^n$ is the set:

$$\mathcal{X} \cap \mathcal{Y} = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{x} \in \mathcal{X} \text{ and } \mathbf{x} \in \mathcal{Y}\}. \tag{2.19}$$

Property 2.5: Intersection of two polytopes

Let $\mathcal{P}, \mathcal{Q} \subset \mathbb{R}^n$ be two polytopes for which a H-representation is known:

$$\mathcal{P} = \{\mathbf{x} \in \mathbb{R}^n \mid \boldsymbol{H}_\mathcal{P}\mathbf{x} \leq \boldsymbol{\theta}_\mathcal{P}\} \qquad \text{and} \qquad \mathcal{Q} = \{\mathbf{x} \in \mathbb{R}^n \mid \boldsymbol{H}_\mathcal{Q}\mathbf{x} \leq \boldsymbol{\theta}_\mathcal{Q}\}.$$

The intersection of $\mathcal{P}$ and $\mathcal{Q}$ is the set:

$$\mathcal{P} \cap \mathcal{Q} = \left\{ \mathbf{x} \in \mathbb{R}^n \;\middle|\; \begin{bmatrix} \boldsymbol{H}_\mathcal{P} \\ \boldsymbol{H}_\mathcal{Q} \end{bmatrix} \mathbf{x} \leq \begin{bmatrix} \boldsymbol{\theta}_\mathcal{P} \\ \boldsymbol{\theta}_\mathcal{Q} \end{bmatrix} \right\}. \tag{2.20}$$

Remark 2.6: Redundant inequalities
In the H-representation (2.20) of the intersection of two polytopes, some inequalities are redundant as shown in Example 2.6. However, algorithms exists, such as the Fourier-Motzkin elimination method (Dantzig, 1972), to remove them. ◊

Example 2.6: Intersection of two polytopes
Let $\mathcal{P}$ be the polytope introduced in Example 2.5 and $\mathcal{Q}$ a polytope induced by $\boldsymbol{H}_\mathcal{Q} = \begin{bmatrix} -11 & -1 & 7 \\ 7 & -3 & 1 \end{bmatrix}^\top$ and $\boldsymbol{\theta}_\mathcal{Q} = \begin{bmatrix} 6 & 6 & 18 \end{bmatrix}^\top$. Figure 2.5 presents the two polytopes $\mathcal{P}$ and $\mathcal{Q}$ as well as their intersection $\mathcal{P} \cap \mathcal{Q}$.

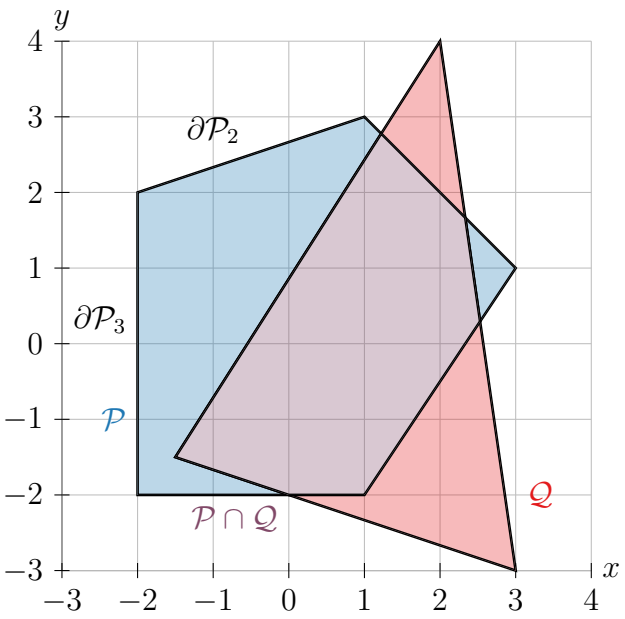


Figure 2.5: Intersection of two polytopes in $\mathbb{R}^2$.

As per Remark 2.6, it is obvious from Figure 2.5 that two inequalities are redundant. These inequalities are the ones associated with the hyperplanes $\partial\mathcal{P}_2$ and $\partial\mathcal{P}_3$. Thus, the second and third rows of $\boldsymbol{H}_\mathcal{P}$ and $\boldsymbol{\theta}_\mathcal{P}$ can be omitted from the

H-representation of $\mathcal{P} \cap \mathcal{Q}$ such that:

$$\mathcal{P} \cap \mathcal{Q} = \left\{ \mathbf{x} \in \mathbb{R}^2 \;\middle|\; \begin{bmatrix} 1 & 1 \\ 0 & -1 \\ 3 & -2 \\ -11 & 7 \\ -1 & -3 \\ 7 & 1 \end{bmatrix} \mathbf{x} \leq \begin{bmatrix} 4 \\ 2 \\ 7 \\ 6 \\ 6 \\ 18 \end{bmatrix} \right\}.$$

In general, the elimination of redundant constraints is not as simple as in Example 2.6 (a plot was still needed to see which constraints were redundant) and the complexity of the problem increases greatly with the number of inequalities and the dimension of the vector space $\mathbb{R}^n$.

Definition 2.16: Translation of a set

The translation of a set $\mathcal{X} \subset \mathbb{R}^n$ by a vector $\mathbf{t} \in \mathbb{R}^n$ is the set:

$$\mathrm{T}(\mathcal{X}, \mathbf{t}) = \{\mathbf{x} + \mathbf{t} \mid \mathbf{x} \in \mathcal{X}\}. \tag{2.21}$$

Property 2.6: Translation of a V-polytope

Let $\mathcal{P} \subset \mathbb{R}^n$ be a polytope for which a V-representation is known. Let $\mathbf{v}_1, \ldots, \mathbf{v}_k \in \mathbb{R}^n$, with $k \geq 2$, be the vertices of $\mathcal{P}$ and consider $\mathbf{t} \in \mathbb{R}^n$. Then:

$$\mathrm{T}(\mathcal{P}, \mathbf{t}) = \mathrm{conv}(\{\mathbf{v}_1 + \mathbf{t}, \ldots, \mathbf{v}_k + \mathbf{t}\}). \tag{2.22}$$

Proof. The proof is immediate from Definition 2.11 and Definition 2.16. ■

Property 2.7: Translation of a H-polytope

Let $\mathcal{P} \subset \mathbb{R}^n$ be a polytope for which a H-representation is known. Let $\boldsymbol{H} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{\theta} \in \mathbb{R}^m$ the matrices inducing $\mathcal{P}$ and $\mathbf{t} \in \mathbb{R}^n$. Then:

$$\mathrm{T}(\mathcal{P}, \mathbf{t}) = \{\mathbf{x} \in \mathbb{R}^n \mid \boldsymbol{H}\mathbf{x} \leq \boldsymbol{\theta} + \boldsymbol{H}\mathbf{t}\}. \tag{2.23}$$

Proof. Let $\mathcal{P} = \{\mathbf{x} \in \mathbb{R}^n \mid \boldsymbol{H}\mathbf{x} \leq \boldsymbol{\theta}\}$ be a polytope with $\boldsymbol{H} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{\theta} \in \mathbb{R}^m$. Let $\mathbf{p}_i \in \partial\mathcal{P}_i$, with $i \in \overline{1, m}$ a point on the i-th border of $\mathcal{P}$, thus verifying $\boldsymbol{h}_i \mathbf{p}_i = \theta_i$ where $\boldsymbol{h}_i$ and θ_i are the i-th rows of $\boldsymbol{H}$ and $\boldsymbol{\theta}$, respectively. The translation of the point $\mathbf{p}_i$ by the vector $\mathbf{t} \in \mathbb{R}^n$ verifies:

$$\boldsymbol{h}_i(\mathbf{p}_i + \mathbf{t}) = \theta_i + \boldsymbol{h}_i \mathbf{t}. \tag{2.24}$$

Thus, if $\mathbf{x} \in \mathbb{R}^n$ is such that $\mathbf{x} \in \mathrm{T}(\partial\mathcal{P}_i, \mathbf{t})$, it has the form $\mathbf{x} = \mathbf{p}_i + \mathbf{t}$ and verifies (2.24). This way, $\mathrm{T}(\partial\mathcal{P}_i, \mathbf{t}) = \{\mathbf{x} \in \mathbb{R}^n \mid \boldsymbol{h}_i \mathbf{x} = \theta_i + \boldsymbol{h}_i \mathbf{t}\}$, $\forall i \in \overline{1, m}$. Then, $\mathrm{T}(\mathcal{P}, \mathbf{t})$ is given by (2.23). ■

Definition 2.17: Minkwoski sum of two sets

The Minkowski sum of two sets $\mathcal{X}, \mathcal{Y} \subset \mathbb{R}^n$ is the set:

$$\mathcal{X} \oplus \mathcal{Y} = \{\mathbf{x} + \mathbf{y} \mid \mathbf{x} \in \mathcal{X}, \mathbf{y} \in \mathcal{Y}\}. \tag{2.25}$$

Remark 2.7: Minkowski sum of two V-polytopes (Ziegler, 2007)
If $\mathcal{X}, \mathcal{Y} \subset \mathbb{R}^n$ are two polytopes for which a V-representation is known, their Minkowski sum can be computed easily. Let $\mathbf{x}_1, \ldots, \mathbf{x}_k, \mathbf{y}_1, \ldots, \mathbf{y}_l \in \mathbb{R}^n$, with $k, l \geq 2$, such that $\mathcal{X} = \text{conv}(\{\mathbf{x}_1, \ldots, \mathbf{x}_k\})$ and $\mathcal{Y} = \text{conv}(\{\mathbf{y}_1, \ldots, \mathbf{y}_l\})$. Then, $\mathcal{X} \oplus \mathcal{Y} = \text{conv}(\mathcal{V})$ where:

$$\mathcal{V} = \{\mathbf{x}_i + \mathbf{y}_j, \forall i \in \overline{1,k}, \forall j \in \overline{1,l}\}. \qquad \lozenge$$

Definition 2.18: Pontryagin difference of two sets

The Pontryagin difference of two sets $\mathcal{X}, \mathcal{Y} \subset \mathbb{R}^n$ is the set:

$$\mathcal{X} \ominus \mathcal{Y} = \{\mathbf{x} \in \mathcal{X} \mid \mathbf{x} + \mathbf{y} \in \mathcal{X}, \forall \mathbf{y} \in \mathcal{Y}\}. \tag{2.26}$$

Example 2.7: Minkowski sum and Pontryagin difference of V-polytopes
Let $\mathcal{P} \subset \mathbb{R}^2$ be the polytope introduced in Example 2.5. Let $\mathcal{Q} \subset \mathbb{R}^2$ be the polytope:

$$\mathcal{Q} = \text{conv}\left(\left\{\mathbf{q}_1 = \begin{bmatrix}-1\\-1\end{bmatrix}, \mathbf{q}_2 = \begin{bmatrix}0\\1\end{bmatrix}, \mathbf{q}_3 = \begin{bmatrix}1\\-1\end{bmatrix}\right\}\right).$$

Figure 2.6 presents the Minkowski sum $\mathcal{P} \oplus \mathcal{Q}$ of $\mathcal{P}$ and $\mathcal{Q}$. As per Remark 2.7, it is obtained by adding all the vertices of $\mathcal{Q}$ to the vertices of $\mathcal{P}$ and by taking the convex hull of all the obtained points. It can be done graphically by centering the set $\mathcal{Q}$ on all the vertices of $\mathcal{P}$ and by joining the maximum points.

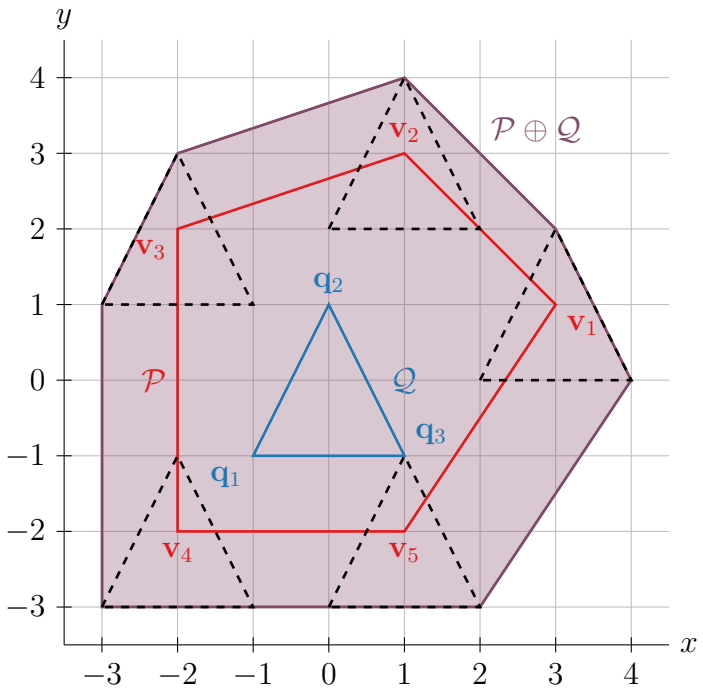


Figure 2.6: Minkowski sum of two polytopes in $\mathbb{R}^2$.

Figure 2.7 presents the Pontryagin difference $\mathcal{P} \ominus \mathcal{Q}$ of $\mathcal{P}$ and $\mathcal{Q}$. In this figure, the set $\mathcal{Q}$ is centered on all the vertices of $\mathcal{P} \ominus \mathcal{Q}$ to show that the sum of an element from $\mathcal{P} \ominus \mathcal{Q}$ and one from $\mathcal{Q}$ is an element of $\mathcal{P}$.

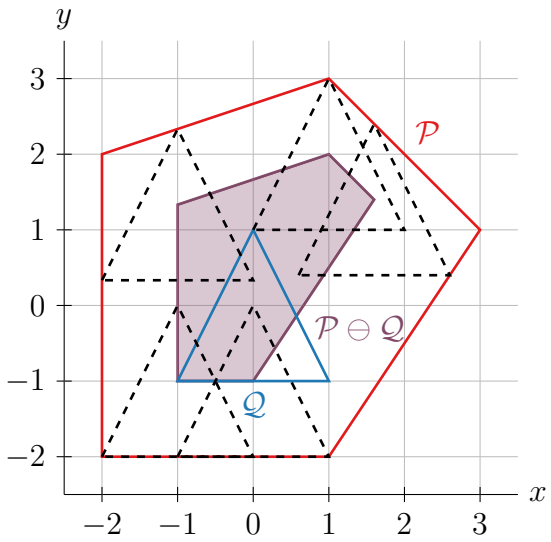


Figure 2.7: Pontryagin difference of two polytopes in $\mathbb{R}^2$.

From the graphical representations of Example 2.7, a special case of Minkowski sum and Pontryagin difference can be formulated.

Property 2.8: Minkowski sum and translation

Let $\mathcal{X} \subset \mathbb{R}^n$ be a set and $\mathbf{t} \in \mathbb{R}^n$ a vector. Then, the Minkowski sum of $\mathcal{X}$ and $\{\mathbf{t}\}$ is the translation of $\mathcal{X}$ by $\mathbf{t}$:

$$\mathcal{X} \oplus \{\mathbf{t}\} = \mathrm{T}(\mathcal{X}, \mathbf{t}). \tag{2.27}$$

Proof. The proof is immediate from Definitions 2.16 and 2.17. ■

Remark 2.8: Minkowski sum of a polytope and a singleton
From Properties 2.7 and 2.8, if $\mathcal{P} \subset \mathbb{R}^n$ is a polytope for which a H-representation is known and $\mathbf{t} \in \mathbb{R}^n$ is a vector, the H-representation of $\mathcal{P} \oplus \{\mathbf{t}\}$ is:

$$\mathcal{P} \oplus \{\mathbf{t}\} = \{\mathbf{x} \in \mathbb{R}^n \mid \boldsymbol{H}\mathbf{x} \leq \boldsymbol{\theta} + \boldsymbol{H}\mathbf{t}\} \tag{2.28}$$

with $\boldsymbol{H} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{\theta} \in \mathbb{R}^m$ the matrices inducing $\mathcal{P}$. ◊

Definition 2.19: Scaling of a set

Let $\mathcal{X} \in \mathbb{R}^n$ be a set and consider $\lambda \in \mathbb{R}_+$. The λ-scaled version of the set $\mathcal{X}$ is:

$$\lambda\mathcal{X} = \{\mathbf{y} \in \mathbb{R}^n \mid \mathbf{y} = \lambda\mathbf{x}, \mathbf{x} \in \mathcal{X}\}. \tag{2.29}$$

Property 2.9: V-representation of a scaled polytope

Let $\mathcal{P} \subset \mathbb{R}^n$ be a polytope for which a V-representation is known. Let $\mathbf{v}_1, \ldots, \mathbf{v}_k \in \mathbb{R}^n$, with $k \geq 2$, be the vertices of $\mathcal{P}$ and consider $\lambda \in \mathbb{R}_+$. The V-representation of $\lambda\mathcal{P}$ is then:

$$\lambda\mathcal{P} = \mathrm{conv}(\{\lambda\mathbf{v}_1, \ldots, \lambda\mathbf{v}_k\}). \tag{2.30}$$

Proof. The proof is immediate from Definition 2.11 and Definition 2.19. ■

Property 2.10: H-representation of a scaled polytope

Let $\mathcal{P} \subset \mathbb{R}^n$ be a polytope for which a H-representation is known. Let $\boldsymbol{H} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{\theta} \in \mathbb{R}^m$ be the matrices inducing $\mathcal{P}$ and consider $\lambda \in \mathbb{R}_+$. The H-representation of $\lambda\mathcal{P}$ is then:

$$\lambda\mathcal{P} = \{\mathbf{x} \in \mathbb{R}^n \mid \boldsymbol{H}\mathbf{x} \leq \lambda\boldsymbol{\theta}\}. \tag{2.31}$$

Proof. Let $\mathcal{P} = \{\mathbf{x} \in \mathbb{R}^n \mid \boldsymbol{H}\mathbf{x} \leq \boldsymbol{\theta}\}$ be a polytope with $\boldsymbol{H} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{\theta} \in \mathbb{R}^m$. Let $\mathbf{p}_i \in \partial\mathcal{P}_i$, with $i \in \overline{1, m}$ a point on the i-th border of $\mathcal{P}$, thus verifying $\boldsymbol{h}_i\mathbf{p}_i = \theta_i$ where $\boldsymbol{h}_i$ and θ_i are the i-th rows of $\boldsymbol{H}$ and $\boldsymbol{\theta}$, respectively. The scaling of the point $\mathbf{p}_i$ by $\lambda \in \mathbb{R}_+$ verifies:

$$\boldsymbol{h}_i\lambda\mathbf{p}_i = \lambda\theta_i. \tag{2.32}$$

Thus, if $\mathbf{x} \in \mathbb{R}^n$ is such that $\mathbf{x} \in \lambda\partial\mathcal{P}_i$, it has the form $\mathbf{x} = \lambda\mathbf{p}_i$ and verifies (2.32). This way, $\lambda\partial\mathcal{P}_i = \{\mathbf{x} \in \mathbb{R}^n \mid \boldsymbol{h}_i\mathbf{x} = \lambda\theta_i\}$, $\forall i \in \overline{1, m}$. Then, $\lambda\mathcal{P}$ is given by (2.31). ■

Example 2.8: Polytope scaling

Let $\mathcal{P} \subset \mathbb{R}^2$ *be the polytope induced by the matrices* $\boldsymbol{H} = \begin{bmatrix} -1 & -3 & 3 & 5 \\ 4 & -2 & -7 & 1 \end{bmatrix}^\top$ *and* $\boldsymbol{\theta} = \begin{bmatrix} 6 & 5 & 11 & 12 \end{bmatrix}^\top$. *Figure 2.8 presents the polytope* $\mathcal{P}$ *as well as its versions scaled by* $\lambda = 1.5$ *and* $\lambda = 0.5$.

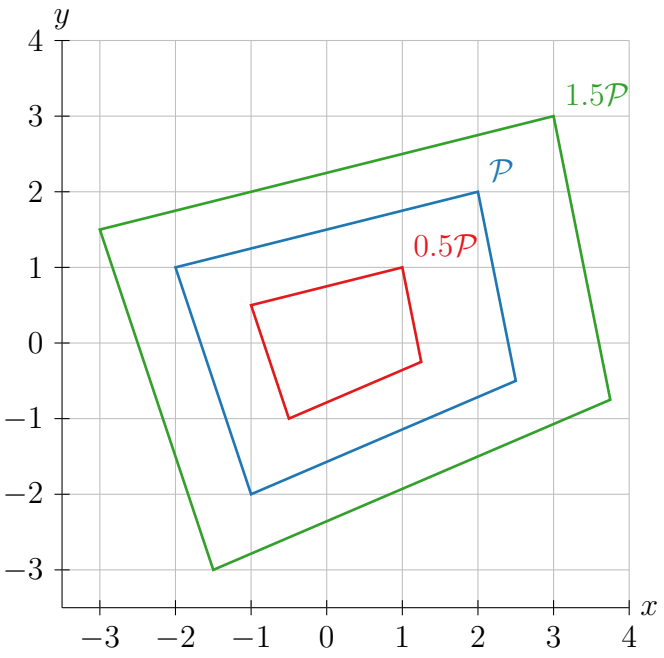


Figure 2.8: Different scalings of a polytope in $\mathbb{R}^2$.

2.2.4 Sets in control theory

The sets and the operations introduced in the previous paragraphs are used in control theory in conjunction with properties of set invariance that are detailed in the following. This paragraph thus introduces definitions of different kinds of invariance properties for dynamical systems.

The first two definitions are given for an autonomous discrete-time linear time invariant (LTI) system. They are related to positive invariance of a set which is one of the most important invariance property in control theory.

Definition 2.20: Positive invariant set (Blanchini and Miani, 2015)

Consider an autonomous discrete-time linear time invariant system with the dynamics:

$$\mathbf{x}(k+1) = \boldsymbol{A}\mathbf{x}(k) \tag{2.33}$$

where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is a Schur matrix and $\mathbf{x}(k) \in \mathbb{R}^n$ for all $k \in \mathbb{N}$. A set $\mathcal{X} \subset \mathbb{R}^n$ is called *positive invariant* for the dynamics (2.33) if $\mathbf{x}(k+1) \in \mathcal{X}$ for all $\mathbf{x}(k) \in \mathcal{X}$ or if:

$$\boldsymbol{A}\mathcal{X} \subseteq \mathcal{X}. \tag{2.34}$$

Definition 2.21: Minimal positive invariant set (Blanchini and Miani, 2015)

The set $\mathcal{X} \subset \mathbb{R}^n$ is called *minimal positive invariant* (mPI) for the dynamics (2.33) if it is the intersection of all the positive invariant sets for the dynamics (2.33).

Definition 2.22: Robustly positive invariant set (Blanchini and Miani, 2015)

Consider a perturbed autonomous discrete-time linear time-invariant system with the dynamics:

$$\mathbf{x}(k+1) = \boldsymbol{A}\mathbf{x}(k) + \mathbf{w}(k) \tag{2.35}$$

where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is a Schur matrix, $\mathbf{x}(k) \in \mathbb{R}^n$ for all $k \in \mathbb{N}$ and $\mathbf{w}(k) \in \mathcal{W} \subset \mathbb{R}^n$ a perturbation signal. A set $\mathcal{X} \subset \mathbb{R}^n$ is called *robustly positive invariant* (RPI) for the dynamics (2.35) if $\mathbf{x}(k+1) \in \mathcal{X}$ for all $\mathbf{x}(k) \in \mathcal{X}$ and $\mathbf{w}(k) \in \mathcal{W}$ or if:

$$\boldsymbol{A}\mathcal{X} \oplus \mathcal{W} \subseteq \mathcal{X}. \tag{2.36}$$

Definition 2.23: Minimal RPI set (Blanchini and Miani, 2015)

The set $\mathcal{X} \subset \mathbb{R}^n$ is called *minimal robustly positive invariant* (mRPI) for the dynamics (2.35) if it is the intersection of all the RPI sets for the dynamics (2.35).

Other works, such as Basile and Marro (1969), have presented the concept of set invariance when the dynamics are not autonomous.

Definition 2.24: Controlled invariant set (Basile and Marro, 1969)

Consider a discrete-time linear time-invariant system with the dynamics:

$$\mathbf{x}(k+1) = \boldsymbol{A}\mathbf{x}(k) + \boldsymbol{B}\mathbf{u}(k) \tag{2.37}$$

where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{B} \in \mathbb{R}^{n\times m}$, $\mathbf{x}(k) \in \mathbb{R}^n$ and $\mathbf{u}(k) \in \mathbb{R}^m$ for all $k \in \mathbb{N}$. The pair $(\boldsymbol{A}, \boldsymbol{B})$ is assumed to be controllable. A set $\mathcal{X} \subset \mathbb{R}^n$ is called *controlled invariant* for the dynamics (2.37) if for all $\mathbf{x}(k) \in \mathcal{X}$, there exists a control law $\mathbf{u}(k) \in \mathcal{U} \subset \mathbb{R}^m$ such that $\mathbf{x}(k+1) \in \mathcal{X}$.

Finally, the scaling of sets can be used to define a last kind of invariance property.

Definition 2.25: Controlled λ-contractive set (Blanchini and Miani, 2015)

A set $\mathcal{X} \subset \mathbb{R}^n$ is called *controlled λ-contractive*, with $\lambda \in [0, 1)$, for the dynamics (2.37) if for all $\mathbf{x}(k) \in \mathcal{X}$, there exists a control law $\mathbf{u}(k) \in \mathcal{U} \subset \mathbb{R}^m$ such that $\mathbf{x}(k+1) \in \lambda\mathcal{X}$.
If $\lambda = 1$, $\mathcal{X}$ is controlled invariant.

The last important element of this section is related to the construction of such invariant sets. Several algorithms exist to compute RPI and mRPI sets (Raković et al., 2005, Kofman et al., 2007, Olaru et al., 2010, Trodden, 2016) or even controlled invariant sets (Fiacchini and Alamir, 2017). Most of these algorithms are based on H-polytopes, leading to one of their main drawbacks, i.e. its complexity. Indeed, these algorithms are iterative, which can make the number of vertices of the considered polytopes quickly increase. This is why it is often better to compute invariant sets offline and use the result online. In order to reduce the complexity of the invariant set computation, ellipsoidal sets could be used. However, such sets would induce quadratic constraints in the optimization problems of the model predictive controllers of Section 4.1, while the constraints remain linear with H-polytopes. Then, while the computation of invariant sets is simplified with the use of ellipsoidal sets, the computation of the input signal with a model predictive controller is more complicated if an ellipsoidal set is used instead of a polytopic set.

A method to compute polytopic invariant sets is based on Kofman et al. (2007) followed by a procedure coming from Olaru et al. (2010). Indeed, Kofman et al. (2007) provides a useful theorem to build an initial RPI set from the dynamics of a perturbed autonomous discrete-time linear time invariant system when the perturbation is bounded.

Theorem 2.5: Initial RPI set (Kofman et al., 2007)

Consider the system:

$$\mathbf{x}(k+1) = \boldsymbol{A}\mathbf{x}(k) + \mathbf{w}(k)$$

together with the notations from Definition 2.22. The perturbation vector is assumed to be such that $\|\mathbf{w}(k)\|_\infty \leq \overline{w}$ for all $k \in \mathbb{N}$, where $\overline{w} \in \mathbb{R}_+$. Let $\boldsymbol{A} = \boldsymbol{V}\boldsymbol{J}\boldsymbol{V}^{-1}$, with $\boldsymbol{V}, \boldsymbol{J} \in \mathbb{R}^{n\times n}$, be the Jordan decomposition of $\boldsymbol{A}$. Then the set:

$$\mathcal{S} = \left\{ \mathbf{x} \in \mathbb{R}^n \;\middle|\; \begin{bmatrix} \boldsymbol{V}^{-1} \\ -\boldsymbol{V}^{-1} \end{bmatrix} \mathbf{x} \leq \mathbf{1}_{2\times 1} \otimes (\boldsymbol{I}_n - |\boldsymbol{J}|)^{-1} \left| \boldsymbol{V}^{-1}\overline{w}\mathbf{1}_{n\times 1} \right| \right\} \tag{2.38}$$

is invariant for the dynamics (2.35).

From this initial shape, Olaru et al. (2010) propose an iterative algorithm to build an ε-approximation of the mRPI set, where $\varepsilon \in \mathbb{R}_+^*$, recalled in Algorithm 2.1. In this algorithm, $\mathscr{E}(\boldsymbol{I}_n, \varepsilon) \subset \mathbb{R}^n$ is a centered ellipsoid and $\mathbb{B}^n$ is a unitary box of $\mathbb{R}^n$ as defined in Definition 2.13. For a complete version, the reader can refer to Olaru et al. (2010).

Algorithm 2.1 provides an approximation of the mRPI set depending on the chosen value of the parameter ε. However, this method can end up being time consuming depending on the dimension of the state space n and on the matrix $\boldsymbol{A}$.

Trodden (2016) provides a faster method, while more conservative, than the one proposed above. However, this method being based on the maximization of a cost

Algorithm 2.1: ε-approximation of the mRPI set

Input: The matrix $\boldsymbol{A}$, the disturbance bound $\overline{w}$, the scalar $\varepsilon \in \mathbb{R}_+^*$

Compute the Jordan decomposition of $\boldsymbol{A}$;
Build the initial RPI set $\mathcal{S}$ with (2.38);
Initialize $\mathcal{S}_\varepsilon = \mathcal{S}$;
Initialize $l = 0$;
while $\boldsymbol{A}^{l+1}\mathcal{S}_\varepsilon \not\subset \mathscr{E}(\boldsymbol{I}_n, \varepsilon)$ **do**
  $l = l + 1$;
  $\mathcal{S}_\varepsilon = \boldsymbol{A}\mathcal{S}_\varepsilon \oplus \overline{w}\mathbb{B}^n$;
end

Output: The ε-approximation $\mathcal{S}_\varepsilon$ of the mRPI set

function under constraints, if the cost function is unbounded, there is no guarantee to get a result. To present the optimization problem proposed by Trodden (2016), it is necessary to introduce notations for some elements of the problem. First, the H-representation of $\mathbb{B}^n$ as used in Algorithm 2.1 is:

$$\mathbb{B}^n = \{\mathbf{x} \in \mathbb{R}^n \mid \boldsymbol{B}\mathbf{x} \leq \boldsymbol{\theta}_{\mathbb{B}^n}\}$$

and second, the set $\mathcal{S}$ from (2.38) has for simplified H-representation:

$$\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^n \mid \boldsymbol{S}\mathbf{x} \leq \boldsymbol{\theta}_{\mathcal{S}}\}.$$

Then, the approximation of the mRPI set $\mathcal{S}^*$ have for H-representation:

$$\mathcal{S}^* = \{\mathbf{x} \in \mathbb{R}^n \mid \boldsymbol{S}\mathbf{x} \leq \boldsymbol{\theta}^*\}$$

with $\boldsymbol{\theta}^* = \mathbf{c}^* + \mathbf{d}^*$ such that (Trodden, 2016):

$$(\mathbf{c}^*, \mathbf{d}^*) = \underset{\substack{c_i, d_i, \boldsymbol{\xi}_i, \boldsymbol{\omega}_i \\ \forall i \in \overline{1,2n}}}{\text{maximize}} \quad \sum_{i=1}^{2n} c_i + d_i \tag{2.39a}$$

subject to

$$c_i \leq \boldsymbol{s}_i \boldsymbol{A}\boldsymbol{\xi}_i, \qquad \forall i \in \overline{1,2n}, \tag{2.39b}$$
$$\boldsymbol{S}\boldsymbol{\xi}_i \leq \mathbf{c} + \mathbf{d}, \qquad \forall i \in \overline{1,2n}, \tag{2.39c}$$
$$d_i \leq \boldsymbol{s}_i \boldsymbol{\omega}_i, \qquad \forall i \in \overline{1,2n}, \tag{2.39d}$$
$$\boldsymbol{B}\boldsymbol{\omega}_i \leq \overline{w}\boldsymbol{\theta}_{\mathbb{B}^n}, \qquad \forall i \in \overline{1,2n} \tag{2.39e}$$

where $\boldsymbol{s}_i$, c_i and d_i, with $i \in \overline{1,2n}$, are the rows of $\boldsymbol{S}$, $\mathbf{c}$ and $\mathbf{d}$, respectively, and $\boldsymbol{\xi}_i \in \mathbb{R}^n$ and $\boldsymbol{\omega}_i \in \mathbb{R}^n$, with $i \in \overline{1,2n}$, are optimization variables. Thus, as mentioned earlier, if the problem is bounded, (2.39) provides an approximation of the mRPI set which can be more conservative than the one provided by Olaru et al. (2010), but is obtained faster. Example 2.9 provides an academic example of construction of the mRPI set with the two methods.

Example 2.9: Comparison of two approximation of the mRPI set of a system

Consider the system $\mathbf{x}(k+1) = \boldsymbol{A}\mathbf{x}(k) + \mathbf{w}(k)$, *with* $\boldsymbol{A} = \begin{bmatrix} 0.8 & 0.1 \\ 0.3 & 0.5 \end{bmatrix}$ *and* $\|\mathbf{w}(k)\|_\infty \leq \overline{w} = 0.1$ *for* $k \in \mathbb{N}$. *The Jordan decomposition of* $\boldsymbol{A}$ *is:*

$$\boldsymbol{A} = \boldsymbol{V}\boldsymbol{J}\boldsymbol{V}^{-1}, \text{ with } \boldsymbol{V} = \begin{bmatrix} -0.2638 & 1.2638 \\ 1 & 1 \end{bmatrix} \text{ and } \boldsymbol{J} = \begin{bmatrix} 0.4209 & 0 \\ 0 & 0.8791 \end{bmatrix}.$$

Using Theorem 2.5, the initial RPI set is:

$$\mathcal{S} = \left\{ \mathbf{x} \in \mathbb{R}^2 \;\middle|\; \begin{bmatrix} -0.6547 & 0.8273 \\ 0.6547 & 0.1727 \\ 0.6547 & -0.8273 \\ -0.6547 & -0.1727 \end{bmatrix} \mathbf{x} \leq \begin{bmatrix} 0.0298 \\ 0.6845 \\ 0.0298 \\ 0.6845 \end{bmatrix} \right\}.$$

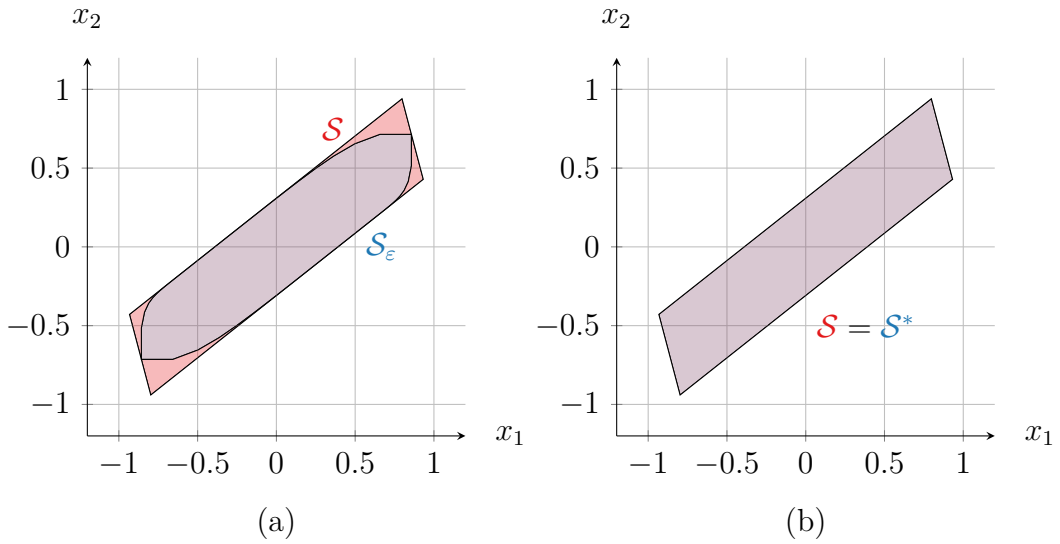


Figure 2.9: Comparison of two approximations of the mRPI set of a system in $\mathbb{R}^2$.

Figure 2.9 presents in red the initial RPI set defined above. The left plot (a) shows the ε-approximation of the mRPI set $\mathcal{S}_\varepsilon$ with $\varepsilon = 10^{-5}$ in blue obtained with Algorithm 2.1. On the right plot (b) is drawn the approximation of the mRPI set $\mathcal{S}^*$ in blue obtained by solving (2.39).

The fact that $\mathcal{S}^*$ is more conservative than $\mathcal{S}_\varepsilon$ is obvious from Figure 2.9 since $\mathcal{S}_\varepsilon \subset \mathcal{S}$, while $\mathcal{S}^* = \mathcal{S}$. However, it is not obvious from the plot that the V-representation of $\mathcal{S}_\varepsilon$ contains 149 vertices, its H-representation is composed of 79 inequalities and $s = 88$ iterations are needed to compute $\mathcal{S}_\varepsilon$. The approximation $\mathcal{S}^*$ has the same V-representation and H-representation as $\mathcal{S}$, i.e. 4 vertices and 4 inequalities.

Finally, a trade-off between conservativeness and complexity can be made by combining the methods from Olaru et al. (2010) and Trodden (2016). For this, the while loop in Algorithm 2.1 can be modified to run for a limited number of iterations arbitrarily chosen or until a maximal number of vertices or inequalities is reached. Then, the H-representation of the obtained set can be used as input to (2.39).

2.3 Multi-vehicle system description

The elements introduced in the previous sections are used in the development of control algorithms for multi-agent systems (MAS). For this, the dynamics of the agents composing the MAS are introduced in the present section as well as the assumptions necessary for the development of the control algorithms proposed in this thesis.

Let Σ be a multi-agent system composed of N agents. Each agent obeys discrete-time LTI dynamics:

$$\mathbf{x}_i(k+1) = \boldsymbol{A}_i\mathbf{x}_i(k) + \boldsymbol{B}_i\mathbf{u}_i(k) \tag{2.40a}$$
$$\mathbf{y}_i(k) = \boldsymbol{C}_i\mathbf{x}_i(k) \tag{2.40b}$$

where $\mathbf{x}_i \in \mathbb{R}^{n_i}$, $\mathbf{u}_i \in \mathbb{R}^{m_i}$, $\mathbf{y}_i \in \mathbb{R}^{p_i}$, $\boldsymbol{A}_i \in \mathbb{R}^{n_i \times n_i}$, $\boldsymbol{B}_i \in \mathbb{R}^{n_i \times m_i}$ and $\boldsymbol{C}_i \in \mathbb{R}^{p_i \times n_i}$, with $i \in \overline{1, N}$. For an agent $i \in \overline{1, N}$, the vector $\mathbf{x}_i$ is called the state vector, $\mathbf{u}_i$ is called the input vector and $\mathbf{y}_i$ is called the output vector. The output set $\mathcal{Y}_i$ is a subset of the state space $\mathcal{X}_i$ and is of lower or equal dimension, i.e. $p_i \leq n_i$. In the literature, several types of multi-vehicle systems (MVS) are considered. Some examples include MVS composed of several unmanned aerial vehicles (Chevet et al., 2020b), unmanned ground vehicles (Kamel et al., 2020) or even both unmanned ground and aerial vehicles (Sharifi et al., 2014).

Assumption 2.1: Controllability

The pair $(\boldsymbol{A}_i, \boldsymbol{B}_i)$ *is controllable for all* $i \in \overline{1, N}$.

Assumption 2.2: Observability

The pair $(\boldsymbol{A}_i, \boldsymbol{C}_i)$ *is observable for all* $i \in \overline{1, N}$.

The goal of the control algorithms that are presented in the following chapters of this thesis is to control the position of vehicles in a two-dimensional space. Hence, each agent of Σ is a vehicle which evolves in an output space subset of $\mathbb{R}^2$, i.e. $p_i = p = 2\ \forall i \in \overline{1, N}$. The output of an agent $i \in \overline{1, N}$ is then the Cartesian position $\mathbf{y}_i(k) = \begin{bmatrix} x_i(k) & y_i(k) \end{bmatrix}^\top$ of agent i in the plane $\mathbb{R}^2$.

Assumption 2.3: Output space

All agents of Σ *share the same output space, i.e.* $\mathcal{Y}_i = \mathcal{Y}_j = \mathcal{Y} \subset \mathbb{R}^2$ *for all* $i, j \in \overline{1, N}$, $i \neq j$.

Assumption 2.4: Structure of the state vector

The state vector $\mathbf{x}_i(k)$ *of an agent* $i \in \overline{1, N}$ *of* Σ *is composed of the Cartesian position* $\mathbf{y}_i(k) = \begin{bmatrix} x_i(k) & y_i(k) \end{bmatrix}^\top$ *of agent* i *in the plane* $\mathbb{R}^2$ *and additional states (e.g. the speed of the agent) such that:*

$$\mathbf{x}_i(k) = \begin{bmatrix} x_i(k) \\ \mathbf{x}_{1,i}(k) \\ y_i(k) \\ \mathbf{x}_{2,i}(k) \end{bmatrix}$$

where $\mathbf{x}_{1,i} \in \mathbb{R}^{n_{1,i}}$ *and* $\mathbf{x}_{2,i} \in \mathbb{R}^{n_{2,i}}$, *with* $n_{1,i} + n_{2,i} + 2 = n_i$, $n_{1,i}, n_{2,i} \geq 0$.

Remark 2.9: Additional states
The additional states $\mathbf{x}_{1,i}$ and $\mathbf{x}_{2,i}$ appearing in Assumption 2.4 could refer to the speed of the agents, their acceleration, etc. ♢

Remark 2.10: Output matrix
Given the structure of the state vector $\mathbf{x}_i(k)$ of agent $i \in \overline{1, N}$ given in Assumption 2.4,

the output matrix $\boldsymbol{C}_i$ is:

$$\boldsymbol{C}_i = \begin{bmatrix} 1 & \mathbf{0}_{1\times n_{1,i}} & 0 & \mathbf{0}_{1\times n_{2,i}} \\ 0 & \mathbf{0}_{1\times n_{1,i}} & 1 & \mathbf{0}_{1\times n_{2,i}} \end{bmatrix}$$

for all $i \in \overline{1,N}$. The case $n_{1,i} = 0$ and/or $n_{2,i} = 0$ is handled by removing the corresponding column in $\boldsymbol{C}_i$. ◊

The evolution of each agent of Σ is conditioned by the position of the other agents in $\mathcal{Y}$.

Assumption 2.5: Knowledge of environment

Each agent of Σ knows, at all time, the position of the other agents of Σ.

To make the notations more compact, the dynamics of each agent can be aggregated in one single state-space model:

$$\mathbf{x}(k+1) = \boldsymbol{A}_\Sigma \mathbf{x}(k) + \boldsymbol{B}_\Sigma \mathbf{u}(k) \tag{2.41a}$$
$$\mathbf{y}(k) = \boldsymbol{C}_\Sigma \mathbf{x}(k) \tag{2.41b}$$

where $\mathbf{x} = \begin{bmatrix}\mathbf{x}_1^\top & \cdots & \mathbf{x}_N^\top\end{bmatrix}^\top \in \mathbb{R}^{\sum_{i\in\overline{1,N}} n_i}$, $\mathbf{u} = \begin{bmatrix}\mathbf{u}_1^\top & \cdots & \mathbf{u}_N^\top\end{bmatrix}^\top \in \mathbb{R}^{\sum_{i\in\overline{1,N}} m_i}$, $\mathbf{y} = \begin{bmatrix}\mathbf{y}_1^\top & \cdots & \mathbf{y}_N^\top\end{bmatrix}^\top \in \mathbb{R}^{2N}$, $\boldsymbol{A}_\Sigma = \operatorname{diag}(\boldsymbol{A}_1, \ldots, \boldsymbol{A}_N)$, $\boldsymbol{B}_\Sigma = \operatorname{diag}(\boldsymbol{B}_1, \ldots, \boldsymbol{B}_N)$ and $\boldsymbol{C}_\Sigma = \operatorname{diag}(\boldsymbol{C}_1, \ldots, \boldsymbol{C}_N)$.

In the formulation given by (2.41), the shape of $\boldsymbol{A}_\Sigma$, $\boldsymbol{B}_\Sigma$ and $\boldsymbol{C}_\Sigma$ show no dependence between the state of the agents. However, as mentioned earlier, some dependency appears from constraints given later in Chapters 3 to 5 since the evolution of each agent of Σ are conditioned by the position of the other agents in $\mathcal{Y}$.

Thus, the individual control input of an agent of Σ depends on the position of the agents of Σ and it is:

$$\mathbf{u}_i(k) = \mathbf{u}_i(\mathbf{y}_1(k), \ldots, \mathbf{y}_N(k)) \tag{2.42}$$

and the aggregated control input is $\mathbf{u}(k) = \mathbf{u}(\mathbf{y}(k))$. In other words, despite the absence of coupling between the agents in open-loop, the closed-loop strategy induces a coupling between the agents, hence the necessity of the development of centralized, distributed or decentralized multi-agent system control strategies.

It can be noted that, depending on the considered control strategy, the control signal $\mathbf{u}_i(k)$ of agent $i \in \overline{1,N}$ can depend only on a part of the agents denoted by $\mathrm{N}_i(k) \subseteq \overline{1,N}$. Thus, (2.42) is modified to:

$$\mathbf{u}_i(k) = \mathbf{u}_i\Big(\mathbf{y}_{\nu_1}(k), \ldots, \mathbf{y}_{\nu_{|\mathrm{N}_i(k)|}}(k)\Big)$$

where $\{\nu_1, \ldots, \nu_{|\mathrm{N}_i(k)|}\} = \mathrm{N}_i(k)$.

Remark 2.11: Set of neighbors

The notation $\mathrm{N}_i(k)$ designates the set of neighbors of the agent i, with $i \in \overline{1,N}$, in the subsequent chapters. ◊

Assumption 2.6: Homogeneity

The dynamics of all agents are the same, i.e. $\boldsymbol{A}_i = \boldsymbol{A}_j$, $\boldsymbol{B}_i = \boldsymbol{B}_j$ and $\boldsymbol{C}_i = \boldsymbol{C}_j$ for all $i, j \in \overline{1,N}$.

Assumption 2.6 is made to simplify the presentation of the results in the following chapters. However, the extension of these results to a heterogeneous MAS is easily obtainable.

2.4 Voronoi tessellation and formation configuration

The definitions about polytopes presented in Section 2.2.2 are used in the present section to introduce the Voronoi tessellation, which is a fundamental tool for the control algorithms presented in Chapters 3 to 5. The Voronoi tessellation consists in a partition of a set in independent subsets based on generator points from this set introduced by Dirichlet (1850) and Voronoï (1908).

Section 2.4.1 presents the basic definition and construction of the Voronoi tessellation (Voronoï, 1908). In Section 2.4.2, some generalizations of the basic definition on which the algorithms presented in this thesis are based are introduced. Finally, Section 2.4.3 defines the Chebyshev configuration of a multi-vehicle system based on the elements proposed in Sections 2.4.1 and 2.4.2.

2.4.1 Conventional Voronoi tessellation

As mentioned in the introduction of this section, the Voronoi tessellation is a way to partition a set into independent subsets based on a finite number of points present in this set, called *generators*. Thus, let $\mathcal{Y} = \{\mathbf{x} \in \mathbb{R}^n \mid \boldsymbol{H}_{\mathcal{Y}}\mathbf{x} \leq \boldsymbol{\theta}_{\mathcal{Y}}\}$, with $\boldsymbol{H}_{\mathcal{Y}} \in \mathbb{R}^{s\times n}$, $\boldsymbol{\theta}_{\mathcal{Y}} \in \mathbb{R}^s$ and $s \in \mathbb{N}$, be a polytopic subset of $\mathbb{R}^n$. Let $\mathbf{y}_1, \ldots, \mathbf{y}_N \in \mathcal{Y}$ be a set of points of $\mathcal{Y}$ such that $\mathbf{y}_i \neq \mathbf{y}_j$ for all $i, j \in \overline{1, N}$, $i \neq j$. The goal is to obtain subsets $\mathcal{V}_i$, with $i \in \overline{1, N}$ of $\mathcal{Y}$ such that $\mathbf{y}_i \in \mathcal{V}_i$ and:

$$\mathcal{Y} = \bigcup_{i=1}^{N} \mathcal{V}_i, \text{ with } \mathcal{V}_i \cap \mathcal{V}_j = \emptyset, \forall i, j \in \overline{1, N}, i \neq j. \tag{2.43}$$

A simple way to partition $\mathcal{Y}$ following (2.43) is to consider that $\mathcal{V}_i$ is the set of all points of $\mathcal{Y}$ that are closer to the generator $\mathbf{y}_i$ than to all the other generators $\mathbf{y}_j$ with respect to a given norm.

Definition 2.26: Voronoi cell

Let $\mathcal{Y}$ be a polytopic subset of $\mathbb{R}^n$. Let $\mathbf{y}_1, \ldots, \mathbf{y}_N \in \mathcal{Y}$, with $N \in \mathbb{N}$, be the *generator points* of the Voronoi tessellation. Then, the Voronoi cell $\mathcal{V}_i$ of a generator point $\mathbf{y}_i$ is the set:

$$\mathcal{V}_i = \left\{\mathbf{y} \in \mathcal{Y} \,\middle|\, \|\mathbf{y} - \mathbf{y}_i\|_2 \leq \|\mathbf{y} - \mathbf{y}_j\|_2, \forall j \in \overline{1, N}, i \neq j\right\}. \tag{2.44}$$

By considering the definition of $\|\mathbf{y}\|_2 = \sqrt{\mathbf{y}^\top\mathbf{y}}$, expression (2.44) can be transformed into:

$$\mathcal{V}_i = \left\{\mathbf{y} \in \mathcal{Y} \,\middle|\, (\mathbf{y}_j - \mathbf{y}_i)^\top\mathbf{y} \leq \frac{1}{2}\left(\|\mathbf{y}_j\|_2^2 - \|\mathbf{y}_i\|_2^2\right), \forall j \in \overline{1, N}, i \neq j\right\} \tag{2.45}$$

which is the H-representation of a polytope. Hence, to take the notations from Definition 2.9:

$$\boldsymbol{H}_{\mathcal{V}_i} = \begin{bmatrix} (\mathbf{y}_1 - \mathbf{y}_i)^\top \\ \vdots \\ (\mathbf{y}_{i-1} - \mathbf{y}_i)^\top \\ (\mathbf{y}_{i+1} - \mathbf{y}_i)^\top \\ \vdots \\ (\mathbf{y}_N - \mathbf{y}_i)^\top \\ \boldsymbol{H}_{\mathcal{Y}} \end{bmatrix} \qquad \boldsymbol{\theta}_{\mathcal{V}_i} = \frac{1}{2} \begin{bmatrix} \|\mathbf{y}_1\|_2^2 - \|\mathbf{y}_i\|_2^2 \\ \vdots \\ \|\mathbf{y}_{i-1}\|_2^2 - \|\mathbf{y}_i\|_2^2 \\ \|\mathbf{y}_{i+1}\|_2^2 - \|\mathbf{y}_i\|_2^2 \\ \vdots \\ \|\mathbf{y}_N\|_2^2 - \|\mathbf{y}_i\|_2^2 \\ \boldsymbol{\theta}_{\mathcal{Y}} \end{bmatrix}$$

the matrices $\boldsymbol{H}_{\mathcal{Y}}$ and $\boldsymbol{\theta}_{\mathcal{Y}}$ inducing $\mathcal{Y}$ being necessary to keep the property that $\mathcal{V}_i \subset \mathcal{Y}$. As per Remark 2.6, some of the lines of $\boldsymbol{H}_{\mathcal{V}_i}$ and $\boldsymbol{\theta}_{\mathcal{V}_i}$ are redundant and can be omitted.

By going back to the geometrical interpretation of a H-representation introduced in Section 2.2.2, this can be seen as the collection of inequalities defining half-spaces. Let $\partial\mathcal{V}_{ij}$ denote the border of $\mathcal{V}_i$ induced by the point $\mathbf{y}_j$. This border is a hyperplane:

$$\partial\mathcal{V}_{ij} = \left\{ \mathbf{y} \in \mathbb{R}^n \;\middle|\; (\mathbf{y}_j - \mathbf{y}_i)^\top \mathbf{y} = \frac{1}{2}\left(\|\mathbf{y}_j\|_2^2 - \|\mathbf{y}_i\|_2^2\right) \right\} \tag{2.46}$$

Such a hyperplane is defined by a normal vector and a point it passes by. The general representation of a hyperplane $\mathcal{H}$ is:

$$\mathcal{H} = \left\{ \mathbf{x} \in \mathbb{R}^n \;\middle|\; \mathbf{n}^\top \mathbf{x} = \mathbf{n}^\top \mathbf{p} \right\}$$

where $\mathbf{n} \in \mathbb{R}^n$ is the normal vector and $\mathbf{p} \in \mathbb{R}^n$ is a point known to belong to $\mathcal{H}$. From (2.46), it is obvious that the normal vector to $\partial\mathcal{V}_{ij}$ is $\mathbf{y}_j - \mathbf{y}_i$. Then, from (2.46) and the definition of the Voronoi cell (2.45), it is easy to deduce a point by which $\partial\mathcal{V}_{ij}$ passes. This point is the point located at half the distance between $\mathbf{y}_i$ and $\mathbf{y}_j$, i.e. $(\mathbf{y}_i + \mathbf{y}_j)/2$.

This geometric interpretation leads to an easy way to graphically represent a Voronoi tessellation in $\mathbb{R}^2$ or $\mathbb{R}^3$. A border $\partial\mathcal{V}_{ij}$ of a cell $\mathcal{V}_i$ induced by a point $\mathbf{y}_j$ is then the bisecting line (in $\mathbb{R}^2$) or bisecting plane (in $\mathbb{R}^3$) of the line segment between $\mathbf{y}_i$ and $\mathbf{y}_j$.

Example 2.10: Construction of a Voronoi tessellation in $\mathbb{R}^2$

Let $\mathcal{Y} \subset \mathbb{R}^2$ be a polytope such that $\mathcal{Y} = 5\mathbb{B}^2$. Let $\mathbf{y}_1 = \begin{bmatrix} 4 & 4 \end{bmatrix}^\top$, $\mathbf{y}_2 = \begin{bmatrix} 3 & -4 \end{bmatrix}^\top$, $\mathbf{y}_3 = \begin{bmatrix} 1 & 0 \end{bmatrix}^\top$, $\mathbf{y}_4 = \begin{bmatrix} -2 & 2 \end{bmatrix}^\top$ and $\mathbf{y}_5 = \begin{bmatrix} -3 & -2 \end{bmatrix}^\top$ be 5 points of $\mathcal{Y}$.

Figure 2.10 presents the construction of the Voronoi cell $\mathcal{V}_3$. The points $\mathbf{p}_{31}$, $\mathbf{p}_{32}$, $\mathbf{p}_{34}$ and $\mathbf{p}_{35}$ are the middle points of the line segments joining $\mathbf{y}_3$ to $\mathbf{y}_1$, $\mathbf{y}_2$, $\mathbf{y}_4$ and $\mathbf{y}_5$, respectively. The normal vectors are also drawn. Finally, all the borders $\partial\mathcal{V}_{31}$, $\partial\mathcal{V}_{32}$, $\partial\mathcal{V}_{34}$ and $\partial\mathcal{V}_{35}$ are constructed as described above with the geometrical elements previously introduced.

The cell $\mathcal{V}_3$ is then constructed as the intersection of all the half-spaces defined by the hyperplanes $\partial\mathcal{V}_{31}$, $\partial\mathcal{V}_{32}$, $\partial\mathcal{V}_{34}$ and $\partial\mathcal{V}_{35}$. It is also limited on the right by the fact that $\mathcal{V}_3 \subset \mathcal{Y}$.

The other cells are obtained the same way as $\mathcal{V}_3$. However, it can be seen on the figure that the construction of several borders can be avoided. For example, the border $\partial\mathcal{V}_{51}$ is redundant with the border $\partial\mathcal{V}_{53}$ which is more restrictive. The resulting Voronoi tessellation of $\mathcal{Y}$ is presented in Figure 2.11.

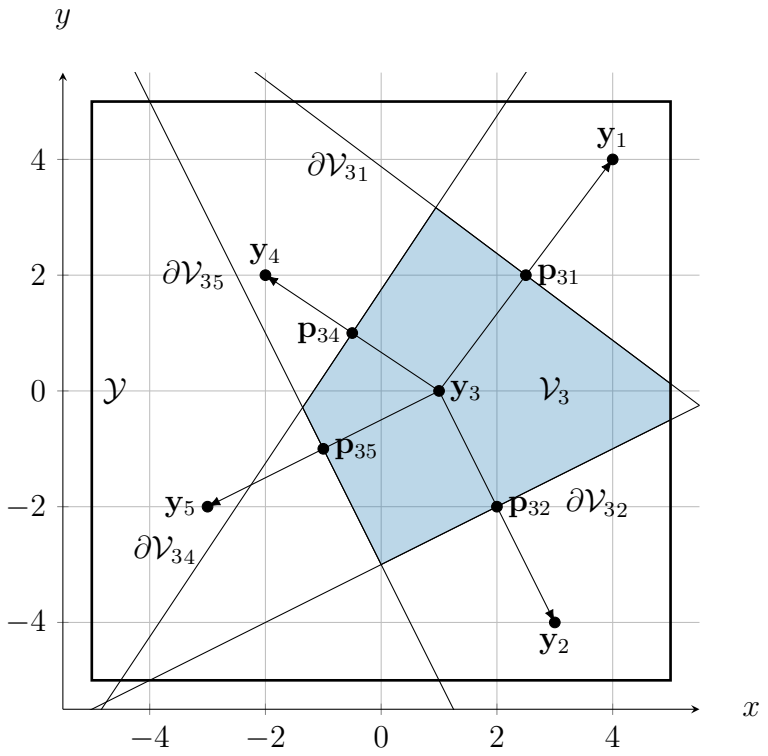


Figure 2.10: Construction of the Voronoi cell $\mathcal{V}_3$.

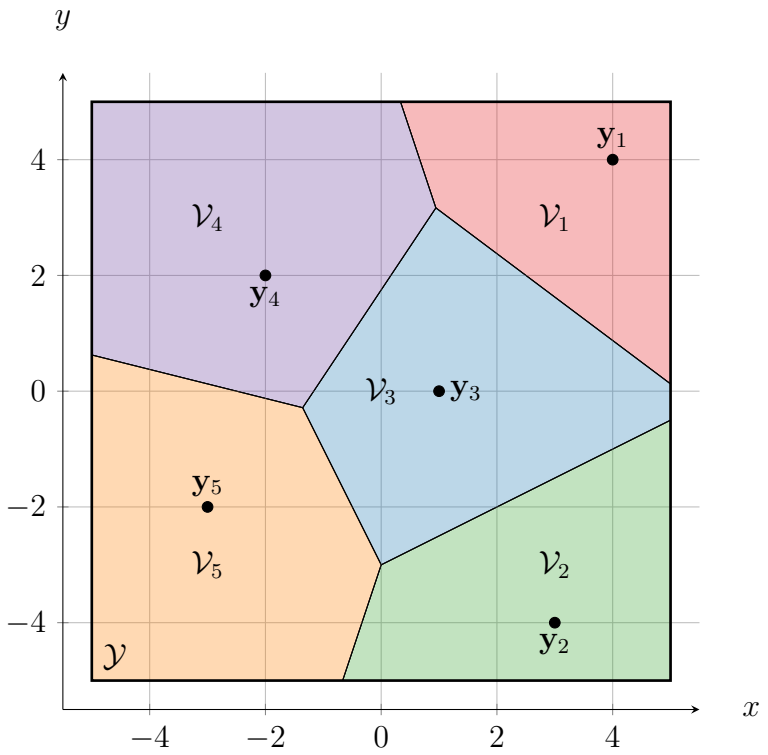


Figure 2.11: An example of Voronoi tessellation in $\mathbb{R}^2$.

2.4.2 Generalized Voronoi tessellations

From the definition of the Voronoi tessellation given in the previous paragraph, generalizations have been proposed such as the ones in Sugihara (1993) or Choset and Burdick (1995). In Choset and Burdick (1995), the generators of the Voronoi tessellation are not defined as points as in Section 2.4.1 but as given subsets of the set being tessellated. Evans and Sember (2008) and Tzes et al. (2018) give results on the shape of a tessellation when the generating subsets are discs (the tessellation is then often called *power Voronoi diagram*). Moreover, Evans and Sember (2008) gives a preliminary result on the complexity of the tessellation when the generating subsets are polytopes. From this, Chevet et al. (2019) introduces the shape of a so-called *box-based guaranteed Voronoi tessellation* when the generating subsets are boxes. The results introduced in Chevet et al. (2019) are recalled and a method to explicitly compute the expression of the borders of the guaranteed cells as well as a linear approximation of these borders is contributed in Paragraph 2.4.2.1. Finally, Paragraph 2.4.2.2 presents a new generalization of the Voronoi tessellation named hereafter *pseudo-Voronoi tessellation*.

2.4.2.1 Box-based guaranteed Voronoi tessellation

A kind of generalization of Voronoi tessellations has been used for some time in control theory (Rowat, 1979, Choset and Burdick, 1995) and has recently been called guaranteed Voronoi (GV) tessellation (Evans and Sember, 2008, Tzes et al., 2018, Chevet et al., 2019) since its use arises from uncertainty on the position of agents due to a perturbation on their measurement signal. The name *guaranteed* thus implies that all the elements in a guaranteed Voronoi cell (GVC) are guaranteed to be closer to all the possible positions of an agent than to all the possible positions of another agent.

Definition 2.27: Guaranteed Voronoi cell

Let $\mathcal{Y}$ be a polytopic subset of $\mathbb{R}^n$. Let $\mathcal{W}_1, \dots, \mathcal{W}_N \subset \mathcal{Y}$, with $N \in \mathbb{N}$, be the *generator sets* of the guaranteed Voronoi (GV) tessellation, assuming that $\mathcal{W}_i \cap \mathcal{W}_j = \emptyset$, $\forall i, j \in \overline{1, N}$, $i \neq j$. Then, the guaranteed[1] Voronoi cell $\mathcal{V}_i^g$ of a generator set $\mathcal{W}_i$ is the set:

$$\mathcal{V}_i^g = \left\{ \mathbf{y} \in \mathcal{Y} \;\middle|\; \|\mathbf{y} - \mathbf{z}_i\|_2 \leq \|\mathbf{y} - \mathbf{z}_j\|_2, \forall \mathbf{z}_i \in \mathcal{W}_i, \mathbf{z}_j \in \mathcal{W}_j, j \in \overline{1, N}, i \neq j \right\}. \quad (2.47)$$

With such a definition, in the general case, it is not guaranteed that the tessellation covers the whole original polytope, i.e.:

$$\bigcup_{i \in \overline{1,N}} \mathcal{V}_i^g \subseteq \mathcal{Y},$$

while it is still guaranteed that for given $i, j \in \overline{1, N}$, $i \neq j$, $\mathcal{V}_i^g \cap \mathcal{V}_j^g \neq \emptyset$.

In the present thesis, the generator sets are considered to be boxes as defined in Definition 2.12 centered on a point $\mathbf{y}_i \in \mathcal{Y}$, with $i \in \overline{1, N}$. Thus, the subsets $\mathcal{W}_1, \dots, \mathcal{W}_N$ of $\mathcal{Y}$ can be defined as:

$$\mathcal{W}_i = \mathbb{B}^n(\boldsymbol{\alpha}_i) \oplus \{\mathbf{y}_i\} \quad (2.48)$$

[1] In the notation $\mathcal{V}_i^g$, the superscript g is used to designate a *guaranteed* Voronoi cell.

where $\boldsymbol{\alpha}_i \in \mathbb{R}^n$ and $\mathbf{y}_i \in \mathcal{Y} \ominus \mathbb{B}^n(\boldsymbol{\alpha}_i)$, such that $\mathcal{W}_i \subset \mathcal{Y}$, with $i \in \overline{1, N}$. It is obvious from Definition 2.27 that a H-representation of the cell is not immediate from (2.47). The first step consists in looking at the border of the guaranteed cell of $\mathcal{W}_i$ induced by $\mathcal{W}_j$:

$$\partial \mathcal{V}_{i\,j}^{g} = \left\{ \mathbf{y} \in \mathcal{Y} \;\middle|\; \max_{\mathbf{z}_i \in \mathcal{W}_i} \|\mathbf{y} - \mathbf{z}_i\|_2 = \min_{\mathbf{z}_j \in \mathcal{W}_j} \|\mathbf{y} - \mathbf{z}_j\|_2 \right\}. \tag{2.49}$$

In the general case, i.e. when the sets $\mathcal{W}_i$ do not have a particular shape, these maximum and minimum would be difficult to compute. However, in the case of boxes as defined in (2.48), they can be explicitly formulated. To find these expressions, it can be useful to look at what happens in two dimensions.

The maximum distance of a point to a box in $\mathbb{R}^2$ is reached for the vertex of the box which is farthest from the point. Figure 2.12 presents an example for a given set $\mathcal{W}_i = \mathbb{B}^2\left(\begin{bmatrix}1 & 2\end{bmatrix}^\top\right) \oplus \{\mathbf{y}_i\}$, with $\mathbf{y}_i = \begin{bmatrix}0 & 1\end{bmatrix}^\top$, and two points $\mathbf{y}_1, \mathbf{y}_2 \in \mathcal{Y}$. Since the objective of this example is to find the expression of the maximum distance from a point to the set $\mathcal{W}_i$, the shape of $\mathcal{Y}$ is irrelevant and $\mathcal{Y} = \mathbb{R}^2$ such that $\mathcal{W}_i \subset \mathcal{Y}$ and $\mathbf{y}_i \in \mathcal{Y} \ominus \mathbb{B}^2\left(\begin{bmatrix}1 & 2\end{bmatrix}^\top\right)$. It can be seen in the figure that $\max_{\mathbf{z}_i \in \mathcal{W}_i} \|\mathbf{y}_1 - \mathbf{z}_i\|_2$, the maximum distance from $\mathbf{y}_1 \in \mathcal{Y}$ to $\mathcal{W}_i$, i.e. the box $\mathbb{B}^2\left(\begin{bmatrix}1 & 2\end{bmatrix}^\top\right)$ centered on $\mathbf{y}_i$, is reached for $\mathbf{z}_i = \mathbf{v}_1$ and $\max_{\mathbf{z}_i \in \mathcal{W}_i} \|\mathbf{y}_2 - \mathbf{z}_i\|_2$ is reached for $\mathbf{z}_i = \mathbf{v}_2$.

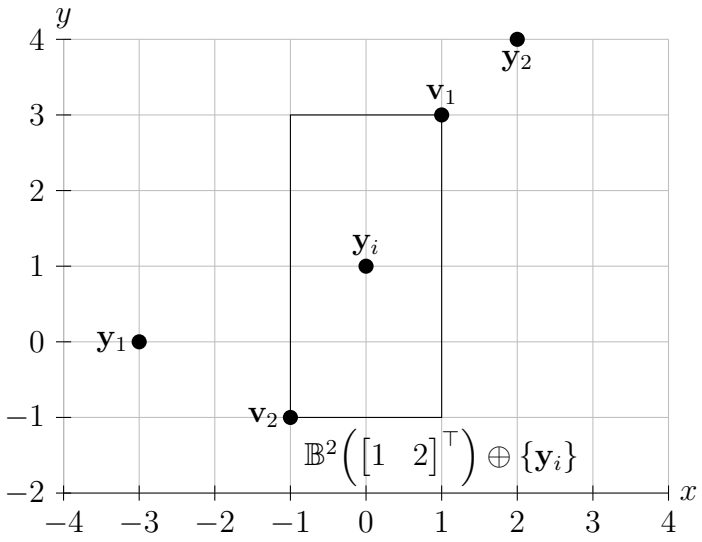


Figure 2.12: Maximum distance of a point to a box in $\mathbb{R}^2$.

From the example of Figure 2.12, an expression for the maximum distance of a point to a box in $\mathbb{R}^2$ can be formulated:

$$\max_{\mathbf{z}_i \in \mathcal{W}_i} \|\mathbf{y} - \mathbf{z}_i\|_2 = \sqrt{\left(|x - x_i| + \alpha_{i,x}\right)^2 + \left(|y - y_i| + \alpha_{i,y}\right)^2}$$

where $\mathcal{W}_i$ is defined as in (2.48), with $n = 2$, $\boldsymbol{\alpha}_i = \begin{bmatrix}\alpha_{i,x} & \alpha_{i,y}\end{bmatrix}^\top$, $\mathbf{y}_i = \begin{bmatrix}x_i & y_i\end{bmatrix}^\top$ and $\mathbf{y} = \begin{bmatrix}x & y\end{bmatrix}^\top$.

This expression can be generalized in $\mathbb{R}^n$, giving:

$$\max_{\mathbf{z}_i \in \mathcal{W}_i} \|\mathbf{y} - \mathbf{z}_i\|_2 = \| |\mathbf{y} - \mathbf{y}_i| + \boldsymbol{\alpha}_i \|_2. \tag{2.50}$$

The expression of the minimum distance of a point to a box is less straightforward. It depends on the relative position of the point to the box and it is reached either on a facet of the box or on a vertex or be zero if the point is inside the box. In the example of Figure 2.13, the set $\mathcal{W}_j$ is $\mathcal{W}_j = \mathbb{B}^2\left(\begin{bmatrix}2 & 1\end{bmatrix}^\top\right) \oplus \{\mathbf{y}_j\}$ with $\mathbf{y}_j = \begin{bmatrix}0 & 1\end{bmatrix}^\top$. Then, $\mathbb{R}^2$ can be divided into nine areas of four types as shown in Figure 2.13. In areas of type 1, the minimum distance from a point to $\mathcal{W}_j$ is the distance to the point that has the same ordinate and is on the closest border of $\mathcal{W}_j$ (e.g. the minimal distance from $\mathbf{y}_1$ to $\mathcal{W}_j$ is the distance from $\mathbf{y}_1$ to $\mathbf{p}_1$). In areas of type 2, the minimum distance is the same as in areas of type 1 while replacing ordinate by abscissa (e.g. the minimal distance from $\mathbf{y}_2$ to $\mathcal{W}_j$ is the distance from $\mathbf{y}_2$ to $\mathbf{p}_2$). In areas of type 3, the minimum distance from a point to $\mathcal{W}_j$ is the distance to the closer vertex of $\mathcal{W}_j$ (e.g. the minimal distance from $\mathbf{y}_3$ to $\mathcal{W}_j$ is the distance from $\mathbf{y}_3$ to $\mathbf{p}_3$). Finally, in areas of type 4, the minimum distance from all points to $\mathcal{W}_j$ is 0 since the points are already in $\mathcal{W}_i$.

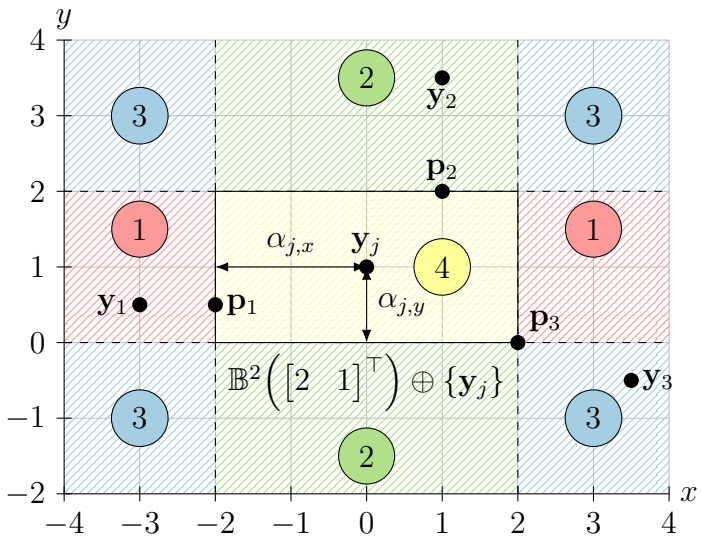


Figure 2.13: Minimum distance of a point to a box in $\mathbb{R}^2$.

From the example of Figure 2.13, an expression for the minimum distance of a point to a box in $\mathbb{R}^2$ can be formulated:

$$\min_{\mathbf{z}_j \in \mathcal{W}_j} \|\mathbf{y} - \mathbf{z}_j\|_2 = \sqrt{(\max\{|x - x_j| - \alpha_{j,x}, 0\})^2 + (\max\{|y - y_j| - \alpha_{j,y}, 0\})^2}$$

where $\mathcal{W}_j$ is defined as in (2.48), with $\boldsymbol{\alpha}_j = \begin{bmatrix}\alpha_{j,x} & \alpha_{j,y}\end{bmatrix}^\top$, $\mathbf{z}_j = \begin{bmatrix}x_j & y_j\end{bmatrix}^\top$ and $\mathbf{y} = \begin{bmatrix}x & y\end{bmatrix}^\top$.

This expression can be generalized in $\mathbb{R}^n$, giving:

$$\min_{\mathbf{z}_j \in \mathcal{W}_j} \|\mathbf{y} - \mathbf{z}_j\|_2 = \sqrt{\sum_{k=1}^{n} (\max\{|y_k - y_{j,k}| - \alpha_{j,k}, 0\})^2} \tag{2.51}$$

where $\mathcal{W}_j = \mathbb{B}^n(\boldsymbol{\alpha}_j) \oplus \{\mathbf{y}_j\}$, with $\boldsymbol{\alpha}_j = \begin{bmatrix}\alpha_{j,1} & \cdots & \alpha_{j,n}\end{bmatrix}^\top \in \mathbb{R}^n$, the center of the box $\mathbf{y}_j = \begin{bmatrix}y_{j,1} & \cdots & y_{j,n}\end{bmatrix}^\top \in \mathcal{Y} \ominus \mathbb{B}^n(\boldsymbol{\alpha}_j) \subset \mathbb{R}^n$ and $\mathbf{y} = \begin{bmatrix}y_1 & \cdots & y_n\end{bmatrix}^\top \in \mathcal{Y} \subset \mathbb{R}^n$.

With the expressions given in (2.50) and (2.51), the shape of $\partial\mathcal{V}^g_{i\,j}$ in $\mathbb{R}^2$ can be deduced and generalized to higher dimensions (which is not treated in the present thesis).

Given (2.50) and (2.51), the equation defining $\partial\mathcal{V}^g_{i\,j}$ is piecewise and continuous. Each piece is defined over an area corresponding to a type of area defined in Figure 2.13. It is easy to verify that the intersection of $\partial\mathcal{V}^g_{i\,j}$ and an area of type 1 or 2 defined for $\mathcal{W}_j$ is a parabolic arc. Indeed, if $\mathbf{y} \in \partial\mathcal{V}^g_{i\,j}$ is in an area of type 1:

$$(|x - x_i| + \alpha_{i,x})^2 + (|y - y_i| + \alpha_{i,y})^2 = (|x - x_j| - \alpha_{j,x})^2$$

or, by developing and gathering the terms depending on x:

$$\begin{aligned}&(|y - y_i| + \alpha_{i,y})^2 \\ &\quad = 2x(x_i - x_j) - 2\alpha_{j,x}|x - x_j| - 2\alpha_{i,x}|x - x_i| + \alpha_{j,x}^2 - \alpha_{i,x}^2 + x_j^2 - x_i^2 \end{aligned} \tag{2.52}$$

which is the equation of a parabola.

If $\mathbf{y}$ is in an area of type 2 for $\mathcal{W}_j$, the role of x and y are inverted in equation (2.52).

The intersection of $\partial\mathcal{V}^g_{i\,j}$ and an area of type 3 for $\mathcal{W}_j$ is a line segment since:

$$(|x - x_i| + \alpha_{i,x})^2 + (|y - y_i| + \alpha_{i,y})^2 = (|x - x_j| - \alpha_{j,x})^2 + (|y - y_j| - \alpha_{j,y})^2$$

or, equivalently:

$$\begin{aligned}&2y(y_j - y_i) + 2\alpha_{i,y}|y - y_i| + 2\alpha_{j,y}|y - y_j| + \alpha_{i,y}^2 - \alpha_{j,y}^2 + y_i^2 - y_j^2 \\ &\qquad = 2x(x_i - x_j) - 2\alpha_{j,x}|x - x_j| - 2\alpha_{i,x}|x - x_i| + \alpha_{j,x}^2 - \alpha_{i,x}^2 + x_j^2 - x_i^2\end{aligned}$$

which is the equation of a line.

Finally, the intersection of $\partial\mathcal{V}^g_{i\,j}$ with an area of type 4 for $\mathcal{W}_j$ is an elliptic arc since:

$$(|x - x_i| + \alpha_{i,x})^2 + (|y - y_i| + \alpha_{i,y})^2 = 0$$

which is the equation of an ellipse.

Example 2.11: Construction of guaranteed borders

Let $\mathcal{W}_i = \mathbb{B}^2\left(\begin{bmatrix}2 & 1\end{bmatrix}^\top\right) \oplus \{\mathbf{y}_i\}$, *where* $\mathbf{y}_i = \begin{bmatrix}-6 & -7\end{bmatrix}^\top$, *and* $\mathcal{W}_j = \mathbb{B}^2\left(\begin{bmatrix}1 & 2\end{bmatrix}^\top\right) \oplus \{\mathbf{y}_j\}$, *where* $\mathbf{y}_j = \begin{bmatrix}5 & 4\end{bmatrix}^\top$ *be two sets of* $\mathbb{R}^2$. *These two sets are drawn in Figure 2.14,* $\mathcal{W}_i$ *in black and* $\mathcal{W}_j$ *in gray, as well as dashed lines delimiting the same areas as in Figure 2.13 for the set* $\mathcal{W}_i$. *For* $\mathcal{W}_i$, *the red hatched area* $\mathcal{A}_1$ *is an area of type 1, the green hatched area* $\mathcal{A}_2$ *is an area of type 2 and the blue hatched area is an area of type 3. To build* $\partial\mathcal{V}^g_{j\,i}$, *it is necessary to find all the* $\mathbf{y} \in \mathbb{R}^2$ *such that:*

$$\min_{\mathbf{z}_i \in \mathcal{W}_i} \|\mathbf{y} - \mathbf{z}_i\|_2 = \max_{\mathbf{z}_j \in \mathcal{W}_j} \|\mathbf{y} - \mathbf{z}_j\|_2.$$

If a point of $\mathcal{A}_1$ *is part of* $\partial\mathcal{V}^g_{j\,i}$, *it has to satisfy:*

$$(|x - 5| + 1)^2 + (|y - 4| + 2)^2 = (|x + 6| - 2)^2$$

which is the equation of of a piecewise continuous parabola. This part of $\partial\mathcal{V}^g_{j\,i}$ *is drawn as a dashdotted blue line in Figure 2.14.*

If a point of $\mathcal{A}_2$ is part of $\partial\mathcal{V}^g_{j\,i}$, it has to satisfy:

$$(|x-5|+1)^2+(|y-4|+2)^2=(|x+7|-1)^2$$

which is again the equation of a piecewise continuous parabola. This part of $\partial\mathcal{V}^g_{j\,i}$ is also drawn in a dashdotted blue line in Figure 2.14.

Finally, if a point of $\mathcal{A}_3$ is part of $\partial\mathcal{V}^g_{j\,i}$, it has to satisfy:

$$(|x-5|+1)^2+(|y-4|+2)^2=(|x+6|-2)^2+(|x+7|-1)^2$$

which is, as discussed before, the equation of a piecewise continuous line. This part of $\partial\mathcal{V}^g_{j\,i}$ is drawn as a plain blue line in Figure 2.14.

In addition, $\partial\mathcal{V}^g_{j\,i}$ also extends to the left and to the right of the part that is presented in Figure 2.14. Indeed, to the left of $\mathcal{A}_2$, the border $\partial\mathcal{V}^g_{j\,i}$ intersects an area of type 3 for $\mathcal{W}_i$ and it is then composed of a piecewise continuous line. The same way, below $\mathcal{A}_1$, the border $\partial\mathcal{V}^g_{j\,i}$ intersects another area of type 3 for $\mathcal{W}_i$ and it is composed of a piecewise continuous line.

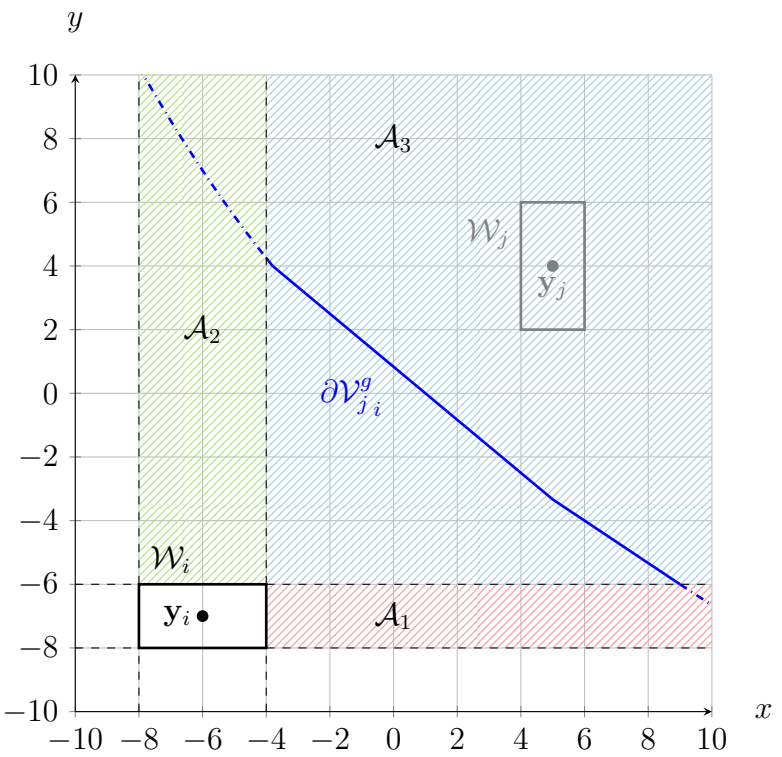


Figure 2.14: Example of guaranteed Voronoi cell border generated by two sets in $\mathbb{R}^2$.

The same way as for $\partial\mathcal{V}^g_{j\,i}$, the border $\partial\mathcal{V}^g_{i\,j}$ is obtained by finding all the $\mathbf{y}\in\mathbb{R}^2$ such that:

$$\max_{\mathbf{z}_i\in\mathcal{W}_i}\|\mathbf{y}-\mathbf{z}_i\|_2=\min_{\mathbf{z}_j\in\mathcal{W}_j}\|\mathbf{y}-\mathbf{z}_j\|_2.$$

Figure 2.15 then presents the resulting borders $\partial\mathcal{V}^g_{i\,j}$ in red and $\partial\mathcal{V}^g_{j\,i}$ in blue for the sets $\mathcal{W}_i$ and $\mathcal{W}_j$.

From the equalities defining $\partial\mathcal{V}^g_{i\,j}$, inequalities defining $\mathcal{V}^g_i$ can be deduced. However, because of the presence of parabolic or elliptic arcs, some of those inequalities are

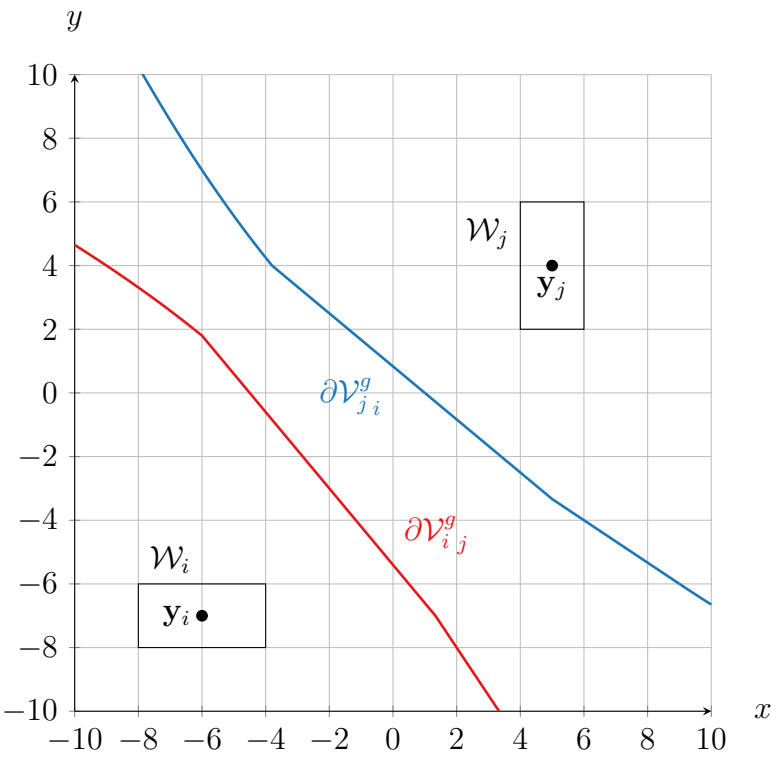


Figure 2.15: Box-based guaranteed Voronoi cell borders generated by two rectangles in $\mathbb{R}^2$.

not linear which could pose a problem for the control algorithms that are developed in the following chapters.

Due to the convexity of these arcs, a linear approximation of the parabolic and elliptic arcs, albeit conservative, is easy to obtain. This approximation consists in replacing the quadratic arcs by a line joining their intersection with the borders of the area in which they are defined. Figure 2.16 gives the linear approximation of the borders $\partial\mathcal{V}^g_{i\,j}$ and $\partial\mathcal{V}^g_{j\,i}$ from Example 2.11 where the original borders are the dashed lines and the approximated borders are the plain light red and light blue lines. In the remainder of this thesis, the linear approximation of the box-based guaranteed Voronoi cells is always used.

From all the developments presented above, a H-representation of the guaranteed Voronoi cells $\mathcal{V}^g_i$, with $i \in \overline{1, N}$, generated by sets $\mathcal{W}_i$ having the form (2.48) can be constructed. The matrices of the H-representation of $\mathcal{V}^g_i$ are then obtained by appending all the linear inequalities induced by the borders $\partial\mathcal{V}^g_{i\,j}$ from (2.49) and the linear inequalities defining the polytopic set $\mathcal{Y}$ being tessellated.

Example 2.12: Box-based guaranteed Voronoi tessellation in $\mathbb{R}^2$

Let $\boldsymbol{H} = \begin{bmatrix} 3 & 0 & -1 & -1 & 2 \\ 2 & 1 & 2 & -1 & -1 \end{bmatrix}^\top$ *and* $\boldsymbol{\theta} = \begin{bmatrix} 24 & 6 & 9 & 15 & 12 \end{bmatrix}^\top$ *inducing a polytope* $\mathcal{Y}$ *in* $\mathbb{R}^2$. *Let:*

$$\mathbf{y}_1 = \begin{bmatrix} 4 \\ 2 \end{bmatrix} \quad \mathbf{y}_2 = \begin{bmatrix} -1 \\ -10 \end{bmatrix} \quad \mathbf{y}_3 = \begin{bmatrix} 0 \\ -4 \end{bmatrix} \quad \mathbf{y}_4 = \begin{bmatrix} -8 \\ -4 \end{bmatrix} \quad \mathbf{y}_5 = \begin{bmatrix} -2 \\ 0 \end{bmatrix}$$

$$\boldsymbol{\alpha}_1 = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix} \quad \boldsymbol{\alpha}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \boldsymbol{\alpha}_3 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \boldsymbol{\alpha}_4 = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} \quad \boldsymbol{\alpha}_5 = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}$$

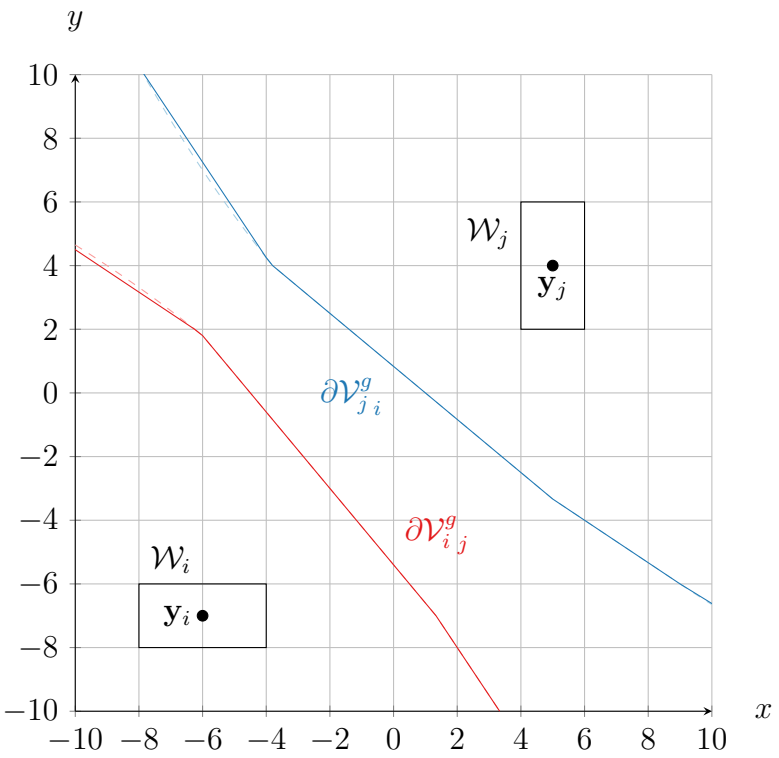


Figure 2.16: Linear approximation of a guaranteed cell border generated by two sets in $\mathbb{R}^2$.

inducing the sets $\mathcal{W}_i = \mathbb{B}^2(\boldsymbol{\alpha}_i) \oplus \{\mathbf{y}_i\}$, *with* $i \in \overline{1,5}$.

The guaranteed Voronoi tessellation of the boxes $\mathcal{W}_i$, *with* $i \in \overline{1,5}$ *is presented in Figure 2.17. It can be seen that* $\mathcal{W}_i$ *is not always totally included inside its guaranteed cell* $\mathcal{V}^g_i$, *e.g.* $\mathcal{W}_3 \not\subset \mathcal{V}^g_3$.

2.4.2.2 Pseudo-Voronoi tessellation

Further generalization of the classical Voronoi tessellation introduced in Section 2.4.1 can be formulated. However, this generalization is presented only in $\mathbb{R}^2$, an extension to higher dimensions being possible but of increased complexity.

Let $\mathcal{Y} \in \mathbb{R}^2$ be a polytope in $\mathbb{R}^2$. Let $\mathbf{y}_1, \ldots, \mathbf{y}_N \in \mathcal{Y}$ be N points of $\mathcal{Y}$ such that $\mathbf{y}_i \neq \mathbf{y}_j$ for all $i, j \in \overline{1,N}$, $i \neq j$. This paragraph introduces a *pseudo-Voronoi tessellation*[2] of $\mathcal{W}$ generated by the points $\mathbf{y}_i$, with $i \in \overline{1,N}$ such that:

$$\mathcal{Y} = \bigcup_{i=1}^{N} \mathcal{V}^p_i, \text{ with } \mathcal{V}^p_i \cap \mathcal{V}^p_j = \emptyset,\ \forall i, j \in \overline{1,N},\ i \neq j. \tag{2.53}$$

In (2.46), the border $\partial\mathcal{V}_{ij}$ of the Voronoi cell generated by the two points $\mathbf{y}_i$ and $\mathbf{y}_j$ has been characterized as the hyperplane passing by $(\mathbf{y}_i + \mathbf{y}_j)/2$ with the normal vector $\mathbf{y}_j - \mathbf{y}_i$. The border $\partial\mathcal{V}^p_{i\,j}$ of the pseudo-Voronoi cell generated by two points $\mathbf{y}_i$ and $\mathbf{y}_j$ is defined as the hyperplane passing by $\alpha_{ij}\mathbf{y}_j + (1-\alpha_{ij})\mathbf{y}_i$, where $\alpha_{ij} \in (0,1)$, with the normal vector $\mathbf{y}_j - \mathbf{y}_i$, i.e.:

$$\partial\mathcal{V}^p_{i\,j} = \left\{ \mathbf{y} \in \mathcal{Y} \;\middle|\; (\mathbf{y}_j - \mathbf{y}_i)^\top \mathbf{y} = (\mathbf{y}_j - \mathbf{y}_i)^\top (\alpha_{ij}\mathbf{y}_j + (1-\alpha_{ij})\mathbf{y}_i) \right\}. \tag{2.54}$$

[2] In the notation $\mathcal{V}^p_i$, the superscript p is used to designate a *pseudo-Voronoi cell*.

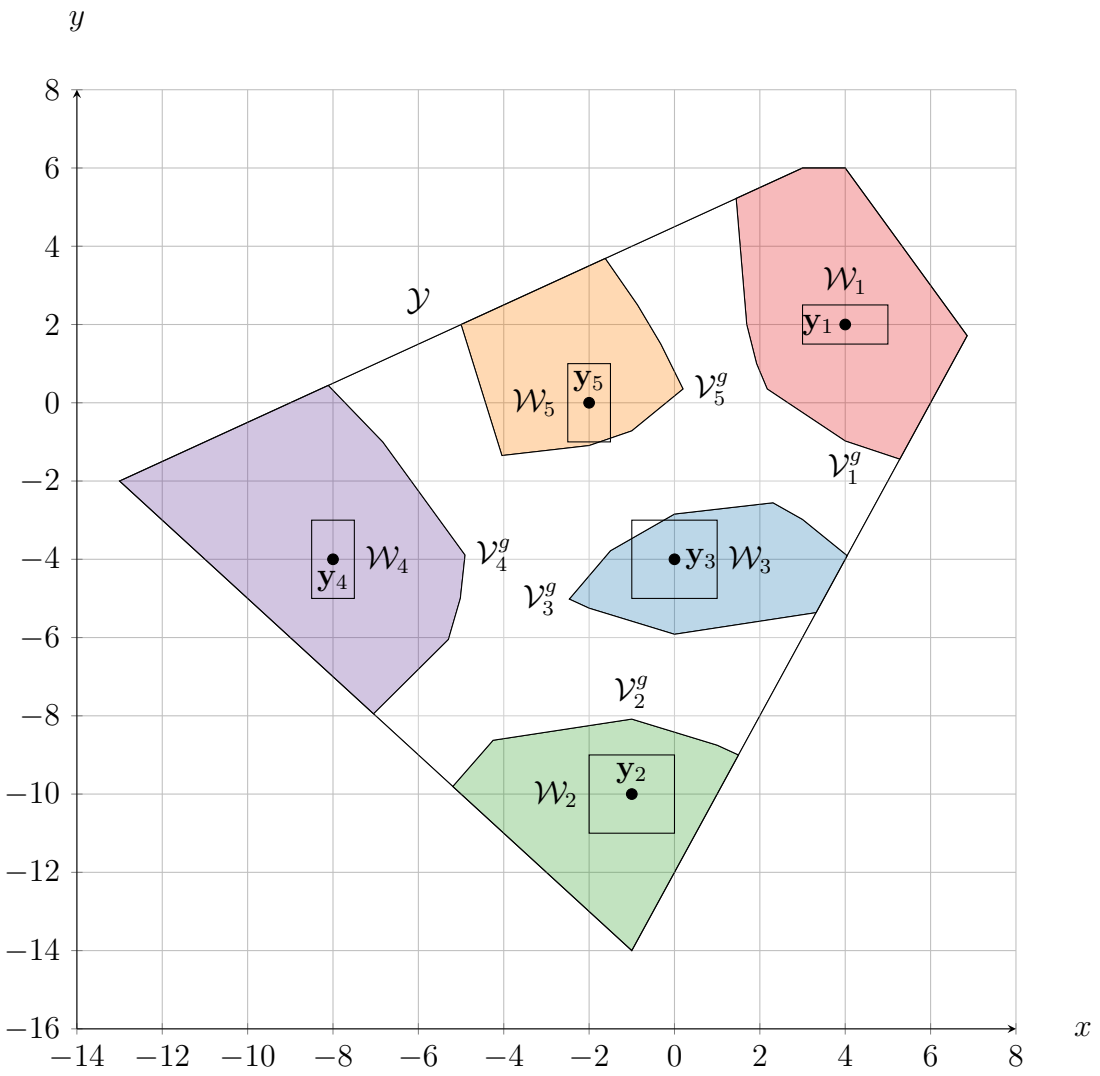


Figure 2.17: Box-based guaranteed Voronoi tessellation of five generator sets in $\mathbb{R}^2$.

In order to have $\mathcal{V}_j^p \cap \mathcal{V}_i^p = \emptyset$, it is necessary that $\alpha_{ij} \leq 1 - \alpha_{ji}$. Moreover, in order to have:

$$\mathcal{Y} = \bigcup_{i=1}^{N} \mathcal{V}_i^p,$$

it is also necessary that $\alpha_{ij} = 1 - \alpha_{ji}$. However, this condition is not enough.

A necessary tool for further comprehension is the Delaunay triangulation, which has been introduced in Delaunay (1934) as the dual graph of the Voronoi tessellation. Several algorithms have been proposed to compute it (Delaunay, 1934, Preparata and Shamos, 1985) but is not investigated here. The Delaunay triangulation associates the generators of a Voronoi tessellation by group of three generators. In this thesis, only a graphical method to understand how to build a Delaunay triangulation of a set of generators $\mathcal{T}(\mathbf{y}_1, \ldots, \mathbf{y}_N)$ is presented, the reader can refer to Preparata and Shamos (1985) for explicit algorithms to build it. Graphically, if a triangle of generators is part of the Delaunay triangulation, the Voronoi cells of each of these generators share one vertex. This vertex is the center of the circumscribed circle to the triangle formed by the three generators (it can be noticed that this fact is the

basis of the algorithm to build the Delaunay triangulation).

Example 2.13: Construction of a Delaunay triangulation
It is now possible to build the Delaunay triangulation $\mathcal{T}(\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \mathbf{y}_4, \mathbf{y}_5)$ *as the dual graph of the Voronoi tessellation presented in Example 2.10. From Figure 2.11, the obvious triangles of generators are* $\mathcal{T}_1 = \{\mathbf{y}_1, \mathbf{y}_3, \mathbf{y}_4\}$, $\mathcal{T}_2 = \{\mathbf{y}_2, \mathbf{y}_3, \mathbf{y}_5\}$, $\mathcal{T}_3 = \{\mathbf{y}_3, \mathbf{y}_4, \mathbf{y}_5\}$. *However, a fourth triangle exists in* $\mathcal{T}(\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \mathbf{y}_4, \mathbf{y}_5)$. *Indeed, if* $\mathcal{Y}$ *was wider, the border* $\partial\mathcal{V}_{12}$ *would appear and intersect* $\partial\mathcal{V}_{13}$ *and* $\partial\mathcal{V}_{23}$. *Then, the Voronoi cells* $\mathcal{V}_1$, $\mathcal{V}_2$ *and* $\mathcal{V}_3$ *would also share a vertex thus* $\mathcal{T}_4 = \{\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3\} \in \mathcal{T}(\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \mathbf{y}_4, \mathbf{y}_5)$. *These triangles are presented in Figure 2.18.*

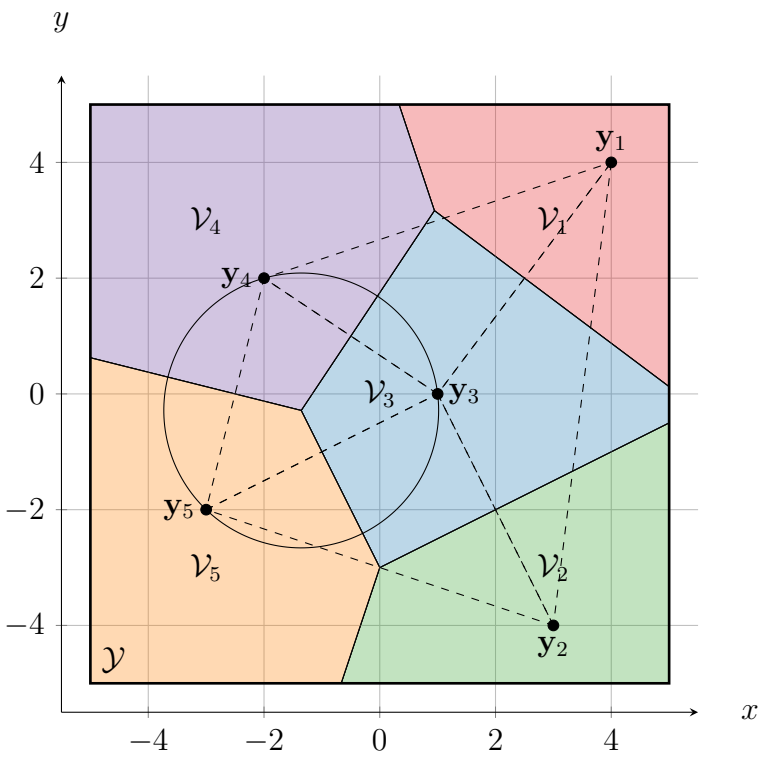


Figure 2.18: An example of Delaunay triangulation in $\mathbb{R}^2$.

A way to ensure that the constraint (2.53) is satisfied with the Delaunay triangulation $\mathcal{T}(\mathbf{y}_1, \ldots, \mathbf{y}_N)$ is to guarantee that for a triangle $\mathcal{T}_m \in \mathcal{T}(\mathbf{y}_1, \ldots, \mathbf{y}_N)$, with $m \in \overline{1, |\mathcal{T}(\mathbf{y}_1, \ldots, \mathbf{y}_N)|}$, such that $\mathcal{T}_m = \{\mathbf{y}_i, \mathbf{y}_j, \mathbf{y}_k\}$, with $i, j, k \in \overline{1, N}$, $i \neq j \neq k$, the borders $\partial\mathcal{V}_{i\,j}^p = \partial\mathcal{V}_{j\,i}^p$, $\partial\mathcal{V}_{i\,k}^p = \partial\mathcal{V}_{k\,i}^p$ and $\partial\mathcal{V}_{j\,k}^p = \partial\mathcal{V}_{k\,j}^p$ are concurrent at a single point as in the classical Voronoi case. If this condition is not respected, there might be parts of $\mathcal{Y}$ not covered by a cell $\mathcal{V}_i^p$.

Consider the vectors $\mathbf{n}_{ij} = \mathbf{y}_j - \mathbf{y}_i$, $\mathbf{n}_{ik} = \mathbf{y}_k - \mathbf{y}_i$, $\mathbf{n}_{jk} = \mathbf{y}_k - \mathbf{y}_j$ in $\mathbb{R}^2$ and the scalars $\theta_{ij} = \mathbf{n}_{ij}^\top(\alpha_{ij}\mathbf{y}_j + (1-\alpha_{ij})\mathbf{y}_i)$, $\theta_{ik} = \mathbf{n}_{ik}^\top(\alpha_{ik}\mathbf{y}_k + (1-\alpha_{ik})\mathbf{y}_i)$ and $\theta_{jk} = \mathbf{n}_{jk}^\top(\alpha_{jk}\mathbf{y}_k + (1-\alpha_{jk})\mathbf{y}_j)$. Then, the borders $\partial\mathcal{V}_{i\,j}^p$, $\partial\mathcal{V}_{i\,k}^p$ and $\partial\mathcal{V}_{j\,k}^p$ are concurrent at a single point if and only if:

$$\det(\mathbf{n}_{ij}, \mathbf{n}_{jk})\theta_{ik} - \det(\mathbf{n}_{ik}, \mathbf{n}_{jk})\theta_{ij} - \det(\mathbf{n}_{ij}, \mathbf{n}_{ik})\theta_{jk} = 0. \tag{2.55}$$

If a tessellation satisfies the conditions that have been presented in this paragraph, it is called a pseudo-Voronoi tessellation because of its similarity with the classical Voronoi tessellation. However, since the basic definition of a Voronoi cell (2.44) is modified, the word “pseudo” is added.

Example 2.14: Construction of a pseudo-Voronoi tessellation in $\mathbb{R}^2$
Let $\mathcal{Y} \subset \mathbb{R}^2$ *be a polytope such that* $\mathcal{Y} = 5\mathbb{B}^2$. *Let* $\mathbf{y}_1 = \begin{bmatrix} 4 & 4 \end{bmatrix}^\top$, $\mathbf{y}_2 = \begin{bmatrix} 3 & -4 \end{bmatrix}^\top$, $\mathbf{y}_3 = \begin{bmatrix} 1 & 0 \end{bmatrix}^\top$, $\mathbf{y}_4 = \begin{bmatrix} -2 & 2 \end{bmatrix}^\top$ *and* $\mathbf{y}_5 = \begin{bmatrix} -3 & -2 \end{bmatrix}^\top$ *be 5 points of* $\mathcal{Y}$. *The Delaunay triangulation of these points* $\mathcal{T}(\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \mathbf{y}_4, \mathbf{y}_5)$ *has been presented in Example 2.13.*

Consider the initial values $\alpha_{12} = 0.3$ *and* $\alpha_{23} = 0.7$ *chosen arbitrarily. Despite the fact that* $\mathcal{V}_1^p$ *and* $\mathcal{V}_2^p$ *do not visibly share a border,* $\mathbf{y}_1$ *and* $\mathbf{y}_2$ *belong to a triangle as shown in Example 2.13, thus the need to chose a value for* α_{12}. *Equation (2.55) results in* $\alpha_{13} = 0.14$. *Let* $\alpha_{14} = 0.4$, *giving* $\alpha_{34} = 23/26$. *Let* $\alpha_{35} = 0.7$, *giving* $\alpha_{45} = 15/34$. *Finally, the previous values lead to* $\alpha_{25} = 0.7$.

Figure 2.19 presents the pseudo-Voronoi tessellation with the values given above. The pseudo-Voronoi cells $\mathcal{V}_i^p$, *with* $i \in \overline{1,5}$, *have the same shape as the cells* $\mathcal{V}_i$ *of the classical Voronoi tessellation in Example 2.10. Indeed, the normal vectors of the borders of* $\mathcal{V}_i^p$ *are the same as the normal vectors of the borders of* $\mathcal{V}_i$. *However, the dimension of the pseudo-Voronoi cells* $\mathcal{V}_i^p$ *and of the Voronoi cells* $\mathcal{V}_i$ *are different.*

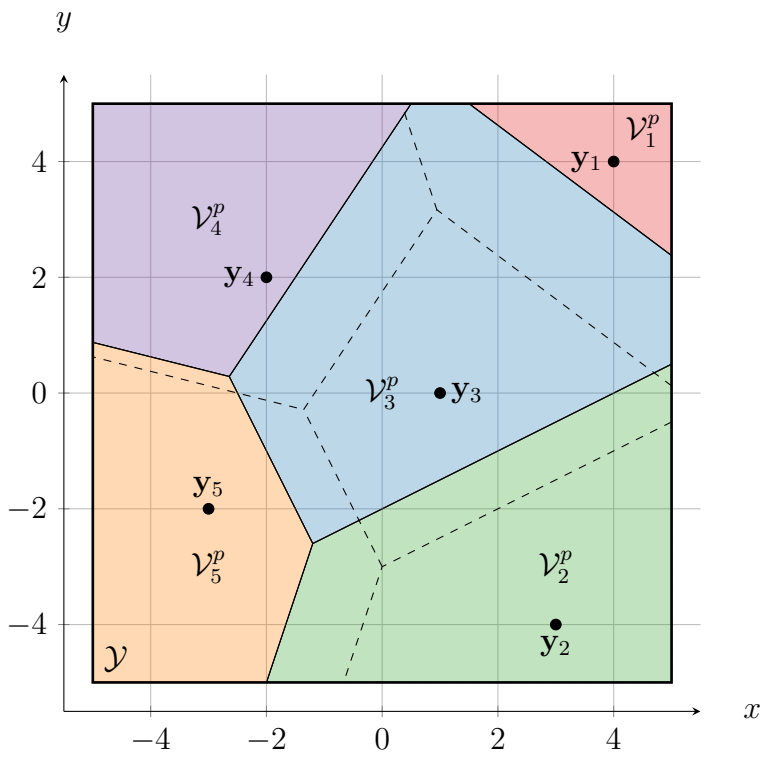


Figure 2.19: An example of pseudo-Voronoi tessellation in $\mathbb{R}^2$.

2.4.3 Chebyshev configuration of a multi-vehicle system

The Voronoi tessellation (either classical or generalized) defined in Sections 2.4.1 and 2.4.2 is used in the following chapter to constrain the movement of a multi-vehicle system as defined in Section 2.3. However, a goal for such a multi-vehicle has also to be introduced, which is the purpose of the present paragraph. To do so, the Chebyshev center (Boyd and Vandenberghe, 2009) of a polytope is presented as well as the definition of a Chebyshev configuration of a multi-vehicle system. The way such objects are used for control purposes is out of the scope of this paragraph and is further introduced in the following chapters.

Before introducing the definition of the Chebyshev center of a polytope, it is necessary to recall how the inclusion of a ball $\mathscr{E}(\boldsymbol{I}_n, \mathbf{c}, r)$ of center $\mathbf{c} \in \mathbb{R}^n$ and radius $r \in \mathbb{R}$ in a half-space $\mathcal{H} = \{\mathbf{x} \in \mathbb{R}^n \mid \boldsymbol{h}\mathbf{x} \leq \theta\}$, with $\boldsymbol{h} \in \mathbb{R}^{1\times n}$ and $\theta \in \mathbb{R}$, is characterized. The ball $\mathscr{E}(\boldsymbol{I}_n, \mathbf{c}, r)$ can be written (Boyd and Vandenberghe, 2009):

$$\mathscr{E}(\boldsymbol{I}_n, \mathbf{c}, r) = \{\mathbf{c} + \mathbf{x} \mid \|\mathbf{x}\|_2 \leq r, \mathbf{x} \in \mathbb{R}^n\}.$$

Then, $\mathscr{E}(\boldsymbol{I}_n, \mathbf{c}, r) \subset \mathcal{H}$ if and only if (Boyd and Vandenberghe, 2009):

$$\max_{\|\mathbf{x}\|_2 \leq r} (\boldsymbol{h}(\mathbf{c} + \mathbf{x})) \leq \theta$$

which is equivalent to:

$$\boldsymbol{h}\mathbf{c} + \max_{\|\mathbf{x}\|_2 \leq r} \boldsymbol{h}\mathbf{x} \leq \theta.$$

The term $\boldsymbol{h}\mathbf{x}$ is the scalar product of the vectors $\boldsymbol{h}^\top$ and $\mathbf{x}$ thus the maximum of $\boldsymbol{h}\mathbf{x}$ is attained when $\boldsymbol{h}^\top$ and $\mathbf{x}$ are colinear and $\|\mathbf{x}\|_2 = r$, i.e.:

$$\max_{\|\mathbf{x}\|_2 \leq r} \boldsymbol{h}\mathbf{x} = r\left\|\boldsymbol{h}^\top\right\|_2.$$

Finally, $\mathscr{E}(\boldsymbol{I}_n, \mathbf{c}, r) \subset \mathcal{H}$ if and only if:

$$\boldsymbol{h}\mathbf{c} + r\left\|\boldsymbol{h}^\top\right\|_2 \leq \theta.$$

A polytope $\mathcal{Y} \subset \mathbb{R}^n$ is the intersection of a finite number of half-spaces such that $\mathcal{Y} = \{\mathbf{x} \in \mathbb{R}^n \mid \boldsymbol{H}_\mathcal{Y}\mathbf{x} \leq \boldsymbol{\theta}_\mathcal{Y}\}$, with $\boldsymbol{H}_\mathcal{Y} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{\theta}_\mathcal{Y} \in \mathbb{R}^m$, where $m \in \mathbb{N}$ is the number of half-spaces composing $\mathcal{Y}$. If the rows of $\boldsymbol{H}_\mathcal{Y}$ and $\boldsymbol{\theta}_\mathcal{Y}$ are denoted by $\boldsymbol{h}_i$ and θ_i, with $i \in \overline{1, m}$, respectively, then, the fact that $\mathscr{E}(\boldsymbol{I}_n, \mathbf{c}, r) \subset \mathcal{Y}$ is characterized by:

$$\boldsymbol{h}_i\mathbf{c} + \left\|\boldsymbol{h}_i^\top\right\|_2 r \leq \theta_i, \ \forall i \in \overline{1, m}.$$

Definition 2.28: Chebyshev center of a polytope

Let $\mathcal{Y} \subset \mathbb{R}^n$ be a polytope in $\mathbb{R}^n$. The Chebyshev center of $\mathcal{Y}$ is the center $\mathbf{c}_\mathcal{Y}$ of the largest ball $\mathscr{E}(\boldsymbol{I}_n, \mathbf{c}_\mathcal{Y}, r_\mathcal{Y})$ and it is computed by solving the optimization problem (Boyd and Vandenberghe, 2009):

$$\begin{aligned} &\underset{\mathbf{c}_\mathcal{Y},\, r_\mathcal{Y}}{\text{maximize}} && r_\mathcal{Y} \\ &\text{subject to} && \\ &r_\mathcal{Y} \geq 0, && \\ &\theta_i \geq \boldsymbol{h}_i\mathbf{c}_\mathcal{Y} + \left\|\boldsymbol{h}_i^\top\right\|_2 r_\mathcal{Y}, && \forall i \in \overline{1, m} \end{aligned} \tag{2.56}$$

where $\boldsymbol{h}_i$ and θ_i, with $i \in \overline{1, m}$, are respectively the rows of $\boldsymbol{H}_\mathcal{Y} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{\theta}_\mathcal{Y} \in \mathbb{R}^m$ the matrices inducing $\mathcal{Y}$.

Example 2.15: Chebyshev center of a polytope in $\mathbb{R}^2$

Let $\boldsymbol{H}$ and $\boldsymbol{\theta}$ be the same matrices as in Example 2.4. Then, considering $\boldsymbol{H} = \boldsymbol{H}_\mathcal{Y}$ and $\boldsymbol{\theta} = \boldsymbol{\theta}_\mathcal{Y}$ induces a polytope $\mathcal{Y}$ in $\mathbb{R}^2$. The MPT3.0 toolbox (Herceg et al., 2013) is used to solve the optimization problem (2.56) for the considered polytopic set $\mathcal{Y}$. Then, the Chebyshev center of the set $\mathcal{Y}$ is $\mathbf{c}_\mathcal{Y} = \begin{bmatrix}-0.8918 & -1.2251\end{bmatrix}^\top$ with a radius $r_\mathcal{Y} = 2.0386$.

Figure 2.20 presents the polytope $\mathcal{Y}$, its Chebyshev center $\mathbf{c}_\mathcal{Y}$ and its Chebyshev ball $\mathscr{E}(\boldsymbol{I}_2, \mathbf{c}_\mathcal{Y}, r_\mathcal{Y})$.

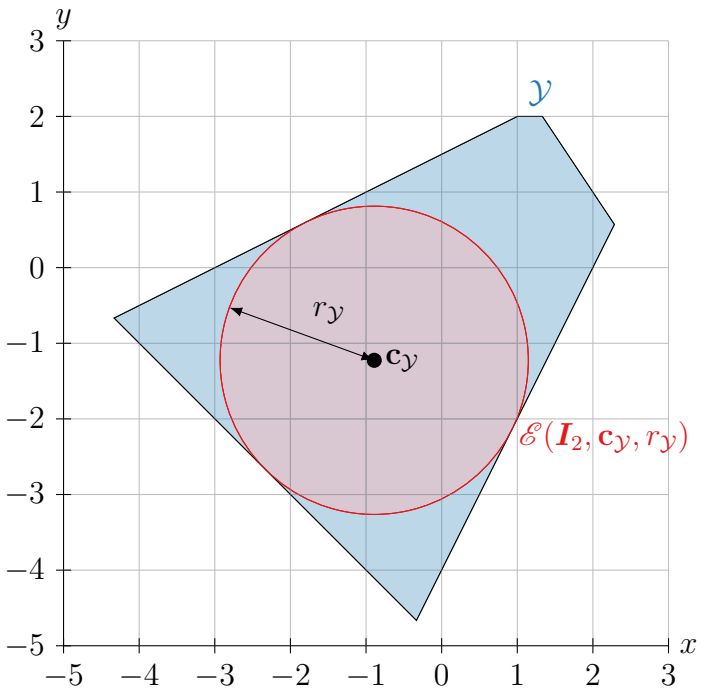


Figure 2.20: Chebyshev center and Chebyshev ball of a polytope in $\mathbb{R}^2$.

With the elements introduced in Section 2.4.1 and Definition 2.28, it is now possible to introduce the Chebyshev configuration of a multi-vehicle system.

Definition 2.29: Chebyshev configuration (Nguyen, 2016)

A Chebyshev configuration (CC) of a multi-vehicle system Σ (2.41a)-(2.41b) is the configuration where the output $\mathbf{y}_i$, with $i \in \overline{1, |\Sigma|}$, of each vehicle coincides with the Chebyshev center of its Voronoi cell $\mathcal{V}_i$, i.e. $\mathbf{y}_i = \mathbf{c}_{\mathcal{V}_i}$ for all $i \in \overline{1, |\Sigma|}$.

Property 2.11: Non-uniqueness of the Chebyshev configuration

A Chebyshev configuration of a multi-vehicle system Σ is not unique and depends on the initial position of the vehicles.

Example 2.16: Chebyshev configuration of a multi-vehicle system in $\mathbb{R}^2$
Let Σ be a multi-vehicle system composed of 5 agents. These agents evolve inside a polytope $\mathcal{Y} = 5\mathbb{B}^2 \subset \mathbb{R}^2$. Figure 2.21 presents a Chebyshev configuration for Σ. In this configuration, each vehicle output coincides with the Chebyshev center of its Voronoi cell.

It can be noticed that while Definition 2.29 has been given for a classical Voronoi tessellation as defined in Section 2.4.1, using a guaranteed or a pseudo-Voronoi tessellation as defined in Section 2.4.2 does not change the definition. Such configurations defined on different Voronoi tessellation are used in the remainder of this thesis.

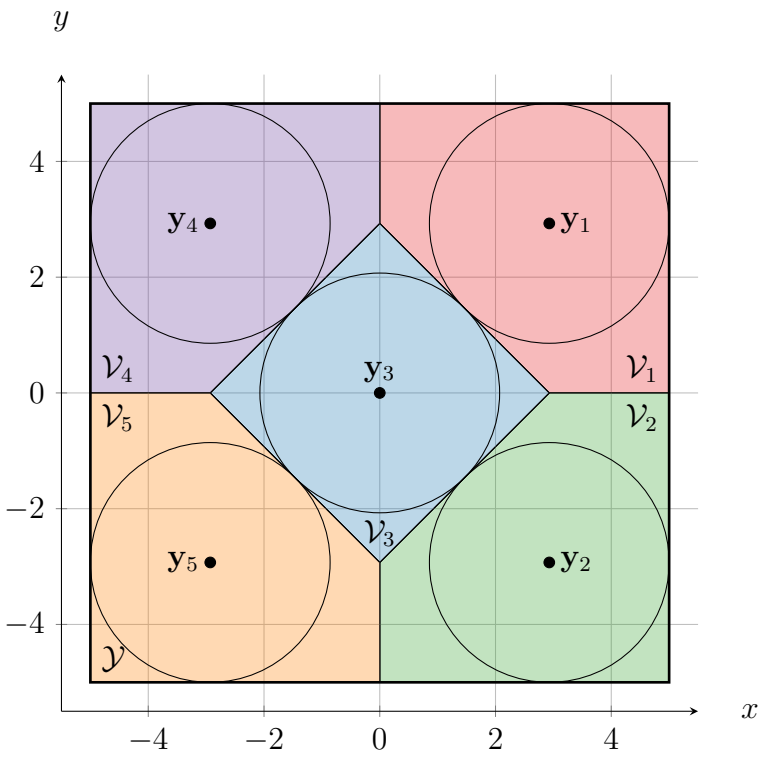


Figure 2.21: An example of Chebyshev configuration for 5 agents in a classical Voronoi tessellation in $\mathbb{R}^2$.

2.5 Continuous random variables and stochastic processes

In numerous real life applications, dynamical systems are subject to stochastic perturbations as presented in Chapter 4. Such perturbations are often modeled as Gaussian white noises which are stochastic processes. Thus, in order to develop control algorithms for such systems, some results on multivariate continuous random variables and stochastic processes have to be introduced. However, these results are limited to what is useful for the development of the algorithm of Section 4.2. Indeed, these results are based on measure theory (Halmos, 1950) and real analysis (DiBenedetto, 2016) which are out of the scope of this thesis. Complements on probability theory and stochastic processes can be found for example to Loève (1977, 1978) and Rosenblatt (1974).

A *continuous random variable*[3] X is an application that maps an element of a sample space Ω to an element of $\mathbb{R}$. The same way, a *multivariate continuous random variable*[4] $\mathbf{X}$ is an application that maps an element of a sample space Ω to an element of $\mathbb{R}^n$, with $n \in \mathbb{N}$. The definition of the sample space Ω is out of the scope of this thesis and $X \in \mathbb{R}$ (respectively $\mathbf{X} \in \mathbb{R}^n$) is used abusively to denote $X(\omega) \in \mathbb{R}$ (respectively $\mathbf{X}(\omega) \in \mathbb{R}^n$), the image of an element $\omega \in \Omega$ by X (respectively $\mathbf{X}$).

Then, specifying the *probability law* $\mathbb{P}$ of a random variable $X \in \mathbb{R}$ (respectively $\mathbf{X} \in \mathbb{R}^n$) consists in finding the probability that $X \in (-\infty, a]$, with $a \in \mathbb{R}$

[3] A *continuous random variable* is also called a *univariate random variable*.

[4] A multivariate random variable $\mathbf{X}$ is a vector of $\mathbb{R}^n$ where each component $X_1, \ldots, X_n \in \mathbb{R}$ of $\mathbf{X} = \begin{bmatrix} X_1 & \cdots & X_n \end{bmatrix}^\top$ is a univariate continuous random variable.

(respectively $\mathbf{X} \in \mathcal{P} \subset \mathbb{R}^n$), denoted by $\mathbb{P}(X \leq a)$ (respectively $\mathbb{P}(\mathbf{X} \in \mathcal{P})$).

The following definition of a *probability density function* is given in the multivariate case but can be obtained in the univariate case by taking $n = 1$ and $\mathcal{P} = (-\infty, a]$, with $a \in \mathbb{R}$.

Definition 2.30: Probability density function

Let $\mathbf{X} \in \mathbb{R}^n$ be a multivariate continuous random variable. The *probability density function* $f_{\mathbf{X}}$ of $\mathbf{X}$ is the non-negative function such that:

$$\mathbb{P}(\mathbf{X} \in \mathcal{P}) = \int_{\mathcal{P}} f_{\mathbf{X}}(\mathbf{x}) \mathrm{d}\mathbf{x} \ \ \forall \mathcal{P} \subset \mathbb{R}^n \text{ and } \int_{\mathbb{R}^n} f_{\mathbf{X}}(\mathbf{x}) \mathrm{d}\mathbf{x} = 1.$$

From the existence of a probability density function, it is possible to define the moments of a random variable. Only two moments are used and presented in this thesis, namely the moment of order 1, or *mathematical expectation*, and the centered moment of order 2, or *variance*.

Definition 2.31: Mathematical expectation

Let $X \in \mathbb{R}$ be a continuous random variable admitting a probability density function f_X. The *mathematical expectation* of X is defined as:

$$\mathbb{E}(X) = \int_{-\infty}^{+\infty} x f_X(x) \mathrm{d}x.$$

Definition 2.32: Expectation of a multivariate random variable

Let $\mathbf{X} = \begin{bmatrix} X_1 & \cdots & X_n \end{bmatrix}^\top \in \mathbb{R}^n$ be a multivariate continuous random variable, where $X_1, \ldots, X_n \in \mathbb{R}$ are univariate random variables. The *mathematical expectation* of $\mathbf{X}$ is the vector composed of the mathematical expectations of its components, i.e.:

$$\mathbb{E}(\mathbf{X}) = \begin{bmatrix} \mathbb{E}(X_1) & \cdots & \mathbb{E}(X_n) \end{bmatrix}^\top.$$

Remark 2.12: Alternative notation for the expectation
The mathematical expectation of a random variable $X \in \mathbb{R}$ or $\mathbf{X} \in \mathbb{R}^n$ is also called the *mean* of X or $\mathbf{X}$. In Chapter 4, the mathematical expectation of a multivariate random variable $\mathbf{X} \in \mathbb{R}^n$ is then denoted by:

$$\mathbb{E}(\mathbf{X}) = \boldsymbol{\mu}_{\mathbf{X}}. \qquad \Diamond$$

The definition of the variance is given only in the multivariate case since only the variance of multivariate continuous random variables is used in this thesis. In this case, the variance of a random variable is often referred as the *variance matrix* of this random variable.

Definition 2.33: Variance matrix

Let $\mathbf{X} \in \mathbb{R}^n$ be a multivariate continuous random variable. The *variance matrix* of $\mathbf{X}$ is defined as:

$$\boldsymbol{\Sigma}_{\mathbf{X}} = \mathbb{E}\Big((\mathbf{X} - \mathbb{E}(\mathbf{X}))(\mathbf{X} - \mathbb{E}(\mathbf{X}))^\top \Big).$$

Definition 2.34: Covariance of two random variables

Let $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^n$ be two multivariate continuous random variables. The *covariance matrix* of $\mathbf{X}$ and $\mathbf{Y}$ is defined as:

$$\text{cov}(\mathbf{X}, \mathbf{Y}) = \text{cov}(\mathbf{Y}, \mathbf{X})^\top = \mathbb{E}\Big((\mathbf{X} - \mathbb{E}(\mathbf{X}))(\mathbf{Y} - \mathbb{E}(\mathbf{Y}))^\top\Big).$$

Before introducing the probability distributions that are used in this thesis, it is necessary to present the concept of independence of two random variables.

Definition 2.35: Independence of two random variables (Flury, 1997)

Two multivariate continuous random variables $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^n$ are called *independent* if:

$$\mathbb{P}((\mathbf{X}, \mathbf{Y}) \in \mathcal{P} \times \mathcal{Q}) = \mathbb{P}(\mathbf{X} \in \mathcal{P})\mathbb{P}(\mathbf{Y} \in \mathcal{Q})$$

where $\mathcal{P}, \mathcal{Q} \subset \mathbb{R}^n$.

The following property, useful for Chapter 4, is then given without proof.

Property 2.12: Covariance of independent variables

Let $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^n$ be two independent multivariate continuous random variables. Then, the covariance of $\mathbf{X}$ and $\mathbf{Y}$ is the null matrix:

$$\text{cov}(\mathbf{X}, \mathbf{Y}) = \mathbf{0}_n.$$

Now that general definitions and properties have been given for continuous random variables, it is necessary to introduce the multivariate normal distribution, which is one of the probability distributions used in control theory.

Definition 2.36: Multivariate normal distribution

Let $\mathbf{X} \in \mathbb{R}^n$ be a multivariate continuous random variable. Then, $\mathbf{X}$ has a *multivariate normal distribution* with mean $\boldsymbol{\mu}_\mathbf{X}$ and variance matrix $\boldsymbol{\Sigma}_\mathbf{X} \succ 0$ if $\mathbf{X}$ admits the probability density function:

$$f_\mathbf{X}(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n \det(\boldsymbol{\Sigma}_\mathbf{X})}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_\mathbf{X})^\top \boldsymbol{\Sigma}_\mathbf{X}^{-1}(\mathbf{x} - \boldsymbol{\mu}_\mathbf{X})\right).$$

Remark 2.13: Normally distributed variable
A multivariate continuous random variable $\mathbf{X} \in \mathbb{R}^n$ having a multivariate normal distribution is called *normally distributed*. ◊

The chi-squared with n degrees of freedom, denoted by χ^2_n is another probability distribution that often appears in various scientific fields, mainly when statistical tests are involved. A univariate continuous random variable having a chi-squared distribution with n degrees of freedom is obtained as the sum of n squared normally distributed univariate random variables. Then, Flury (1997) builds a univariate random variable having a chi-squared distribution with n degrees of freedom from a normally distributed multivariate continuous random variable $\mathbf{X} \in \mathbb{R}^n$.

Property 2.13: Chi-squared distribution (Flury, 1997)

Let $\mathbf{X} \in \mathbb{R}^n$ be a normally distributed multivariate continuous random variable with mean $\boldsymbol{\mu}_{\mathbf{X}}$ and variance matrix $\boldsymbol{\Sigma}_{\mathbf{X}} \succ 0$. Then, the univariate continuous random variable:

$$Y = (\mathbf{X} - \boldsymbol{\mu}_{\mathbf{X}})^\top \boldsymbol{\Sigma}_{\mathbf{X}}^{-1} (\mathbf{X} - \boldsymbol{\mu}_{\mathbf{X}})$$

has a chi-squared distribution with n degrees of freedom.

The probability density function of a random variable having a chi-squared distribution is not presented in this thesis. However, if $X \in \mathbb{R}$ is a univariate continuous random variable with a chi-squared distribution with n degrees of freedom, tables exist in the literature (Elderton, 1902) to obtain the solution α of the equation:

$$\mathbb{P}(X \leq \alpha) = P$$

where $P \in (0, 1)$.

Finally, the notion of normally distributed white noise is defined. Indeed, this kind of noise appears in numerous real world applications as a perturbation on dynamical systems. A white noise is the realization of a stochastic process. For this thesis, even though it is an abuse of notation, a stochastic process is considered to be a continuous random variable which depends on time. Then, a continuous-time stochastic process is denoted by $\mathbf{X}(t) \in \mathbb{R}^n$ and a discrete-time stochastic process is denoted by $\mathbf{X}(k) \in \mathbb{R}^n$.

Definition 2.37: Normally distributed white noise

The continuous-time multivariate continuous random variable $\mathbf{X}(t) \in \mathbb{R}^n$ is a normally distributed white noise if:

- $\mathbf{X}(t)$ is normally distributed with mean $\boldsymbol{\mu}_{\mathbf{X}}(t) = \mathbf{0}_n$ and variance matrix $\boldsymbol{\Sigma}_{\mathbf{X}}(t) \succ 0$ for all $t \in \mathbb{R}_+^*$;
- $\mathbf{X}(t_1)$ and $\mathbf{X}(t_2)$ are independent for all $t_1, t_2 \in \mathbb{R}_+^*$ such that $t_1 \neq t_2$.

The above definition can be obtained in the discrete-time case by replacing $t \in \mathbb{R}_+^*$ by $k \in \mathbb{N}^*$.

2.6 Conclusion

This chapter introduces the main mathematical tools necessary for the comprehension of the results proposed in the remainder of this thesis. Thus, basic operations on ellipsoids and polytopic sets are presented, together with the conventional and generalized (box-based guaranteed and pseudo-) Voronoi tessellation of a polytopic space in $\mathbb{R}^2$. Elements on the Chebyshev configuration for a classical tessellation are also provided to the aim of using them in the context of multi-agent vehicles. Finally, in order to develop control algorithms for systems subject to stochastic perturbations, basic notions on continuous random variables and stochastic processes are recalled.

Chapter 3

Decentralized control for the deployment of a multi-vehicle system

Table of Contents

3.1 Overview of the existing deployment algorithms in the nominal case

A fundamental problem in control of autonomous multi-vehicle systems (MVS) is the deployment over a given environment to carry out a certain mission. Autonomous vehicles missions cover a wide spectrum of real world applications such as forest fire monitoring (Merino et al., 2012, Yuan et al., 2019), ground and resource monitoring (Laliberte and Rango, 2009, Jin and Tang, 2010, d'Oleire Oltmanns et al., 2012), mapping and modeling (Nex and Remondino, 2014, Han and Chen, 2014, Torres et al., 2016) or even surveillance missions (Li et al., 2019, Trujillo et al., 2019). Several of these applications have been tested with one unmanned ground or aerial vehicle (UGV or UAV) or multi-agent systems composed of either UGV, UAV or both. For all these tasks, the word deployment is used in a broader sense than in Schwager et al. (2011) to denote the fact that vehicles move over a given environment to complete their mission.

For the results presented in this thesis, the primary sense of deployment given by Schwager et al. (2011), which considers the deployment of a multi-vehicle system as a strategically appropriate spreading of the MVS in a given environment in order to reach a fixed configuration, is used. The environment and the fixed configuration remain to be characterized. A classical way to approach this problem is to consider the deployment of the multi-vehicle system inside a convex bounded area. This area is considered as the environment mentioned in the definition of Schwager et al. (2011). Several approaches exist to deal with the deployment problem. Choset (2001) gives an overview of methods used when the problem is seen as a motion planning problem. Other techniques have been subsequently proposed such as potential field (Howard et al., 2002), probabilistic (Li and Cassandras, 2005) or Voronoi-based (Cortés et al., 2004). When the deployment of a MAS is meant to ensure a maximal coverage of an environment according to a given criterion, Schwager et al. (2011) show that the last three approaches can be unified through the use of a mixing function encoding the associated criteria.

In the following, methods based on the tessellation of the environment are considered. It is then partitioned into regions depending on the position of the vehicles. An easy way to get such regions is to consider the Voronoi tessellation of the environment where the position of each vehicle in the environment is a generator point. Since the vehicles can move inside the environment, the resulting Voronoi cells are time-varying. The overall problem is then known as a *Voronoi-based deployment* of a multi-vehicle system. The deployment is then handled by driving the vehicles towards a given point defined over their Voronoi cells.

One of the most widely used types of Voronoi-based deployment strategies is the centroidal Voronoi configuration. The vehicles are then driven towards the *center of mass* $\mathbf{c}_{\mathcal{V}_i}^M$ of their Voronoi cell $\mathcal{V}_i$ computed as:

$$\mathbf{c}_{\mathcal{V}_i}^M = \frac{\int_{\mathcal{V}_i} \phi(\mathbf{y})\mathbf{y}\mathrm{d}\mathbf{y}}{\int_{\mathcal{V}_i} \phi(\mathbf{y})\mathrm{d}\mathbf{y}}$$

where i is an identifier of the vehicle and ϕ is a mass density function defined over the environment. The center of mass is computed each time the Voronoi tessellation changes, i.e. at each time sample where the position of the vehicles is observed. Such a strategy has been applied over the years to different types of multi-agent systems (MAS): mobile sensor networks (Cortés et al., 2004), multi-robot systems (Schwager et al., 2011) or autonomous vehicle systems (Sharifi et al., 2014, Moarref and Rodrigues, 2014). The simplest way to reach the centroidal Voronoi configuration is to steer each agent individually towards the center of mass of its associated Voronoi cell, this center of mass being regularly updated given the configuration of the Voronoi tessellation. The MAS then eventually reaches a static configuration where each agent lies on its center of mass. This decentralized control algorithm is known in the literature as *Lloyd's method* (Lloyd, 1982).

However, depending on the complexity of the mass density function, the center of mass defined above can end up being difficult to compute. Nguyen (2016) then proposes a simpler approach based on the *Chebyshev center* (Boyd and Vandenberghe, 2009) of the Voronoi cells of the agents. Such a center is the solution of an optimization problem which is reduced to the linear optimization problem (2.56) due to the

polytopic nature of the Voronoi cell. The Chebyshev center is then often simpler to obtain than the center of mass. Therefore, in this thesis, the objective point for each agent is the Chebyshev center of its Voronoi cell.

Several control methods can be applied to the Voronoi-based deployment problem. Cortés et al. (2004) propose a decentralized state feedback approach for both continuous and discrete-time single integrator dynamics, as well as Hatleskog (2018) and Nguyen et al. (2017) who consider a decentralized discrete-time state feedback, at least as a first approach. Schwager et al. (2011) propose distributed continuous-time consensus-like (Ren and Beard, 2008) controller. In Moarref and Rodrigues (2014) or Nguyen and Stoica Maniu (2016), a decentralized optimal control approach is considered to drive the agents towards a centroidal Voronoi configuration. Finally, Sharifi et al. (2014) consider distributed continuous-time feedback linearization for the MAS to be deployed in a generalized Voronoi tessellation. For purposes other than Voronoi-based deployment, e.g. for formation control, other types of control law can be used such as consensus-based control (Olfati-Saber and Murray, 2004, Ren and Beard, 2008) or sliding mode control (Galzi and Shtessel, 2006, Hu et al., 2015, Li et al., 2017a).

Despite the results obtained with all the control strategies mentioned above, systems are often subject to constraints, either due to limitations in their sensors or actuators or to limitations imposed by the user or the specific application. The control strategies presented in the previous paragraph cannot take such constraints into consideration and assume that the values of the control input remain within acceptable physical limits or impose saturations on the inputs to force the system to remain in a feasible area. One of the control methods that allows to take such constraints into account is model predictive control (MPC). Such a control method has been introduced in Nguyen and Stoica Maniu (2016) and Nguyen (2016) to drive a MAS towards a Chebyshev configuration (i.e. a configuration in which all the agents output coincides with the Chebyshev center of its Voronoi cell as defined in Definition 2.29).

This chapter is then focused on the study of MPC to drive a multi-vehicle system towards a Chebyshev configuration in a convex bounded region. First of all, Section 3.2 formally presents the problem of the deployment of a multi-vehicle system in a polytopic region of $\mathbb{R}^2$. Two model predictive control strategies are then proposed for this deployment, a centralized approach in Section 3.2.1 and a decentralized approach in Section 3.2.2, in the continuity of the work of Nguyen (2016). The convergence and the performance of the proposed decentralized MPC algorithm is analyzed for a MVS where all vehicles have single integrator dynamics. In a second stage, Section 3.3 introduces the dynamics of a quadrotor unmanned aerial vehicle (UAV) and applies the decentralized control algorithm of Section 3.2 to a MVS composed of quadrotor UAVs. Finally, Section 3.4 discusses the convergence and stability of the control strategy applied to a fleet of quadrotor UAVs. In the remainder of this thesis, the multi-vehicle system is called indifferently multi-vehicle system (MVS) or multi-agent system (MAS). In the latter case, it is implied that all agents in the system are vehicles.

3.2 Problem formulation and first results

This section is organized as follows. Section 3.2.1 formulates the control strategy for the Voronoi-based deployment problem of a multi-vehicle system as a centralized MPC optimization problem. The assumptions allowing the control algorithm to be decentralized over all vehicles in the MVS are presented in Section 3.2.2 and a discussion on the convergence of this algorithm is drawn in Section 3.2.3. Finally, simulation results on single integrator dynamics systems are given in Section 3.2.4.

3.2.1 Centralized MPC approach

Let Σ be a multi-agent system composed on N agents. Each agent obeys the dynamics (2.40). Since the MAS is homogeneous according to Assumption 2.6, all the agents have the same dynamics and the dependency on the agent identifier $i \in \overline{1, N}$ can be dropped in the matrices such that:

$$\mathbf{x}_i(k+1) = \boldsymbol{A}\mathbf{x}_i(k) + \boldsymbol{B}\mathbf{u}_i(k) \tag{3.1a}$$
$$\mathbf{y}_i(k) = \boldsymbol{C}\mathbf{x}_i(k) \tag{3.1b}$$

where $\mathbf{x}_i \in \mathcal{X}_i \subset \mathbb{R}^n$, $\mathbf{u}_i \in \mathcal{U}_i \subset \mathbb{R}^m$, $\mathbf{y}_i \in \mathcal{Y}_i \subset \mathbb{R}^2$, $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{B} \in \mathbb{R}^{n\times m}$ and $\boldsymbol{C} \in \mathbb{R}^{2\times n}$, with $i \in \overline{1, N}$. From Assumption 2.3, all the agents share the same output space, then it is possible to also drop the dependency on the agent identifier such that $\mathcal{Y}_i = \mathcal{Y}$ for all $i \in \overline{1, N}$. For the sake of simplicity, it can be assumed that, in the nominal case, the state spaces and the input spaces satisfy $\mathcal{X}_i = \mathcal{X}$ and $\mathcal{U}_i = \mathcal{U}$ for all $i \in \overline{1, N}$. The output $\mathbf{y}_i$ of agent $i \in \overline{1, N}$ is the Cartesian position of agent i in the plane $\mathbb{R}^2$.

Assumption 3.1: Workspace

The output space $\mathcal{Y}$ of the MAS is a convex bounded polytope of $\mathbb{R}^2$. This polytope is also called the workspace *of the MAS.*

The objective of the control algorithm that is presented in this chapter is to drive the agents of the MAS towards a static configuration belonging to $\mathcal{Y}$ by controlling their positions in a two-dimensional space. For the configuration to be static, the points of $\mathcal{Y}$ that the agents reach have to be equilibrium points.

Assumption 3.2: Equilibrium points

The system dynamics (3.1) are such that for all $\mathbf{y}_0 \in \mathcal{Y}$, there exist a state $\mathbf{x}_0 \in \mathcal{X}$ and an input $\mathbf{u}_0 \in \mathcal{U}$ such that:

$$\begin{aligned} \mathbf{x}_0 &= \boldsymbol{A}\mathbf{x}_0 + \boldsymbol{B}\mathbf{u}_0 \\ \mathbf{y}_0 &= \boldsymbol{C}\mathbf{x}_0. \end{aligned} \tag{3.2}$$

Remark 3.1
Assumption 3.2 is satisfied if the matrix:

$$\begin{bmatrix} \boldsymbol{A} - \boldsymbol{I}_n & \boldsymbol{B} \\ \boldsymbol{C} & \mathbf{0}_{2\times m} \end{bmatrix}$$

is invertible. This is trivially the case for systems with integrator dynamics. ◊

The objective of the control algorithm then follows the same idea as the one introduced in Nguyen (2016). The MAS is deployed inside a convex bounded polytope $\mathcal{Y}$ as defined in Assumption 3.1. A discrete-time control law is applied to the MAS to drive it towards a static final configuration. From the knowledge of the position of each agent, it is possible to compute the Voronoi tessellation of $\mathcal{Y}$ by defining the Voronoi cell $\mathcal{V}_i(k)$, with $i \in \overline{1, N}$, of each agent. In addition to its initial definition in Section 2.4.1, a time dependency is added to the Voronoi cell. Indeed, since the agents move inside the workspace $\mathcal{Y}$, the Voronoi tessellation is time-varying. From the knowledge of the Voronoi tessellation, the Chebyshev center can be obtained from (2.56) with a slight modification in the centralized case which is presented below in (3.6). A control input for the MAS is then computed to drive each agent towards the Chebyshev center of its time-varying cell while constrained to remain inside it.

As in Section 2.3, the dynamics of the agents can be gathered in one general state-space representation of the MAS:

$$\mathbf{x}(k+1) = \boldsymbol{A}_\Sigma \mathbf{x}(k) + \boldsymbol{B}_\Sigma \mathbf{u}(k) \tag{3.3a}$$
$$\mathbf{y}(k) = \boldsymbol{C}_\Sigma \mathbf{x}(k) \tag{3.3b}$$

where $\mathbf{x}$, $\mathbf{u}$ and $\mathbf{y}$ are defined as in Section 2.3 with $\mathbf{x}(k) = \begin{bmatrix}\mathbf{x}_1(k)^\top & \cdots & \mathbf{x}_N^\top(k)\end{bmatrix}^\top \in \mathbb{R}^{Nn}$, $\mathbf{u}(k) = \begin{bmatrix}\mathbf{u}_1(k)^\top & \cdots & \mathbf{u}_N^\top(k)\end{bmatrix}^\top \in \mathbb{R}^{Nm}$ and $\mathbf{y}(k) = \begin{bmatrix}\mathbf{y}_1(k)^\top & \cdots & \mathbf{y}_N^\top(k)\end{bmatrix}^\top \in \mathbb{R}^{2N}$, and the matrices expressions can be simplified with respect to the ones given in Section 2.3 since now $\boldsymbol{A}_\Sigma = \boldsymbol{I}_N \otimes \boldsymbol{A}$, $\boldsymbol{B}_\Sigma = \boldsymbol{I}_N \otimes \boldsymbol{B}$ and $\boldsymbol{C}_\Sigma = \boldsymbol{I}_N \otimes \boldsymbol{C}$, the matrix $\boldsymbol{C}$ being defined as in Remark 2.10.

From the output vector $\mathbf{y}_i$, with $i \in \overline{1, N}$, of each agent, the Voronoi tessellation is computed as presented in (2.45) such that:

$$\mathcal{V}_i(k) = \left\{ \mathbf{y} \in \mathcal{Y} \;\middle|\; (\mathbf{y}_j(k) - \mathbf{y}_i(k))^\top \mathbf{y} \leq \frac{1}{2}\left(\|\mathbf{y}_j(k)\|_2^2 - \|\mathbf{y}_i(k)\|_2^2\right),\ \forall j \in \overline{1, N},\ i \neq j \right\}$$

with $i \in \overline{1, N}$ which is reduced to the more compact H-representation:

$$\mathcal{V}_i(k) = \{\mathbf{y} \in \mathcal{Y} \mid \boldsymbol{H}_i(k)\mathbf{y} \leq \boldsymbol{\theta}_i(k)\} \tag{3.4}$$

where $\boldsymbol{H}_i(k) \in \mathbb{R}^{m_i(k)\times 2}$ and $\boldsymbol{\theta}_i(k) \in \mathbb{R}^{m_i(k)}$, with $m_i(k)$ the number of sides of the cell $\mathcal{V}_i(k)$. For the detailed construction of such a cell, the reader can refer to Section 2.4.1.

The next step rests on the computation of the Chebyshev center of each Voronoi cell. For a centralized control algorithm, all the computations are run on a single machine and the result is sent to each agent. They then have to implement the control input that has been found for them by the central machine. Thus, depending on the computing capabilities of this machine, the Chebyshev center can be computed by solving sequentially or in parallel the N linear problems (2.56):

$$\begin{aligned}
&\underset{\mathbf{c}_i(k),\, r_i(k)}{\text{maximize}} && r_i(k) \\
&\text{subject to} && \\
&\quad r_i(k) \geq 0, && \\
&\theta_{i,j}(k) \geq \boldsymbol{h}_{i,j}(k)\mathbf{c}_i(k) + \left\|\boldsymbol{h}_{i,j}^\top(k)\right\|_2 r_i(k), && \forall j \in \overline{1, m_i(k)}
\end{aligned} \tag{3.5}$$

where $\theta_{i,j}(k)$ and $\boldsymbol{h}_{i,j}(k)$, with $j \in \overline{1, m_i(k)}$ and $i \in \overline{1, N}$, are the rows of $\boldsymbol{\theta}_i(k)$ and $\boldsymbol{H}_i(k)$ introduced in (3.4) and $\mathbf{c}_i(k)$ and $r_i(k)$, which are a shortened form of $\mathbf{c}_{\mathcal{V}_i(k)}$ and $r_{\mathcal{V}_i(k)}$, are the *Chebyshev center* and *Chebyshev radius* of $\mathcal{V}_i(k)$.

Since the control algorithm is meant here to be centralized, the computation of the Chebyshev centers can also be centralized. This centralization leads to a slightly modified version of the previous optimization problem:

$$\begin{aligned}
&\underset{\substack{\mathbf{c}_i(k), r_i(k),\\ \forall i \in \overline{1,N}}}{\text{maximize}} && \sum_{i=1}^{N} r_i(k) \\
&\text{subject to} && \\
&\quad r_i(k) \geq 0, && \forall i \in \overline{1, N}, \\
&\theta_{i,j}(k) \geq \boldsymbol{h}_{i,j}(k)\mathbf{c}_i(k) + \left\|\boldsymbol{h}_{i,j}^{\top}(k)\right\|_2 r_i(k), && \forall j \in \overline{1, m_i(k)}, \forall i \in \overline{1, N}
\end{aligned} \tag{3.6}$$

It is immediate that running N problems of type (3.5) is equivalent to running one problem (3.6) since (3.6) maximizes the sum of N independent real positive variables. In the following, $\mathbf{c}(k)$ denotes the aggregated vector of Chebyshev centers such that $\mathbf{c}(k) = \begin{bmatrix}\mathbf{c}_1(k)^{\top} & \cdots & \mathbf{c}_N(k)^{\top}\end{bmatrix}^{\top} \in \mathbb{R}^{2N}$. The objective of each agent $i \in \overline{1, N}$ is to track the movement of the Chebyshev center $\mathbf{c}_i(k)$ of its Voronoi cell $\mathcal{V}_i(k)$ to obtain $\mathbf{y}_i(k) = \mathbf{c}_i(k)$ for all $i \in \overline{1, N}$ or, in a centralized fashion, $\mathbf{y}(k) = \mathbf{c}(k)$. This is done by imposing that at all time instant k, the Chebyshev center $\mathbf{c}_i(k)$ is an equilibrium point for agent $i \in \overline{1, N}$ obeying the dynamics (3.1). With Assumption 3.2, it is possible to obtain the couple $(\mathbf{x}_{\mathbf{c}_i}(k), \mathbf{u}_{\mathbf{c}_i}(k))$ associated with the equilibrium point $\mathbf{c}_i(k)$ for all $i \in \overline{1, N}$, this couple being used as elements of the reference trajectories for the state and input of agent i. The Chebyshev center tracking continues until the MAS reaches a static configuration, i.e. until there is no further evolution of the Voronoi cells $\mathcal{V}_i(k)$ and thus of the Chebyshev centers $\mathbf{c}_i(k)$. From Assumption 3.2, it is possible to define the couple $(\mathbf{x}_{\mathbf{c}}(k), \mathbf{u}_{\mathbf{c}}(k))$ for the MAS obeying the dynamics (3.3) such that:

$$\begin{aligned}
\mathbf{x}_{\mathbf{c}}(k) &= \boldsymbol{A}_{\Sigma}\mathbf{x}_{\mathbf{c}}(k) + \boldsymbol{B}_{\Sigma}\mathbf{u}_{\mathbf{c}}(k) \\
\mathbf{c}(k) &= \boldsymbol{C}_{\Sigma}\mathbf{x}_{\mathbf{c}}.
\end{aligned} \tag{3.7}$$

With all the elements computed above, the input $\mathbf{u}(k)$ from (3.3a) is computed by finding the solution of the MPC optimization problem:

$$\underset{\substack{\mathbf{u}(k+l),\\ \forall l \in \overline{0, N_p-1}}}{\text{minimize}} \quad \sum_{l=0}^{N_p-1} \ell(\mathbf{x}(k+l), \mathbf{u}(k+l), \mathbf{x}_{\mathbf{c}}(k), \mathbf{u}_{\mathbf{c}}(k)) + V(\mathbf{x}(k+N_p), \mathbf{x}_{\mathbf{c}}(k)) \tag{3.8a}$$

subject to

$$\mathbf{x}(k+l+1) = \boldsymbol{A}_{\Sigma}\mathbf{x}(k+l) + \boldsymbol{B}_{\Sigma}\mathbf{u}(k+l), \qquad \forall l \in \overline{0, N_p-1}, \tag{3.8b}$$

$$\mathbf{x}(k+l) \in \mathcal{X}^N, \qquad \forall l \in \overline{0, N_p-1}, \tag{3.8c}$$

$$\mathbf{u}(k+l) \in \mathcal{U}^N, \qquad \forall l \in \overline{0, N_p-1}, \tag{3.8d}$$

$$\boldsymbol{C}_{\Sigma}\mathbf{x}(k+l) \in \mathcal{V}_1(k) \times \cdots \times \mathcal{V}_N(k), \qquad \forall l \in \overline{0, N_p-1}, \tag{3.8e}$$

$$\mathbf{x}(k+N_p) \in \Omega(k) \tag{3.8f}$$

where $\ell(\mathbf{x}(k+l), \mathbf{u}(k+l), \mathbf{x_c}(k), \mathbf{u_c}(k))$, with $l \in \overline{0, N_p - 1}$, is the stage cost:

$$\ell(\mathbf{x}(k+l), \mathbf{u}(k+l), \mathbf{x_c}(k), \mathbf{u_c}(k)) = \|\mathbf{x}(k+l) - \mathbf{x_c}(k)\|^2_{\boldsymbol{Q}} + \|\mathbf{u}(k+l) - \mathbf{u_c}(k)\|^2_{\boldsymbol{R}} \quad (3.9)$$

and $V(\mathbf{x}(k+N_p), \mathbf{x_c}(k))$ is the terminal cost:

$$V(\mathbf{x}(k+N_p), \mathbf{x_c}(k)) = \|\mathbf{x}(k+N_p) - \mathbf{x_c}(k)\|^2_{\boldsymbol{P}}. \quad (3.10)$$

The weighting matrices $\boldsymbol{Q}, \boldsymbol{P} \in \mathbb{R}^{Nn \times Nn}$ and $\boldsymbol{R} \in \mathbb{R}^{Nm \times Nm}$ in (3.9) and (3.10) are chosen such that $\boldsymbol{Q} = \boldsymbol{Q}^\top \succ 0$, $\boldsymbol{P} = \boldsymbol{P}^\top \succ 0$ and $\boldsymbol{R} = \boldsymbol{R}^\top \succ 0$. The prediction horizon N_p is a positive integer.

The constraint (3.8b) is used to predict the future state $\mathbf{x}(k+l+1)$ for all $l \in \overline{0, N_p - 1}$ of the MAS given the value of the input sequence $\mathbf{u}(k+l)$ for all $l \in \overline{0, N_p - 1}$. The other constraints are meant to restrict the movement of the agents of the MAS.

Constraint (3.8e) is meant to ensure that, over the prediction horizon N_p, the output of each agent lies inside the agent's Voronoi cell $\mathcal{V}_i(k)$, with $i \in \overline{1, N}$, computed at time k.

The constraint (3.8c) constrains the state vector of the MAS inside a convex polytope $\mathcal{X}^N$ over the prediction horizon, where $\mathcal{X}$ is the state space of the agents. However, the constraint (3.8e) is more restrictive than $\boldsymbol{C}_\Sigma \mathbf{x}(k+l) \in \boldsymbol{C}_\Sigma \mathcal{X}^N$ for all $l \in \overline{0, N_p - 1}$. Indeed, $\boldsymbol{C}_\Sigma \mathcal{X}^N = \mathcal{Y}^N$ which means, at the level of an agent, that (3.8c) imposes that the position of each agent over the prediction horizon evolves inside the workspace $\mathcal{Y}$, i.e. $\boldsymbol{C}\mathbf{x}_i(k+l) \in \mathcal{Y}$ for all $l \in \overline{0, N_p - 1}$. Since, by definition, $\mathcal{V}_i(k) \subset \mathcal{Y}$ for all $i \in \overline{1, N}$, the constraint (3.8e) is more restrictive on $\boldsymbol{C}_\Sigma \mathbf{x}(k+l)$ than (3.8c). Then, given the structure of the state vector given in Assumption 2.4, constraint (3.8c) constrains the other states than those participating in the output.

Constraint (3.8d), ensures that the input signal of the MAS remains inside a convex polytope $\mathcal{U}^N$, where $\mathcal{U}$ is the input space of the agents.

Finally, (3.8f) is a terminal constraint, with $\Omega(k)$ the terminal set, added for stability. The expression of the terminal set is discussed in Section 3.2.3 in the decentralized case that is presented in Section 3.2.2 with an extension to the present centralized case.

With standard MPC policy, the first element $\mathbf{u}(k)$ obtained from (3.8) is applied to the MAS. As mentioned in Section 2.3, although the agents in the MAS appear to be decoupled in the open-loop equations (3.3), the closed-loop control policy (3.8) induces a coupling between the agents by means of the constraint (3.8e) and the objective point $(\mathbf{x_c}(k), \mathbf{u_c}(k), \mathbf{c}(k))$, which depends itself on the Voronoi cells by means of equation (3.6). The whole procedure can be summarized by Algorithm 3.1, this algorithm being applied at each time instant.

Remark 3.2: Output constraint
The Voronoi cells $\mathcal{V}_i(k)$ for all $i \in \overline{1, N}$ are considered constant over the prediction horizon N_p for the optimization problem (3.8) at time k. ♢

Remark 3.3: Different stage cost
The stage cost function (3.9) can be modified to take into account additional elements to be minimized. For example, Chevet et al. (2018) or Chevet et al. (2020b) add a term of the form $\|\mathbf{u}(k+l+1) - \mathbf{u}(k+l)\|^2_{\boldsymbol{S}}$, where $\boldsymbol{S} \in \mathbb{R}^{Nm \times Nm}$ such that $\boldsymbol{S} = \boldsymbol{S}^\top \succ 0$, to smooth the evolution of the control input. ♢

Algorithm 3.1: Centralized nominal MPC algorithm.

Input: The current state $\mathbf{x}(k)$ of the MAS
for $i \in \overline{1, N}$ **do**
  Compute the Voronoi cell $\mathcal{V}_i(k)$ of agent i with (2.45);
end
Compute the aggregated vector of Chebyshev centers $\mathbf{c}(k)$ with (3.6);
Compute the couple $(\mathbf{x}_\mathbf{c}(k), \mathbf{u}_\mathbf{c}(k))$ with (3.7) such that $(\mathbf{x}_\mathbf{c}(k), \mathbf{u}_\mathbf{c}(k), \mathbf{c}(k))$ is an equilibrium point of (3.3);
Solve the optimization problem (3.8);
Output: The input signal $\mathbf{u}(k)$

3.2.2 Decentralized algorithm

Often, centralized policies for control of MAS are not the most suitable control policies. Indeed, when the goal is to drive a group of agents towards a given configuration, a centralized control algorithm can be limited by several factors covering for example the heavy computational load for the central computing unit or the delays or loss of communication between this central unit and one or several agents. This is why distributed or decentralized policies are preferred since they can, in some instances, reduce the communication time between the machines while increasing the robustness of the global MAS and the tolerance to faults. Here, due to Assumption 2.5, a decentralized policy is chosen: each agent knows the position of the other agents of the MAS, thus each agent is able to independently compute its Voronoi cell and its associated Chebyshev center and can then solve its own MPC optimization problem.

Let $i \in \overline{1, N}$ be an agent of the MAS. Then the computation of its Voronoi cell $\mathcal{V}_i(k)$ and its Chebyshev center $\mathbf{c}_i(k)$ have already been covered in Section 3.2.1, the agent computing $\mathbf{c}_i(k)$ by solving (3.5).

With Assumption 3.2, it is possible to find the couple $(\mathbf{x}_{\mathbf{c}_i}(k), \mathbf{u}_{\mathbf{c}_i}(k))$ such that $(\mathbf{x}_{\mathbf{c}_i}(k), \mathbf{u}_{\mathbf{c}_i}(k), \mathbf{c}_i(k))$ is an equilibrium point of the dynamics (3.1):

$$\begin{aligned} \mathbf{x}_{\mathbf{c}_i}(k) &= \boldsymbol{A}\mathbf{x}_{\mathbf{c}_i}(k) + \boldsymbol{B}\mathbf{u}_{\mathbf{c}_i}(k) \\ \mathbf{c}_i(k) &= \boldsymbol{C}\mathbf{x}_{\mathbf{c}_i}(k). \end{aligned} \tag{3.11}$$

With all the elements computed above, the input $\mathbf{u}_i(k)$ for agent $i \in \overline{1, N}$ from (3.1) is computed by finding the solution of the optimization problem:

$$\underset{\substack{\mathbf{u}_i(k+l), \\ \forall l \in \overline{0, N_p-1}}}{\text{minimize}} \quad \sum_{l=0}^{N_p-1} \ell(\mathbf{x}_i(k+l), \mathbf{u}_i(k+l), \mathbf{x}_{\mathbf{c}_i}(k), \mathbf{u}_{\mathbf{c}_i}(k)) + V(\mathbf{x}_i(k+N_p), \mathbf{x}_{\mathbf{c}_i}(k)) \tag{3.12a}$$

subject to

$$\mathbf{x}_i(k+l+1) = \boldsymbol{A}\mathbf{x}_i(k+l) + \boldsymbol{B}\mathbf{u}_i(k+l), \qquad \forall l \in \overline{0, N_p-1}, \tag{3.12b}$$

$$\mathbf{x}_i(k+l) \in \mathcal{X}, \qquad \forall l \in \overline{0, N_p-1}, \tag{3.12c}$$

$$\mathbf{u}_i(k+l) \in \mathcal{U}, \qquad \forall l \in \overline{0, N_p-1}, \tag{3.12d}$$

$$\boldsymbol{C}\mathbf{x}_i(k+l) \in \mathcal{V}_i(k), \qquad \forall l \in \overline{0, N_p-1}, \tag{3.12e}$$

$$\mathbf{x}_i(k+N_p) \in \Omega_i(k) \tag{3.12f}$$

where $\ell(\mathbf{x}_i(k+l), \mathbf{u}_i(k+l), \mathbf{x}_{\mathbf{c}_i}(k), \mathbf{u}_{\mathbf{c}_i}(k))$, with $l \in \overline{0, N_p - 1}$, is the stage cost:

$$\ell(\mathbf{x}_i(k+l), \mathbf{u}_i(k+l), \mathbf{x}_{\mathbf{c}_i}(k), \mathbf{u}_{\mathbf{c}_i}(k)) = \|\mathbf{x}_i(k+l) - \mathbf{x}_{\mathbf{c}_i}(k)\|^2_{\boldsymbol{Q}_i} + \|\mathbf{u}_i(k+l) - \mathbf{u}_{\mathbf{c}_i}(k)\|^2_{\boldsymbol{R}_i} \quad (3.13)$$

and $V(\mathbf{x}_i(k+N_p), \mathbf{x}_{\mathbf{c}_i}(k))$ is the terminal cost:

$$V(\mathbf{x}_i(k+N_p), \mathbf{x}_{\mathbf{c}_i}(k)) = \|\mathbf{x}_i(k+N_p) - \mathbf{x}_{\mathbf{c}_i}(k)\|^2_{\boldsymbol{P}_i}. \quad (3.14)$$

The weighting matrices $\boldsymbol{Q}_i, \boldsymbol{P}_i \in \mathbb{R}^{n\times n}$ and $\boldsymbol{R}_i \in \mathbb{R}^{m\times m}$ in (3.13) and (3.14) are chosen such that $\boldsymbol{Q}_i = \boldsymbol{Q}_i^\top \succ 0$, $\boldsymbol{P}_i = \boldsymbol{P}_i^\top \succ 0$ and $\boldsymbol{R}_i = \boldsymbol{R}_i^\top \succ 0$. The prediction horizon N_p is a positive integer.

Given the shape of the optimization problem (3.12), it is immediate that the goal of the cost function (3.12a) and the constraints (3.12b)-(3.12f) in the decentralized case (3.12) is identical to the goal of the same elements in the centralized case (3.8). The difference between (3.8) and (3.12) lies in the fact that the constraints of (3.8) are applied to the state, input and output vectors of the entire MAS Σ, while the constraints of (3.12) only restrain the state, input and output vectors of one agent $i \in \overline{1, N}$ of Σ. Then, since problem (3.12) is reduced to only one agent, the control algorithm can be embedded into each agent, thus reducing the computational burden and communication time.

For all $l \in \overline{0, N_p - 1}$, the constraint (3.12b) predicts the future state $\mathbf{x}_i(k+l+1)$ of the agent i given the value of the input sequence $\mathbf{u}_i(k+l)$.

Constraint (3.12e) is meant to ensure that, over the prediction horizon N_p, the output of agent i belongs to the agent's Voronoi cell $\mathcal{V}_i(k)$ computed at time k.

The constraint (3.12c) restricts the state of agent i to remain inside a convex polytope $\mathcal{X}$ over the prediction horizon. However, as discussed in Section 3.2.1, (3.12e) is more restrictive on $\boldsymbol{C}\mathbf{x}_i(k+l)$ for all $l \in \overline{0, N_p - 1}$ than (3.12c). Then, the constraint (3.12c) constrains the other states than those participating in the output.

Constraint (3.12d) ensures that the input signal of the agent remains inside a convex polytope $\mathcal{U}$, the input space of the agent.

Finally, the terminal constraint (3.12f) is added for stability purposes. This constraint is discussed in Section 3.2.3.

Remark 3.4: Weighting matrices
Per Assumption 2.6, all the agents of Σ share the same dynamics, hence, without loss of generality, it is simpler to consider, $\boldsymbol{Q}_i = \boldsymbol{Q}$, $\boldsymbol{R}_i = \boldsymbol{R}$ and $\boldsymbol{P}_i = \boldsymbol{P}$ for all $i \in \overline{1, N}$. ♢

Remark 3.5: Output constraint
As in the centralized case, the Voronoi cells $\mathcal{V}_i(k)$, $\forall i \in \overline{1, N}$, are considered constant over the prediction horizon N_p for the optimization problem (3.12) at time k. ♢

Then, Algorithm 3.2 summarizes the entire control procedure for an agent. This procedure is applied for each agent at each time instant (it is assumed that the agents' clocks are synchronized).

3.2.3 Discussion on the convergence of the decentralized algorithm

The contribution of the following paragraph is to provide a framework for the proof of convergence of the decentralized algorithm of Section 3.2.2 (though it can be easily

Algorithm 3.2: Decentralized nominal MPC algorithm for one agent.

Input: The current state $\mathbf{x}_i(k)$ of agent i

1 Acquire the position of the other agents $\mathbf{y}_j(k)$, with $j \in \overline{1,N}$, $j \neq 1$;
2 Compute the Voronoi cell $\mathcal{V}_i(k)$ of agent i with (2.45);
3 Compute the Chebyshev center $\mathbf{c}_i(k)$ of agent i with (3.5);
4 Compute the couple $(\mathbf{x}_{\mathbf{c}_i}(k), \mathbf{u}_{\mathbf{c}_i}(k))$ with (3.11) such that $(\mathbf{x}_{\mathbf{c}_i}(k), \mathbf{u}_{\mathbf{c}_i}(k), \mathbf{c}_i(k))$ is an equilibrium point of (3.1);
5 Solve the optimization problem (3.12);

Output: The input signal $\mathbf{u}_i(k)$ of agent i

modified to be extended to the centralized algorithm of Section 3.2.1) for a specific dynamics: the single integrator dynamics. This framework stems from the works of Nguyen (2016) and Hatleskog (2018), the latter providing a proof of convergence for the deployment of a MAS when a linear state-feedback controller is used instead of MPC.

Before starting the convergence analysis of the proposed algorithm, it is necessary to introduce an existing result on a generalization of the definition of the controlled λ-contractiveness of a set given in Definition 2.25.

Definition 3.1: Generalized controlled λ-contractive set (Nguyen, 2016)

A convex set $\mathcal{Y} \subset \mathbb{R}^p$ is said to be controlled λ-contractive, with $\lambda \in [0,1)$, for the dynamics (3.1) if for any $\mathbf{y}_0 \in \mathcal{Y}$ and any $\mathbf{x}(k) \in \mathcal{X} \subset \mathbb{R}^n$ such that $\boldsymbol{C}\mathbf{x}(k) = \mathbf{y}(k) \in \mathcal{Y}$, there exists a control law $\mathbf{u}(k) \in \mathcal{U} \subset \mathbb{R}^m$ such that $\mathbf{y}(k+1) \in \{\mathbf{y}_0\} \oplus \lambda(\mathcal{Y} \oplus \{-\mathbf{y}_0\})$.

In his work, Hatleskog (2018) considers that the control input for an agent $i \in \overline{1,N}$ obeying the dynamics (3.1) is given by $\mathbf{u}_i(k) = \mathcal{K}(\mathbf{x}_i(k), \mathcal{V}_i(k))$, where $\mathcal{K}$ is a continuous function. Then, to prove the convergence of the MAS to a static Chebyshev configuration, Hatleskog (2018) assumes four regularity conditions. Three of these conditions have already been provided:

- the system (3.1) is controllable according to Assumption 2.1;
- the system (3.1) is observable according to Assumption 2.2;
- the system (3.1) admits equilibrium points according to Assumption 3.2.

The last regularity condition is based on the generalization of the notion of controlled λ-contractiveness of a set given in Definition 3.1.

Assumption 3.3: λ-contractiveness of the Voronoi cells

For any agent $i \in \overline{1,N}$ of the MAS, its Voronoi cell $\mathcal{V}_i(k)$ is controlled λ-contractive with respect to the dynamics (3.1).

One of the few dynamics that verifies such restrictive conditions is the single integrator dynamics of the type:

$$\mathbf{x}_i(k+1) = \mathbf{x}_i(k) + T_s\mathbf{u}_i(k) \tag{3.15a}$$
$$\mathbf{y}_i(k) = \mathbf{x}_i(k) \tag{3.15b}$$

where T_s is the sampling rate. In this case, the state vector $\mathbf{x}_i(k) \in \mathbb{R}^2$, with $i \in \overline{1, N}$, is also the output $\mathbf{y}_i(k)$ of the system, i.e. the position of agent i in the plane $\mathbb{R}^2$, while the input vector $\mathbf{u}_i(k) \in \mathbb{R}^2$ is the horizontal velocity of agent i. While these dynamics might seem simplistic, numerous vehicle dynamics can be reduced to single integrator dynamics of type (3.15) (Ren and Beard, 2008). In the following, all the agents are assumed to obey the single integrator dynamics (3.15).

In Hatleskog (2018), the chosen control law $\mathbf{u}_i(k) = \mathcal{K}(\mathbf{x}_i(k), \mathcal{V}_i(k))$ is an unconstrained state-feedback such that $\mathbf{u}_i(k) = \boldsymbol{K}(\mathbf{x}_i(k) - \mathbf{c}_i(k))$, with $\mathbf{c}_i(k)$ the Chebyshev center of the Voronoi cell $\mathcal{V}_i(k)$. The gain matrix $\boldsymbol{K} \in \mathbb{R}^{2\times 2}$ can be chosen by any method as long as the closed-loop dynamics is stable (Hatleskog, 2018). Since the control law proposed in the present thesis is subject to constraints, additional assumptions have to be made.

Assumption 3.4: Shape of the input constraints

The input constraints set $\mathcal{U}$ used in (3.12d) is a box $\mathbb{B}^2(\boldsymbol{\alpha}_u)$, where $\boldsymbol{\alpha}_u \in \mathbb{R}^2$.

Remark 3.6: Shape of the state constraints
From the dynamics (3.15), the state space $\mathcal{X}$ and the output space $\mathcal{Y}$ are equal. Then, in the case of single integrator dynamics, the constraint (3.12c) is redundant with (3.12e) and it is dropped. ◊

Assumption 3.5: Terminal constraint

The set Ω_i, with $i \in \overline{1, N}$, in the terminal constraint (3.12f) is chosen to be:

$$\Omega_i(k) = \{\mathbf{c}_i(k)\} \oplus \lambda_i(\mathcal{V}_i(k) \oplus \{-\mathbf{c}_i(k)\}), \tag{3.16}$$

with $\lambda_i \in [0, 1)$, where $\mathbf{c}_i(k)$ is the Chebyshev center of agent i's Voronoi cell $\mathcal{V}_i(k)$.

Given Assumption 3.3, it is guaranteed that there exists a real $\lambda_i \in [0, 1)$ such that (3.12f) is feasible.

Remark 3.7: Choice of the value of the contraction factor
The contraction factor $\lambda_i \in [0, 1)$, with $i \in \overline{1, N}$, can be chosen in different ways. A first approach would be to chose λ_i such that $\Omega_i(k)$ is reachable in N_p steps by the system. Another approach would be to add $\lambda_i(k)$ as a decision variable in (3.12) by adding $\lambda_i(k)$ to the cost function (3.12a) to be minimized and adding:

$$0 \leq \lambda_i(k) < 1 \tag{3.17}$$

as a constraint. ◊

Theorem 3.1: Feasibility of the decentralized MPC optimization problem

If the agents in the MAS obey the dynamics (3.15) and Assumptions 3.3 to 3.5 are verified, then the optimization problem (3.12) is always feasible.

Proof. For the optimization problem (3.12) to be feasible, the set of constraints (3.12c)-(3.12f) has to be satisfied. The claim that problem (3.12) is always feasible is mathematically translated to the claim that the constraint set is never empty. For the remainder of this proof, $\mathbf{x}_i$, with $i \in \overline{1, N}$, is used instead of $\mathbf{y}_i$ since the system obeys the dynamics (3.15), where $\mathbf{y}_i = \mathbf{x}_i$.

The first step of the proof is to obtain the H-representation of the sets $\mathcal{V}_i(k)$ and $\Omega_i(k)$ appearing in constraints (3.12e) and (3.12f) to formulate those two constraints as linear inequalities. To do so, the expression of the Voronoi cell $\mathcal{V}_i(k)$ given in (2.45) is used. From this equation the rows of the matrices $\boldsymbol{H}_i(k)$ and $\boldsymbol{\theta}_i(k)$ inducing the Voronoi cell $\mathcal{V}_i(k)$ of agent i are written:

$$\boldsymbol{h}_{i,j}(k) = (\mathbf{x}_j(k) - \mathbf{x}_i(k))^\top \tag{3.18}$$

$$\boldsymbol{\theta}_{i,j}(k) = \frac{1}{2}\big(\|\mathbf{x}_j(k)\|_2^2 - \|\mathbf{x}_i(k)\|_2^2\big), \tag{3.19}$$

with $j \in \overline{1, N} \setminus \{i\}$. However, (3.19) can be written:

$$\begin{aligned}
\boldsymbol{\theta}_{i,j}(k) &= \frac{1}{2}\big(\|\mathbf{x}_j(k)\|_2^2 - \|\mathbf{x}_i(k)\|_2^2\big) = \frac{1}{2}(\mathbf{x}_j(k) - \mathbf{x}_i(k))^\top(\mathbf{x}_j(k) + \mathbf{x}_i(k)) \\
&= (\mathbf{x}_j(k) - \mathbf{x}_i(k))^\top\left(\frac{1}{2}\mathbf{x}_j(k) + \left(1 - \frac{1}{2}\right)\mathbf{x}_i(k)\right) \\
&= \frac{1}{2}\underbrace{(\mathbf{x}_j(k) - \mathbf{x}_i(k))^\top}_{\boldsymbol{h}_{i,j}(k)}(\mathbf{x}_j(k) - \mathbf{x}_i(k)) + (\mathbf{x}_j(k) - \mathbf{x}_i(k))^\top\mathbf{x}_i(k) \\
&= \frac{1}{2}\big\|\boldsymbol{h}_{i,j}(k)^\top\big\|_2^2 + \boldsymbol{h}_{i,j}(k)\mathbf{x}_i(k)
\end{aligned}$$

giving, according to Remark 2.8:

$$\mathcal{V}_i(k) = \{\mathbf{x}_i(k)\} \oplus \mathcal{C}_i(k) \tag{3.20}$$

where:

$$\mathcal{C}_i(k) = \big\{\mathbf{x} \in \mathbb{R}^2 \,\big|\, \boldsymbol{H}_i(k)\mathbf{x} \leq \boldsymbol{\kappa}_i(k)\big\}$$

with the elements of $\boldsymbol{\kappa}_i(k) \in \mathbb{R}^{N-1}$ being the $\kappa_{i,j}(k) = \frac{1}{2}\big\|\boldsymbol{h}_{i,j}(k)^\top\big\|_2^2 \geq 0$ for all $j \in \overline{1, N} \setminus \{i\}$. Using Remark 2.8, $\mathcal{V}_i(k)$ can be rewritten such that:

$$\mathcal{V}_i(k) = \big\{\mathbf{x} \in \mathbb{R}^2 \,\big|\, \boldsymbol{H}_i(k)\mathbf{x} \leq \boldsymbol{\kappa}_i(k) + \boldsymbol{H}_i(k)\mathbf{x}_i(k)\big\}.$$

Moreover, given the definition of $\Omega_i(k)$ in Assumption 3.5, it can be verified with Remark 2.8 and Property 2.10 that:

$$\Omega_i(k) = \{\mathbf{x} \in \mathbb{R}^n \mid \boldsymbol{H}_i(k)\mathbf{x} \leq \lambda_i\boldsymbol{\kappa}_i(k) + \lambda_i\boldsymbol{H}_i(k)\mathbf{x}_i(k) + (1 - \lambda_i)\boldsymbol{H}_i(k)\mathbf{c}_i(k)\}$$

or, equivalently:

$$\Omega_i(k) = \{\lambda_i\mathbf{x}_i(k) + (1 - \lambda_i)\mathbf{c}_i(k)\} \oplus \lambda_i\mathcal{C}_i(k).$$

These results will come in handy later in this proof.

For the next part of the proof, the feasibility of the constraints (3.12e) and (3.12f) is studied separately. Indeed, while the constraint on the input vector (3.12d) is a constraint on $\mathbf{u}_i(k + l)$ for all $l \in \overline{0, N_p - 1}$, the constraint on the output (or state vector in the present case) (3.12e) is a constraint on $\mathbf{u}_i(k + l)$ for all $l \in \overline{0, N_p - 2}$ and the terminal constraint (3.12f) is a constraint on $\mathbf{u}_i(k + N_p - 1)$. Then, as a first step, the constraints (3.12b), (3.12e) and (3.12d) are aggregated over the horizon $\overline{0, N_p - 2}$ before studying (3.12f) along with (3.12b) and (3.12d) for $l = N_p - 1$.

Consider:

$$\mathbf{X}_i(k) = \begin{bmatrix} \mathbf{x}_i(k+1) \\ \vdots \\ \mathbf{x}_i(k+N_p-1) \end{bmatrix} \qquad \mathbf{U}_i(k) = \begin{bmatrix} \mathbf{u}_i(k) \\ \vdots \\ \mathbf{u}_i(k+N_p-2) \end{bmatrix}$$

$$\boldsymbol{F} = \mathbf{1}_{N_p-1\times 1} \otimes \boldsymbol{I}_2 \qquad \boldsymbol{G} = T_s \cdot \begin{bmatrix} 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ 1 & \cdots & \cdots & 1 \end{bmatrix} \otimes \boldsymbol{I}_2$$

such that the constraint (3.12b) can be rewritten, for all $l \in \overline{0, N_p-2}$:

$$\mathbf{X}_i(k) = \boldsymbol{F}\mathbf{x}_i(k) + \boldsymbol{G}\mathbf{U}_i(k). \tag{3.21}$$

Thus, since $\mathbf{x}_i = \mathbf{y}_i$, the constraints (3.12d)-(3.12e) can be aggregated as:

$$\boldsymbol{H}_i^h(k)\mathbf{X}_i(k) \leq \boldsymbol{\kappa}_i^h(k) + \boldsymbol{H}_i^h(k)\big(\mathbf{1}_{(N_p-1)\times 1} \otimes \mathbf{x}_i(k)\big) \tag{3.22a}$$

$$\boldsymbol{H}_{\mathcal{U}}^h(k)\mathbf{U}_i(k) \leq \boldsymbol{\theta}_{\mathcal{U}}^h(k) \tag{3.22b}$$

where $\boldsymbol{H}_i^h(k) = \boldsymbol{I}_{N_p-1} \otimes \boldsymbol{H}_i(k)$, $\boldsymbol{\kappa}_i^h(k) = \mathbf{1}_{(N_p-1)\times 1} \otimes \boldsymbol{\kappa}_i(k)$, $\boldsymbol{H}_{\mathcal{U}}^h(k) = \boldsymbol{I}_{N_p-1} \otimes \boldsymbol{H}_{\mathcal{U}}(k)$ and $\boldsymbol{\theta}_{\mathcal{U}}^h(k) = \mathbf{1}_{(N_p-1)\times 1} \otimes \boldsymbol{\theta}_{\mathcal{U}}(k)$, with $\boldsymbol{H}_{\mathcal{U}}(k)$ and $\boldsymbol{\theta}_{\mathcal{U}}(k)$ the matrices inducing $\mathcal{U}$ in $\mathbb{R}^2$. It can be noticed that, with dynamics (3.15), $\mathbf{1}_{(N_p-1)\times 1} \otimes \mathbf{x}_i(k) = \boldsymbol{F}\mathbf{x}_i(k)$. Injecting (3.21) into (3.22a), the inequalities (3.22) become:

$$\begin{aligned} \boldsymbol{H}_i^h(k)\boldsymbol{G}\mathbf{U}_i(k) &\leq \boldsymbol{\kappa}_i^h(k) \\ \boldsymbol{H}_{\mathcal{U}}^h(k)\mathbf{U}_i(k) &\leq \boldsymbol{\theta}_{\mathcal{U}}^h(k). \end{aligned}$$

Given its definition, $\boldsymbol{\kappa}_i^h(k) \geq 0$ and, since $\mathcal{U}$ is a box, $\boldsymbol{\theta}_{\mathcal{U}}^h(k) \geq 0$, by Definition 2.12. Then, there exists no vector $\mathbf{y} \geq 0$ such that $\begin{bmatrix} \boldsymbol{\kappa}_i^h(k)^\top & \boldsymbol{\theta}_{\mathcal{U}}^h(k)^\top \end{bmatrix} \mathbf{y} < 0$ and, by Theorem 2.2, the set of constraints:

$$\left\{ \mathbf{U} \in \mathbb{R}^{2(N_p-1)} \;\middle|\; \begin{bmatrix} \boldsymbol{H}_i^h(k)\boldsymbol{G} \\ \boldsymbol{H}_{\mathcal{U}}^h(k) \end{bmatrix} \mathbf{U} \leq \begin{bmatrix} \boldsymbol{\kappa}_i^h(k) \\ \boldsymbol{\theta}_{\mathcal{U}}^h(k) \end{bmatrix} \right\}$$

is not empty.

Finally, given Assumption 3.5 and the associated Remark 3.7, the terminal constraint is built such that the terminal sets $\Omega_i(k)$ is reachable in N_p steps with the constraint $\mathbf{u}(k+l) \in \mathcal{U}$ for all $l \in \overline{0, N_p-1}$.

Thus, the set of constraints of problem (3.12) is never empty, guaranteeing the feasibility of problem (3.12). ■

The feasibility of the decentralized MPC algorithm has been proven. With the proposed control algorithm, the regularity assumptions of Hatleskog (2018) are verified, hence, for a proof of convergence of the overall decentralized algorithm, the reader can refer to Hatleskog (2018).

Remark 3.8: Different dynamics
Theorem 3.1 is valid for the case of agents obeying single integrator dynamics. In Section 3.3, the agents are assumed to obey double integrator dynamics. Then, a discussion on the convergence of the deployment algorithm in this case is drawn in Section 3.4. ◊

Remark 3.9: Feasibility of (3.8)
By following the same steps as in the proof of Theorem 3.1, it can be proven that the centralized problem (3.8) is feasible. In the centralized case, the terminal constraint for optimization problem (3.8) would be chosen such that:

$$\Omega(k) = \Omega_1(k) \times \cdots \times \Omega_N(k)$$

where $\Omega_i(k)$, with $i \in \overline{1, N}$, is defined in (3.16). ◊

3.2.4 Deployment results in the case of single integrator dynamics

Let Σ be a system composed of N agents. The agents are vehicles obeying the single integrator dynamics presented in (3.15). This is a reasonable assumption often made for vehicles such as unmanned ground vehicles (UGV) (Ren and Beard, 2008, Pickem et al., 2017). In the case of UGV, the agent's state vector $\mathbf{x}_i$, with $i \in \overline{1, N}$, is the position of this agent in the plane $\mathbb{R}^2$ denoted by $\mathbf{x}_i(k) = \begin{bmatrix} x_i(k) & y_i(k) \end{bmatrix}^\top$, while the input vector $\mathbf{u}_i$ is the horizontal velocity of the vehicle denoted by $\mathbf{u}_i(k) = \begin{bmatrix} v_{x,i}(k) & v_{y,i}(k) \end{bmatrix}^\top$. The output vector $\mathbf{y}_i$ being identical to the state vector $\mathbf{x}_i$, it is not used in the following.

Let the state space be:

$$\mathcal{X} = \left\{ \mathbf{x} \in \mathbb{R}^2 \;\middle|\; \begin{bmatrix} 3 & 2 \\ 0 & 1 \\ -1 & 2 \\ -1 & -1 \\ 2 & -1 \end{bmatrix} \mathbf{x} \leq \begin{bmatrix} 24 \\ 6 \\ 9 \\ 15 \\ 12 \end{bmatrix} \right\} \tag{3.23}$$

and the input space be:

$$\mathcal{U} = \mathbb{B}^2(\mathbf{1}_{2\times 1}) = \left\{ \mathbf{x} \in \mathbb{R}^2 \;\middle|\; \begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \end{bmatrix} \mathbf{x} \leq \mathbf{1}_{4\times 1} \right\}. \tag{3.24}$$

The sets $\mathcal{X} = \mathcal{Y}$ and $\mathcal{U}$ satisfy Assumptions 3.1 and 3.4.

For a first example, consider $N = 10$ agents in Σ. The agents use the decentralized MPC strategy of Algorithm 3.2. The sampling period used in (3.15) is $T_s = 0.2\,\text{s}$ and the prediction horizon is $N_p = 10$. In this problem, given the size of the workspace $\mathcal{X}$ and the constraints on the control input, the contraction factor λ_i is chosen such that $\lambda_i = 0.9$ for all $i \in \overline{1, N}$. With such a value, an agent located on a vertex of $\mathcal{X}$ can reach $\lambda_i \mathcal{X}$ in less than N_p steps. The weighting matrices are chosen such that $\boldsymbol{Q} = \boldsymbol{R} = \boldsymbol{I}_2$ and $\boldsymbol{P}$ is the solution of the algebraic Riccati equation:

$$\boldsymbol{A}^\top \boldsymbol{P} \boldsymbol{A} - \boldsymbol{P} - \boldsymbol{A}^\top \boldsymbol{P} \boldsymbol{B} \left(\boldsymbol{B}^\top \boldsymbol{P} \boldsymbol{B} + \boldsymbol{R} \right)^{-1} \boldsymbol{B}^\top \boldsymbol{P} \boldsymbol{A} + \boldsymbol{Q} = \boldsymbol{0}_2$$

with $\boldsymbol{A} = \boldsymbol{I}_2$ and $\boldsymbol{B} = T_s \boldsymbol{I}_2$. The solver for the optimization problem (3.12) is generated with CVXGEN (Mattingley and Boyd, 2012, 2013).

The vehicles start from random positions inside the polytopic set $\mathcal{X}$ and start moving towards their Chebyshev centers. The initial configuration is displayed in

Figure 3.1. The agents are represented by circles and the Chebyshev center of their Voronoi cells by stars. The Voronoi cell of each agent is also presented. It can be noticed that in the present example, some agents start from very close positions such as agents 2, 4, 5, 7 and 10. It would not be possible for a real system since each vehicle would have a given size prohibiting them to be too close from each other. However, in the case of the present simulation, punctiform systems are considered. It has to be noted that it is possible to bias the randomness of the starting position by ensuring that two agents cannot be closer than a given distance.

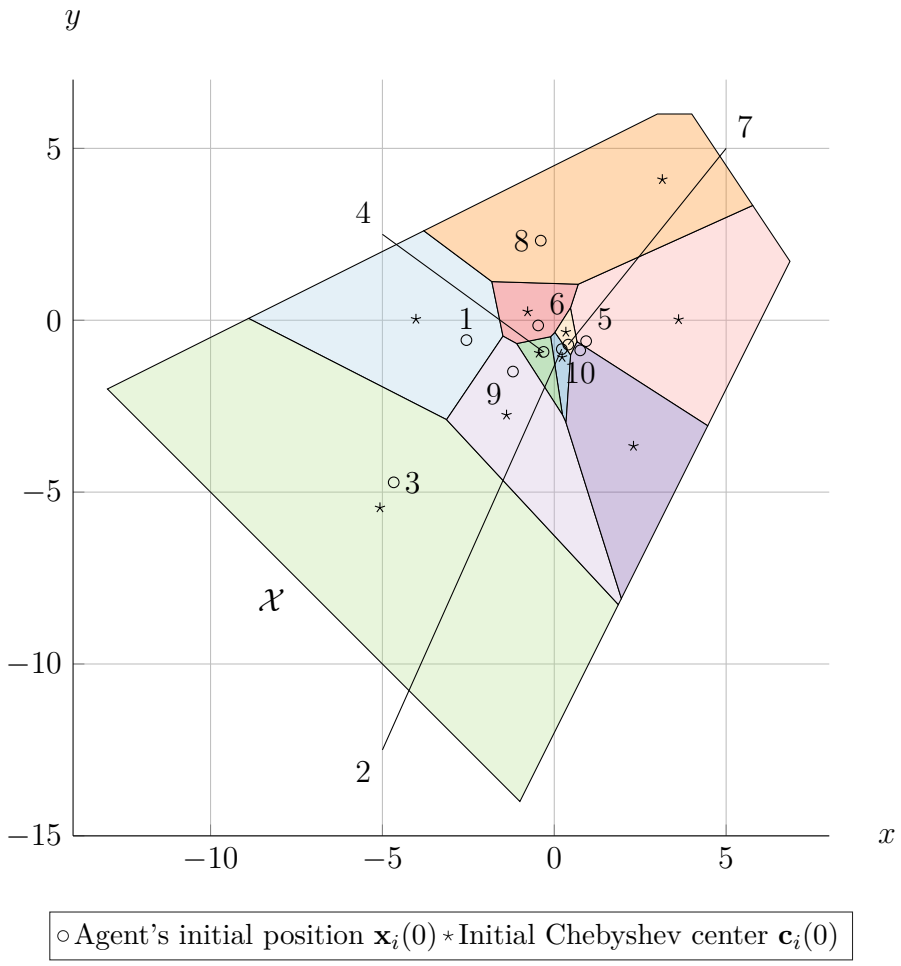
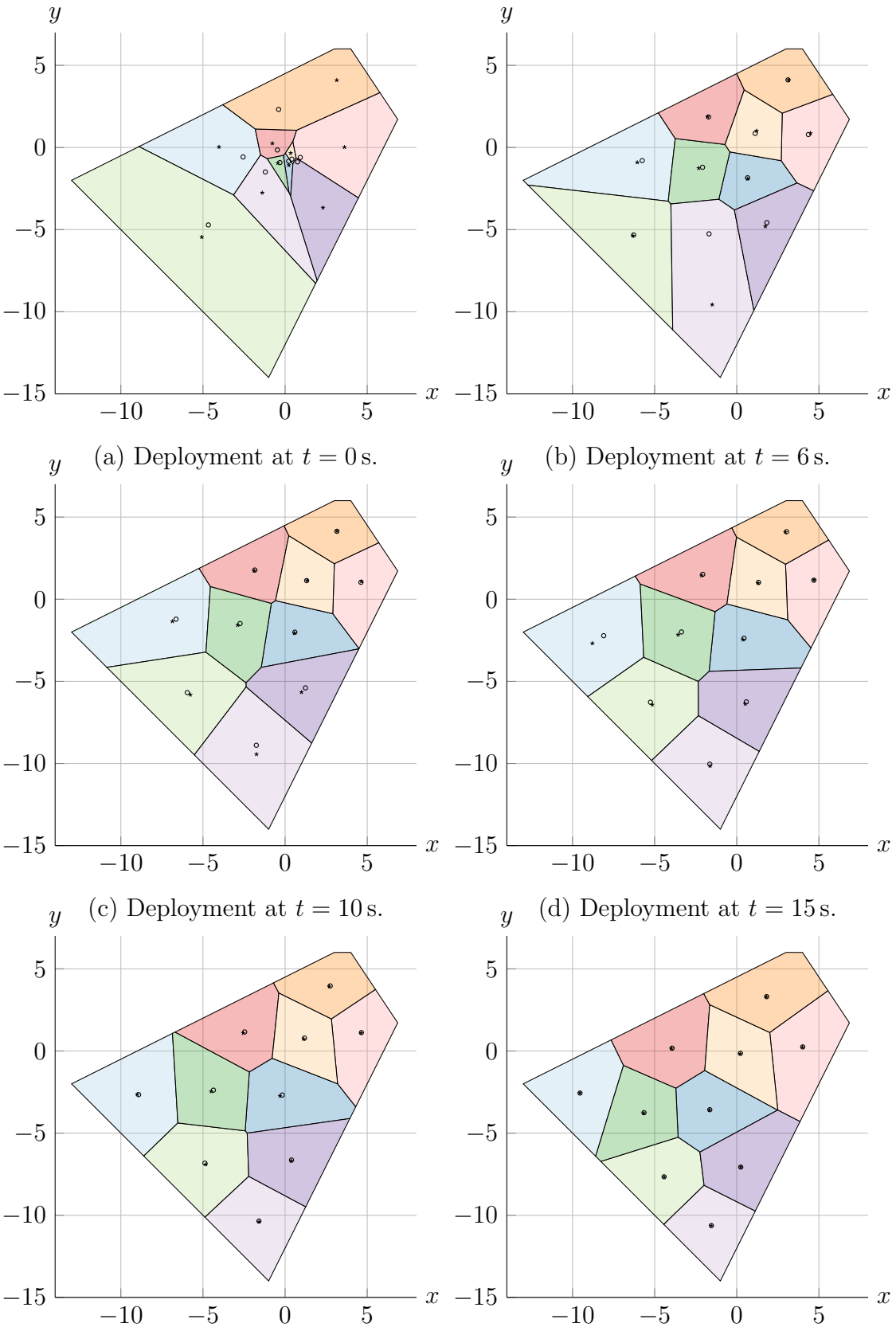


Figure 3.1: Initial position of the agents of Σ in $\mathcal{X}$.

The agents of Σ then evolve inside $\mathcal{X}$ by following their Chebyshev center. Some snapshots of the deployment are presented in Figure 3.2. These snapshots show the deployment of the agents (represented as in Figure 3.1 by circles) and the Chebyshev centers of their Voronoi cells (represented as in Figure 3.1 by stars) as well as the Voronoi tessellation at different time instants.

Figure 3.2 can be analyzed along with Figure 3.3. Indeed, Figure 3.3 presents the complete trajectory of all agents as well as the trajectory of the Chebyshev centers. In this figure, it is possible to see that the Chebyshev center trajectory (dashed line) can undergo abrupt changes such as the trajectories of $\mathbf{c}_3$, $\mathbf{c}_5$, $\mathbf{c}_6$ or $\mathbf{c}_7$. These abrupt changes can be explained by the fact that the shape of a Voronoi cell changes from one time sample to the other due to the global movement of the MAS. For example, agents 1, 2, 4 and 10 have a tendency to move to the left of $\mathcal{X}$, liberating space for



(a) Deployment at $t = 0\,\mathrm{s}$.

(b) Deployment at $t = 6\,\mathrm{s}$.

(c) Deployment at $t = 10\,\mathrm{s}$.

(d) Deployment at $t = 15\,\mathrm{s}$.

(e) Deployment at $t = 20\,\mathrm{s}$.

(f) Deployment at $t = 50\,\mathrm{s}$.

Figure 3.2: Configuration of Σ at different time instants.

agents 5, 6 and 7, making their Chebyshev centers coming back to the left of $\mathcal{X}$ after a first movement to the right. However, these changes do not affect the convergence of the overall algorithm since it is obvious from Figure 3.3 that each agent reaches its Chebyshev center and the MAS reaches a static Chebyshev configuration.

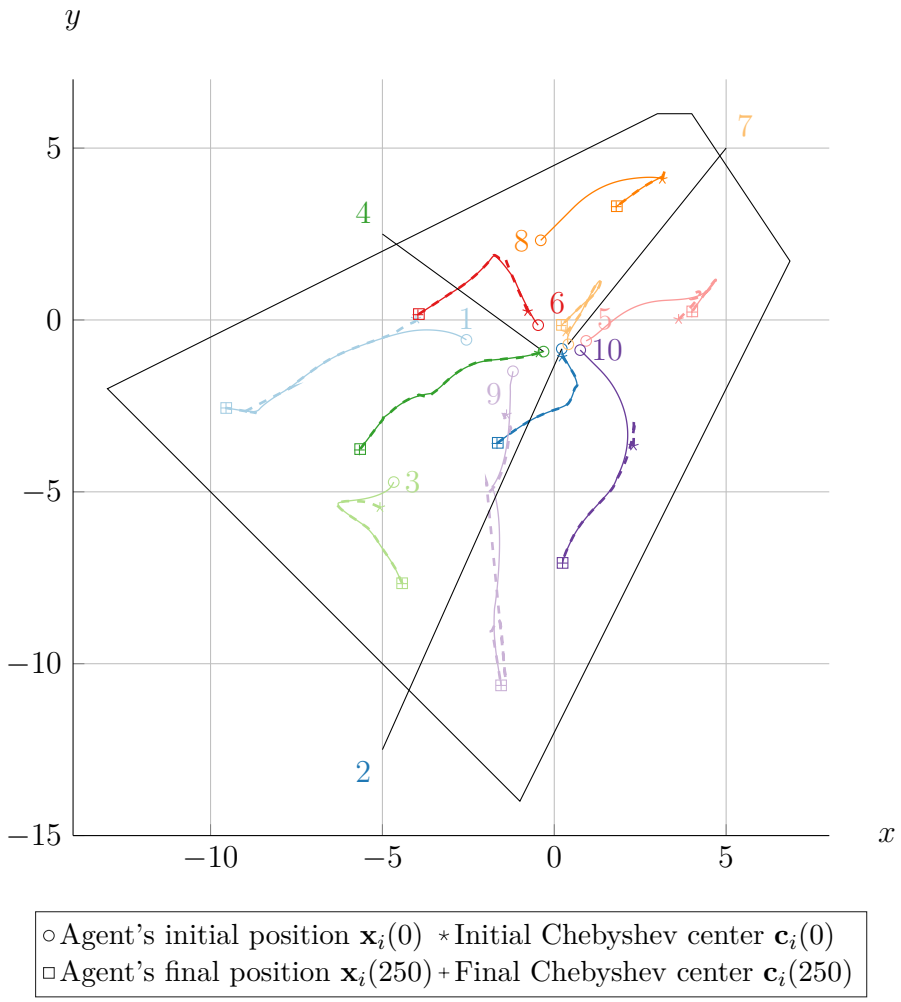


Figure 3.3: Trajectories of the agents of Σ and their associated Chebyshev centers.

This convergence is even more obvious when analyzing Figure 3.4, which shows the distance of each agent to its Chebyshev center over time, denoted by $d_i(k)$, with $i \in \overline{1, N}$. Pikes appear on the distance between agent 1 and 9 and their respective Chebyshev centers. This sudden increase in the distance can be observed in Figure 3.2. Around $t = 14\,\text{s}$, the position of the Chebyshev center of agent 1 greatly changes, as well as the position of the Chebyshev center of agent 9 around $t = 6\,\text{s}$, causing the pikes in Figure 3.4.

Overall, the system reaches the objective described previously, i.e. it deploys to a static Chebyshev configuration as defined in Definition 2.29: each agent lies on the Chebyshev center of its Voronoi cell. As expected, the optimization problem is always feasible with the choice of λ_i, with $i \in \overline{1, N}$, made earlier, knowing that the feasibility of the terminal constraint (3.12f) can be guaranteed by adding λ_i as a decision variable in (3.12) as mentioned in Remark 3.7.

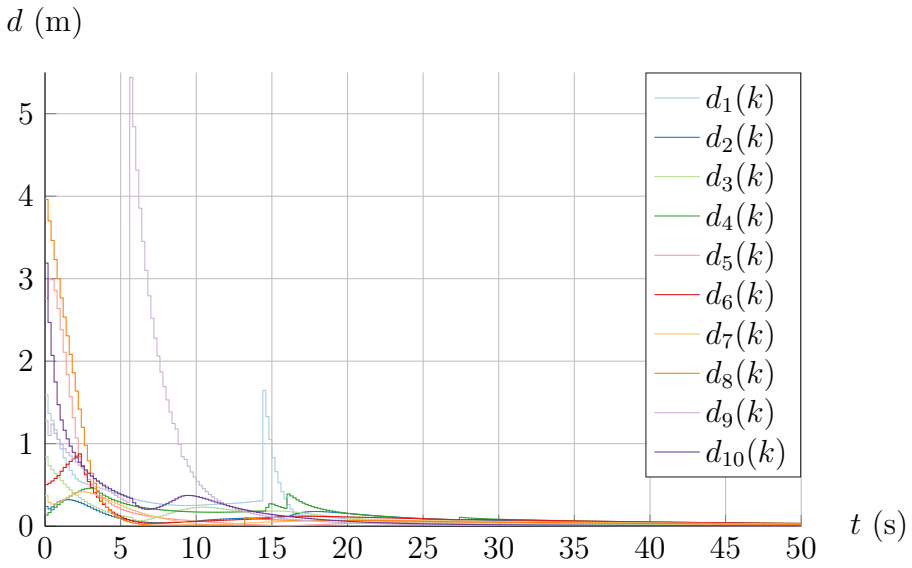


Figure 3.4: Distance of each agent of Σ to its Chebyshev center over time.

3.3 Deployment of a quadrotor UAV fleet

As specified in Section 3.2.3, the assumptions made for the convergence of the problem reduce the possible dynamics of the agents to a very strict class. The aim of this section is to apply the decentralized control algorithm (i.e. Algorithm 3.2) to a fleet of quadrotor unmanned aerial vehicles (UAV). The model of a quadrotor UAV is continuous-time nonlinear with twelve states and not observable with the output matrix described in Remark 2.10. Thus, the model of the UAV has to be linearized, discretized and decomposed into several subsystems. With such a decomposition, the position subsystem of a quadrotor UAV is modeled as double integrator dynamics on which it is possible to apply the decentralized MPC strategy of Algorithm 3.2. Then, the present section shows in simulation that the convergence is also ensured in the case of double integrator dynamics which are also widely used to represent vehicles (Ren and Beard, 2008). The first paragraph of this section, Section 3.3.1, presents the continuous-time nonlinear model of a quadrotor UAV that is used to test the decentralized deployment algorithm Algorithm 3.2. Then, Section 3.3.2 presents the control architecture allowing to consider only double integrator dynamics for the position control of such a vehicle. Finally, Section 3.3.3 shows simulation results of the deployment algorithm applied on a UAV fleet.

3.3.1 Agent model

3.3.1.1 Continuous-time nonlinear dynamics

The dynamics of a quadrotor UAV has been extensively studied. It can be derived from Lagrangian mechanics (Bouabdallah et al., 2004, Castillo et al., 2005, Sabatino, 2015) however, such a derivation is out of the scope of the present thesis.

The continuous-time nonlinear dynamics of a quadrotor UAV can be expressed as a function of the state vector:

$$\mathbf{x} = \begin{bmatrix} x & y & z & \phi & \theta & \psi & v_x & v_y & v_z & \omega_x & \omega_y & \omega_z \end{bmatrix}^\top \tag{3.25}$$

Table 3.1: State variables of a quadrotor UAV.

x, y, z	Cartesian coordinates of the UAV in Earth's frame
ϕ, θ, ψ	Roll, pitch and yaw angles of the UAV
v_x, v_y, v_z	Linear speed of the UAV is Earth's frame
ω_x, ω_y, ω_z	Angular speed of the UAV
f_t	Total upward thrust of the UAV
τ_x, τ_y, τ_z	Torques on the axes of the UAV

and the input vector:

$$\mathbf{u} = \begin{bmatrix} f_t & \tau_x & \tau_y & \tau_z \end{bmatrix}^\top \tag{3.26}$$

such that:

$$\dot{\mathbf{x}} = \boldsymbol{f}(\mathbf{x}, \mathbf{u}). \tag{3.27}$$

The physical meaning of all variables presented in (3.25) and (3.26) are gathered in Table 3.1. In (3.25), (3.26) and (3.27), as well as for the remainder of this paragraph, the time dependence is dropped to simplify the notations.

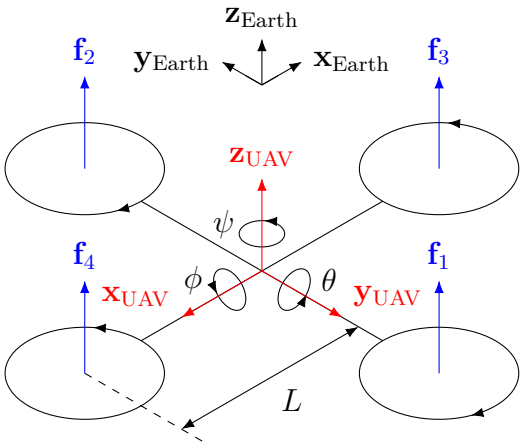


Figure 3.5: Schematic representation of a quadrotor UAV.

Figure 3.5 gathers some elements necessary to understand the model considered in this thesis. The vectors $\mathbf{x}_{\text{UAV}}$, $\mathbf{y}_{\text{UAV}}$ and $\mathbf{z}_{\text{UAV}}$ are attached to the UAV's frame while $\mathbf{x}_{\text{Earth}}$, $\mathbf{y}_{\text{Earth}}$ and $\mathbf{z}_{\text{Earth}}$ are attached to the Earth's frame. The total upward thrust f_t is such that:

$$f_t = \left(\sum_{i=1}^{4} \mathbf{f}_i \right)^\top \mathbf{z}_{\text{UAV}}.$$

The torques τ_x, τ_y and τ_z are such that a positive value of the torque induces a variation of ϕ, θ and ψ respectively in the direction indicated in Figure 3.5. The length L denotes the length of an arm of the quadrotor, from the center of inertia to one rotor's rotation axis. This length is not used here but influences some control strategies (Wang et al., 2016a, Chevet et al., 2020b).

Table 3.2: Values of UAV model's parameters (Chevet et al., 2020b).

Mass	$m = 1.4\,\mathrm{kg}$
Inertia components	$I_x = I_y = 0.03\,\mathrm{kg \cdot m^2}$
	$I_z = 0.04\,\mathrm{kg \cdot m^2}$
Arm's length	$L = 0.2\,\mathrm{m}$
Gravitational acceleration	$g = 9.81\,\mathrm{m \cdot s^{-2}}$

With these elements, the continuous-time nonlinear state space model can be written (Chevet et al., 2020b):

$$\dot{x} = v_x \tag{3.28a}$$
$$\dot{y} = v_y \tag{3.28b}$$
$$\dot{z} = v_z \tag{3.28c}$$
$$\dot{\phi} = \omega_x + (\omega_y \sin\phi + \omega_z \cos\phi)\tan\theta \tag{3.28d}$$
$$\dot{\theta} = \omega_y \cos\phi - \omega_z \sin\phi \tag{3.28e}$$
$$\dot{\psi} = \omega_y \frac{\sin\phi}{\cos\theta} + \omega_z \frac{\cos\phi}{\cos\theta} \tag{3.28f}$$
$$\dot{v}_x = \frac{f_t}{m}(\cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi) \tag{3.28g}$$
$$\dot{v}_y = \frac{f_t}{m}(\cos\phi \sin\theta \sin\psi - \sin\phi \cos\psi) \tag{3.28h}$$
$$\dot{v}_z = \frac{f_t}{m}\cos\phi\cos\theta - g \tag{3.28i}$$
$$\dot{\omega}_x = \frac{I_y - I_z}{I_x}\omega_y\omega_z + \frac{\tau_x}{I_x} \tag{3.28j}$$
$$\dot{\omega}_y = \frac{I_z - I_x}{I_y}\omega_x\omega_z + \frac{\tau_y}{I_y} \tag{3.28k}$$
$$\dot{\omega}_z = \frac{I_x - I_y}{I_z}\omega_x\omega_y + \frac{\tau_z}{I_z} \tag{3.28l}$$

where m is the mass of the quadrotor UAV, I_x, I_y and I_z the moments of inertia around each axis of the UAV and g is the gravitational acceleration. The numerical values of these parameters are gathered in Table 3.2.

3.3.1.2 Discrete-time linear dynamics

The state vectors of the form $\overline{\mathbf{x}} = \begin{bmatrix}\overline{x} & \overline{y} & \overline{z} & \mathbf{0}_{1\times 2} & \overline{\psi} & \mathbf{0}_{6\times 1}\end{bmatrix}^\top$ associated with an input vector of the form $\overline{\mathbf{u}} = \begin{bmatrix} mg & \mathbf{0}_{1\times 3}\end{bmatrix}^\top$ are the only admissible type of equilibrium points for the system described by equations (3.28). The nonlinear system can thus be linearized around an equilibrium point $(\overline{\mathbf{x}}, \overline{\mathbf{u}})$ as:

$$\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{0}_3 & \mathbf{0}_3 & \boldsymbol{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \boldsymbol{I}_3 \\ \mathbf{0}_3 & \boldsymbol{M} & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \end{bmatrix}(\mathbf{x} - \overline{\mathbf{x}}) + \begin{bmatrix} \mathbf{0}_{8\times 1} & \mathbf{0}_{8\times 1} & \mathbf{0}_{8\times 1} & \mathbf{0}_{8\times 1} \\ m^{-1} & 0 & 0 & 0 \\ 0 & I_x^{-1} & 0 & 0 \\ 0 & 0 & I_y^{-1} & 0 \\ 0 & 0 & 0 & I_z^{-1} \end{bmatrix}(\mathbf{u} - \overline{\mathbf{u}}) \tag{3.29}$$

where:

$$\boldsymbol{M} = g \begin{bmatrix} \sin\overline{\psi} & \cos\overline{\psi} & 0 \\ -\cos\overline{\psi} & \sin\overline{\psi} & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Remark 3.10: Notation
In the following, to simplify the notations, $\mathbf{x}$ and $\mathbf{u}$ designate the variations of the state and input of the position subsystem around the equilibrium point. ◊

Since it is quite sparse, the system (3.29) can be separated into a position and an attitude subsystem. However, only one is of interest for the deployment objective that is presented later in this section: the *horizontal position subsystem*. This subsystem is written:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{v}_x \\ \dot{y} \\ \dot{v}_y \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x - \overline{x} \\ v_x \\ y - \overline{y} \\ v_y \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & g\cos\overline{\psi} \\ 0 & 0 \\ -g\cos\overline{\psi} & 0 \end{bmatrix} \begin{bmatrix} \phi \\ \theta \end{bmatrix}. \tag{3.30}$$

To simplify and without loss of generality, it can be assumed that the considered equilibrium point of (3.28) is $\overline{\mathbf{x}} = \begin{bmatrix} \mathbf{0}_{1\times 2} & \overline{z} & \mathbf{0}_{1\times 9} \end{bmatrix}^\top$. With this value for the equilibrium state vector, it is possible to discretize the dynamics (3.30) with the Euler method with the sampling period T_s:

$$\mathbf{x}(k+1) = \begin{bmatrix} x(k+1) \\ v_x(k+1) \\ y(k+1) \\ v_y(k+1) \end{bmatrix} = \begin{bmatrix} 1 & T_s & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T_s \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x(k) \\ v_x(k) \\ y(k) \\ v_y(k) \end{bmatrix} + \frac{1}{2} g T_s \begin{bmatrix} 0 & T_s \\ 0 & 2 \\ -T_s & 0 \\ -2 & 0 \end{bmatrix} \begin{bmatrix} \phi \\ \theta \end{bmatrix}. \tag{3.31}$$

Assumption 3.6: Output of the position subsytem

The output of the horizontal position subsystem (3.31) *is* $\begin{bmatrix} x(k) & y(k) \end{bmatrix}^\top$.

It is not necessary to describe the other subsystems since the control strategy for the variables other than x, y, v_x and v_y is based on feedback linearization as described in Paragraph 3.3.2.2.

3.3.2 Global control strategy

In order to apply the decentralized MPC strategy of Algorithm 3.2 to the position subsystem (3.31), a control architecture for the quadrotor UAV (3.28) has to be designed. Paragraph 3.3.2.1 introduces the overall control architecture for a UAV, while Paragraph 3.3.2.2 and Paragraph 3.3.2.3 present the control strategies for the inner and outer loops that appear in the architecture of Paragraph 3.3.2.1.

3.3.2.1 Overall architecture

Given the model of the position subsystem given in (3.31) and the full state-space model (3.28) of a quadrotor UAV, a cascaded control scheme seems to be a solution for the control of the system. From the models in Section 3.3.1, the outer loop is meant to control the horizontal position of the vehicle, while the inner loop is meant to control the attitude of the vehicle, i.e. the pitch and roll angles ϕ and θ. However,

with such a structure, some variables present in (3.28) do not appear in this control structure, i.e. the altitude z and the yaw angle ψ. However, it is also necessary to control the altitude and yaw angle of the quadrotor UAV. Then, the altitude is controlled with a subsidiary loop, while the control of the yaw angle is integrated to the inner loop described before.

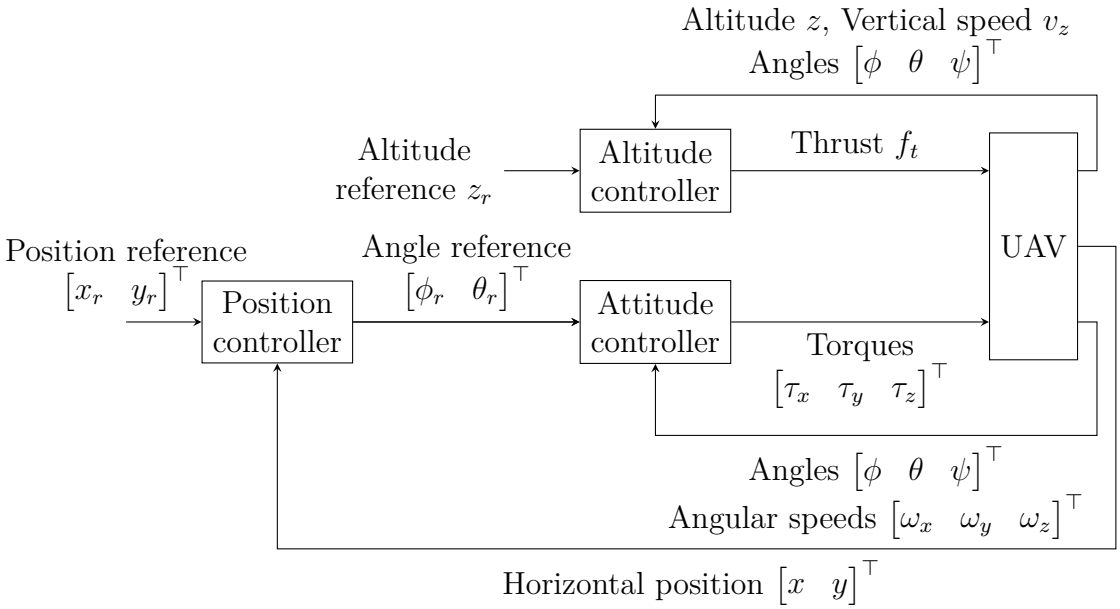


Figure 3.6: Overall control architecture for a quadrotor UAV.

Figure 3.6 provides a schematic view of the overall control structure for a UAV:

- The altitude controller is used for take-off and landing (which is out of the scope of this thesis) and for maintaining altitude of the quadrotor UAV during the flight. It receives measurements of attitude, altitude and vertical speed as well as a constant reference $z_r = \overline{z}$ (the equilibrium value $\overline{z}$ described in Paragraph 3.3.1.2) to compute a value of total thrust provided to the UAV.

- The attitude controller is used to follow the angle reference necessary to move the quadrotor UAV from a given position in the plane to a given objective. It receives measurements of attitude and angular speeds as well as a pitch and roll reference from the position controller (the yaw reference ψ_r being $\psi_r = \overline{\psi} = 0°$ as described in Paragraph 3.3.1.2) to compute torque values that are provided to the UAV.

- The position controller receives a position reference, namely the Chebyshev center of the Voronoi cell associated to the UAV, as well as position measurements from the UAV itself that enable the deployment algorithm to be carried out.

- The total thrust and torques are then converted by the UAV to a rotor speed value for each of the four rotors. Such a conversion is not treated in this thesis.

Both the attitude and altitude controller are described in Paragraph 3.3.2.2. However, this description is only presented for reproducibility of the results described

in Section 3.3.3, a complete study of such controllers being out of the scope of this thesis. The position controller is extensively described in Paragraph 3.3.2.3.

3.3.2.2 Inner-loop control

In this paragraph, the altitude and the attitude controller are briefly described. As mentioned in Paragraph 3.3.1.2, these controllers are obtained via *feedback linearization* (Isidori, 1995) and a full procedure for the overall control of a quadrotor UAV can be found in Voos (2009).

In order to derive the attitude controller, it is first necessary to make an assumption on the angles.

Assumption 3.7: Small angles

The roll, pitch and yaw angles are sufficiently small so that $\dot{\phi} = \omega_x$, $\dot{\theta} = \omega_y$ *and* $\dot{\psi} = \omega_z$. *Such a behavior is obtained when* $|\phi|, |\theta|, |\psi| < 30°$.

Remark 3.11: Limit value of the angles
In the literature (Abdolhosseini et al., 2013), the limit value of $|\phi|, |\theta|, |\psi| < 15°$ can be found. In this case, a linear MPC strategy is used to control the position and attitude hence the need to constrain the angles tightly for the linearized model to be valid over the entire control space. The control structure proposed here being more robust, it is possible to relax the limit value of the angles to the one proposed in Assumption 3.7. ♢

With Assumption 3.7, it is possible to derive the attitude controller. The angular speed dynamics are described in (3.28j)-(3.28l) as:

$$\dot{\omega}_x = \frac{I_y - I_z}{I_x}\omega_y\omega_z + \frac{\tau_x}{I_x} \tag{3.32a}$$

$$\dot{\omega}_y = \frac{I_z - I_x}{I_y}\omega_x\omega_z + \frac{\tau_y}{I_y} \tag{3.32b}$$

$$\dot{\omega}_z = \frac{I_x - I_y}{I_z}\omega_x\omega_y + \frac{\tau_z}{I_z}. \tag{3.32c}$$

By following the procedure of Voos (2009), the control inputs are of the form:

$$\tau_x = f_x(\omega_x, \omega_y, \omega_z) + \tau_x^* \tag{3.33a}$$

$$\tau_y = f_y(\omega_x, \omega_y, \omega_z) + \tau_y^* \tag{3.33b}$$

$$\tau_z = f_z(\omega_x, \omega_y, \omega_z) + \tau_z^* \tag{3.33c}$$

where τ_x^*, τ_y^* and τ_z^* are new input variables. In order to maintain the linearity in ω_x and τ_x^*, it is then necessary that:

$$\begin{aligned}\dot{\omega}_x &= \frac{I_y - I_z}{I_x}\omega_y\omega_z + \frac{1}{I_x}f_x(\omega_x, \omega_y, \omega_z) + \frac{\tau_x^*}{I_x} \\ &= K_x\omega_x + \frac{\tau_x^*}{I_x}\end{aligned}$$

or, by doing the same thing for ω_y and ω_z:

$$\frac{I_y - I_z}{I_x}\omega_y\omega_z + \frac{1}{I_x}f_x(\omega_x, \omega_y, \omega_z) = K_x\omega_x \tag{3.34a}$$
$$\frac{I_z - I_x}{I_y}\omega_x\omega_z + \frac{1}{I_y}f_y(\omega_x, \omega_y, \omega_z) = K_y\omega_y \tag{3.34b}$$
$$\frac{I_x - I_y}{I_z}\omega_x\omega_y + \frac{1}{I_z}f_z(\omega_x, \omega_y, \omega_z) = K_z\omega_z \tag{3.34c}$$

where K_x, K_y and K_z are gains. Obtaining f_x, f_y and f_z from (3.34) and using them in (3.33), the control inputs are:

$$\tau_x = K_x I_x\omega_x - (I_y - I_z)\omega_y\omega_z + \tau_x^* \tag{3.35a}$$
$$\tau_y = K_y I_y\omega_y - (I_z - I_x)\omega_x\omega_z + \tau_y^* \tag{3.35b}$$
$$\tau_z = K_z I_z\omega_z - (I_x - I_y)\omega_x\omega_y + \tau_z^*. \tag{3.35c}$$

Using the inputs defined in (3.35) in (3.32) and Assumption 3.7:

$$\ddot{\phi} = K_x\dot{\phi} + \frac{\tau_x^*}{I_x}$$
$$\ddot{\theta} = K_y\dot{\theta} + \frac{\tau_y^*}{I_y}$$
$$\ddot{\psi} = K_z\dot{\psi} + \frac{\tau_z^*}{I_z}.$$

By choosing $\tau_x^* = K_\phi(\phi_r - \phi)$, $\tau_y^* = K_\theta(\theta_r - \theta)$ and $\tau_z^* = K_\psi(\psi_r - \psi)$, it is then possible to tune the values of K_x, K_ϕ, K_y, K_θ, K_z and K_ψ to obtain the desired behavior for the attitude controller with classical closed-loop second order tuning.

Thus, the attitude controller is such that:

$$\tau_x = K_x I_x\omega_x - (I_y - I_z)\omega_y\omega_z + K_\phi(\phi_r - \phi) \tag{3.36a}$$
$$\tau_y = K_y I_y\omega_y - (I_z - I_x)\omega_x\omega_z + K_\theta(\theta_r - \theta) \tag{3.36b}$$
$$\tau_z = K_z I_z\omega_z - (I_x - I_y)\omega_x\omega_y + K_\psi(\psi_r - \psi) \tag{3.36c}$$

where the values of I_x, I_y and I_z are given in Table 3.2 and $K_x = K_y = K_z = -72\,\mathrm{kg}^{-1}\cdot\mathrm{m}^{-2}$, $K_\phi = K_\theta = 48\,\mathrm{N}\cdot\mathrm{kg}\cdot\mathrm{m}^3$ and $K_\psi = 64\,\mathrm{N}\cdot\mathrm{kg}\cdot\mathrm{m}^3$, assuming that the yaw reference ψ_r is always 0°. With such values, there is no overshoot in the step response of the attitude loop and a 5% response time of 0.1 s.

The same procedure can be applied for the altitude controller. The dynamics of the vertical speed is given by (3.28i):

$$\dot{v_z} = \frac{f_t}{m}\cos\phi\cos\theta - g$$

thus, the input guaranteeing the linearity in f_t^* is:

$$f_t = f_v(v_z, \phi, \theta) + \frac{m}{\cos\phi\cos\theta}f_t^*. \tag{3.37}$$

Injecting (3.37) into (3.28i) gives the expression of f_v guaranteeing linearity in v_z:

$$f_v(v_z, \phi, \theta) = \frac{m}{\cos\phi\cos\theta}(g + K_v v_z)$$

where K_v is a gain. Again, by choosing $f_t^* = K_w(z_r - z)$, where z_r is the altitude reference to be followed, the altitude controller is such that:

$$f_t = \frac{m}{\cos\phi\cos\theta}(g + K_v v_z + K_w(z_r - z)) \tag{3.38}$$

where $K_v = 36\,\mathrm{s}^{-1}$ and $K_w = 400\,\mathrm{s}^{-2}$ to have no overshoot in the step response of the altitude loop and a 5% response time of 0.2 s.

3.3.2.3 Outer-loop control

With the inner loop and the subsidiary control loop of the overall control structure of Figure 3.6, it is now possible to focus on the position controller. This controller follows Algorithm 3.2 for one UAV agent. An additional step of state estimation is considered. Indeed, in Section 3.2.2, the assumptions that the system (3.1) is observable (Assumption 2.2) and that equilibrium points exist for the dynamics (3.1) (Assumption 3.2) are made. It is implicit in Section 3.2.2 that the full state vector is known for the control strategy to work.

Here, the considered system for the control design is the linearized discrete-time position subsystem defined in (3.31). To simplify the discussion, the following notations are valid until the end of Section 3.3:

$$\begin{aligned}
\mathbf{x}_i(k) &= \begin{bmatrix} x_i(k) \\ v_{x,i}(k) \\ y_i(k) \\ v_{y,i}(k) \end{bmatrix} & \boldsymbol{A} &= \begin{bmatrix} 1 & T_s & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T_s \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
\mathbf{u}_i(k) &= \begin{bmatrix} \phi_i(k) \\ \theta_i(k) \end{bmatrix} & \boldsymbol{B} &= \frac{1}{2} g T_s \begin{bmatrix} 0 & T_s \\ 0 & 2 \\ -T_s & 0 \\ -2 & 0 \end{bmatrix} \\
\mathbf{y}_i(k) &= \begin{bmatrix} x_i(k) \\ y_i(k) \end{bmatrix} & \boldsymbol{C} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}
\end{aligned} \tag{3.39}$$

where the index i is appended to the notations introduced in Section 3.3.1 to denote that the variable is related to the UAV agent i of Σ, where $i \in \overline{1, N}$, with $N = |\Sigma|$. The assumptions of controllability (Assumption 2.1) and observability of the system (Assumption 2.2) as well as the existence of equilibrium points (Assumption 3.2) are verified by the dynamics:

$$\begin{aligned}
\mathbf{x}_i(k+1) &= \boldsymbol{A}\mathbf{x}_i(k) + \boldsymbol{B}\mathbf{u}_i(k) \\
\mathbf{y}_i(k) &= \boldsymbol{C}\mathbf{x}_i(k)
\end{aligned} \tag{3.40}$$

with the matrices $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ defined in (3.39). However, it is also easy to verify that the Voronoi cells are not controlled λ-contractive in this case, thus Assumption 3.3 does not hold. This is overlooked for the time being and a discussion on this matter is lead in Section 3.4.

Assumption 3.8: Availability of measurements

Only the position measurements $\mathbf{y}_i(k)$ coming from the continuous-time nonlinear UAV are available to the position controller.

Due to Assumption 3.8, it is necessary to introduce an observer to retrieve the full state vector $\mathbf{x}_i(k)$ from the value of $\mathbf{y}_i(k)$, with $i \in \overline{1, N}$. Since the system is nominal and with quite simple dynamics, a Luenberger observer (Luenberger, 1964) is adequate. This observer is designed such that:

$$\widehat{\mathbf{x}}_i(k+1) = \boldsymbol{A}\widehat{\mathbf{x}}_i(k) + \boldsymbol{B}\mathbf{u}_i(k) + \boldsymbol{L}(\mathbf{y}_i(k) - \widehat{\mathbf{y}}_i(k)) \tag{3.41}$$

where $\mathbf{y}_i$, with $i \in \overline{1, N}$, is the measured output from the UAV agent, $\widehat{\mathbf{x}}_i$ is the state obtained from the observer, $\widehat{\mathbf{y}}_i = \boldsymbol{C}\widehat{\mathbf{x}}_i$ is the estimated output and $\boldsymbol{L} \in \mathbb{R}^{4\times 2}$ is the gain of the Luenberger observer.

The structure of the position controller for an agent $i \in \overline{1, N}$ is then presented in Figure 3.7. Algorithm 3.2 runs the MPC optimization problem (3.12) with the difference that the observed state $\widehat{\mathbf{x}}_i(k)$ is used instead of the state $\mathbf{x}_i(k)$ to compute the input signal $\mathbf{u}_i(k)$. This input signal then passes through a zero order hold (ZOH) running at the sampling rate of the outer loop T_s to send the reference to the inner-loop controller. The measurement $\mathbf{y}_i(k)$ comes back from the UAV agent, is processed by the Luenberger observer (3.41) and the observed state is sent to the MPC algorithm.

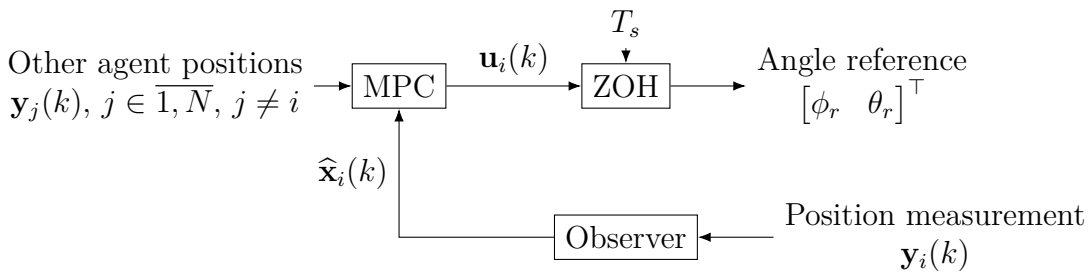


Figure 3.7: Structure of the position controller for a quadrotor UAV.

Another issue that has to be addressed here is the terminal constraint (3.12f). It has been mentioned that Assumption 3.3 does not hold anymore with double integrator dynamics such as (3.40) with the matrices $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ given in (3.39). However, the terminal constraint can be built on the same concept of λ-contractive controlled invariant set described in Definition 3.1. In his thesis, Nguyen (2016) introduces a concept that is interesting for a constraint such as the one used in Assumption 3.5, the N-step controlled λ-contractiveness.

Definition 3.2: N-step controlled λ-contractiveness (Nguyen, 2016)

A convex set $\mathcal{Y} \in \mathbb{R}^p$ is said to be N-step controlled λ-contractive, with $\lambda \in [0, 1)$, for the dynamics (3.1) if for any $\mathbf{y}_0 \in \mathcal{Y}$ and any $\mathbf{x}(k) \in \mathcal{X} \subset \mathbb{R}^n$ such that $\boldsymbol{C}\mathbf{x}(k) = \mathbf{y}(k) \in \mathcal{Y}$, there exists a control sequence $\{\mathbf{u}(k), \ldots, \mathbf{u}(k+N-1)\}$ such that $\mathbf{y}(k+N) \in \{\mathbf{y}_0\} \oplus \lambda(\mathcal{Y} \oplus \{-\mathbf{y}_0\})$.

With the concept exposed in Definition 3.2, a useful result arises.

Theorem 3.2: Controlled λ-contractiveness of a polytope (Nguyen, 2016)

Let a system following the dynamics (3.1) satisfy Assumptions 2.1, 2.2 and 3.2 with $\mathcal{X}$ bounded. Then, for all $\lambda \in [0, 1)$, there exists a finite integer $N(\lambda)$ such that any

convex set $\mathcal{Y} \subset \mathbb{R}^p$ is $N(\lambda)$-step controlled λ-contractive.

Then, with Theorem 3.2, it is always possible, taking a long enough prediction horizon N_p, to consider a terminal constraint of the form given in Assumption 3.5. Thus, the terminal constraint (3.12f) is modified to be such that:

$$\boldsymbol{C}\mathbf{x}_i(k+N_p) \in \{\mathbf{c}_i(k)\} \oplus \lambda_i(k)(\mathcal{V}_i(k) \oplus \{-\mathbf{c}_i(k)\}). \tag{3.42}$$

3.3.3 Deployment results

Let Σ be a system composed of N agents. These agents are UAVs obeying the dynamics (3.28). Each agent uses the control structure presented in Section 3.3.2 to move inside a polytopic set. Let the output space be the same as in Section 3.2.4:

$$\mathcal{Y} = \left\{ \mathbf{x} \in \mathbb{R}^2 \;\middle|\; \begin{bmatrix} 3 & 2 \\ 0 & 1 \\ -1 & 2 \\ -1 & -1 \\ 2 & -1 \end{bmatrix} \mathbf{x} \leq \begin{bmatrix} 24 \\ 6 \\ 9 \\ 15 \\ 12 \end{bmatrix} \right\} \tag{3.43}$$

and the input space for the MPC algorithm be, according to Assumption 3.7 and Assumption 3.4:

$$\mathcal{U} = \mathbb{B}^2\left(\frac{\pi}{6} \cdot \mathbf{1}_{2\times 1}\right) = \left\{ \mathbf{x} \in \mathbb{R}^2 \;\middle|\; \begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \end{bmatrix} \mathbf{x} \leq \frac{\pi}{6} \cdot \mathbf{1}_{4\times 1} \right\}. \tag{3.44}$$

Since the MPC algorithm concerns the position subsystem (3.40) with the matrices from (3.39), it is also necessary to give a constraint for the other state variables, i.e. $v_{x,i}$ and $v_{y,i}$, with $i \in \overline{1,N}$, that are controlled via the MPC. Thus, the state space is:

$$\mathcal{X} = \left\{ \mathbf{x} \in \mathbb{R}^4 \;\middle|\; \boldsymbol{C}\mathbf{x} \in \mathcal{Y} \text{ and } \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x} \leq 2 \cdot \mathbf{1}_{4\times 1} \right\} \tag{3.45}$$

which constrains the speed to be such that $|v_{x,i}|, |v_{y,i}| \leq 2\,\mathrm{m}\cdot\mathrm{s}^{-1}$, with $i \in \overline{1,N}$.

The example presented here follows the same idea as the one presented in Section 3.2.4. Consider $N = 10$ agents in Σ and the prediction horizon $N_p = 10$ steps for all agents with a contraction factor for the terminal constraint $\lambda_i = 0.9$, with $i \in \overline{1,N}$. The sampling rate used in (3.31) is $T_s = 0.2\,\mathrm{s}$. The weighting matrices are chosen such that $\boldsymbol{Q} = \boldsymbol{I}_4$, $\boldsymbol{R} = \boldsymbol{I}_2$ and $\boldsymbol{P}$ is the solution of the algebraic Riccati equation:

$$\boldsymbol{A}^\top\boldsymbol{P}\boldsymbol{A} - \boldsymbol{P} - \boldsymbol{A}^\top\boldsymbol{P}\boldsymbol{B}\left(\boldsymbol{B}^\top\boldsymbol{P}\boldsymbol{B} + \boldsymbol{R}\right)^{-1}\boldsymbol{B}^\top\boldsymbol{P}\boldsymbol{A} + \boldsymbol{Q} = \boldsymbol{0}_4$$

with $\boldsymbol{A}$ and $\boldsymbol{B}$ defined in (3.39). The observer gain is obtained as a linear quadratic regulator with weighting matrices $\boldsymbol{Q}_{\text{obs}} = 10\boldsymbol{I}_4$ and $\boldsymbol{R}_{\text{obs}} = \boldsymbol{I}_2$, giving:

$$\boldsymbol{L} = \begin{bmatrix} 1.0971 & 0 \\ 0.8303 & 0 \\ 0 & 1.0971 \\ 0 & 0.8303 \end{bmatrix}.$$

Finally, the solver for the optimization problem (3.12) is generated with CVXGEN (Mattingley and Boyd, 2012, 2013).

The UAVs start from random positions inside the polytopic set $\mathcal{Y}$. The initial value of the other variables is 0 except the altitude $z_i(0) = z_r = 5\,\text{m}$, with $i \in \overline{1, N}$. Each vehicle then starts moving towards its Chebyshev center. The behavior is globally the same as the one obtained for single integrator dynamics in Section 3.2.4. In each of the figures referenced until the end of this paragraph, the position of the UAV agents is the "real" position $\mathbf{y}_i$, with $i \in \overline{1, N}$, the output of the continuous-time nonlinear dynamics (3.28), and not the observed position $\widehat{\mathbf{y}}_i$.

The initial configuration of Σ in $\mathcal{Y}$ is displayed in Figure 3.8. The UAV agents are represented by circles and the Chebyshev center of their associated cell by stars. The initial Voronoi cell of each agent is also presented. Contrary to the example in Section 3.2.4, the UAVs are placed sufficiently far from one another so that they are not superposed (since their diameter is of $2L = 40\,\text{cm}$ according to Table 3.2).

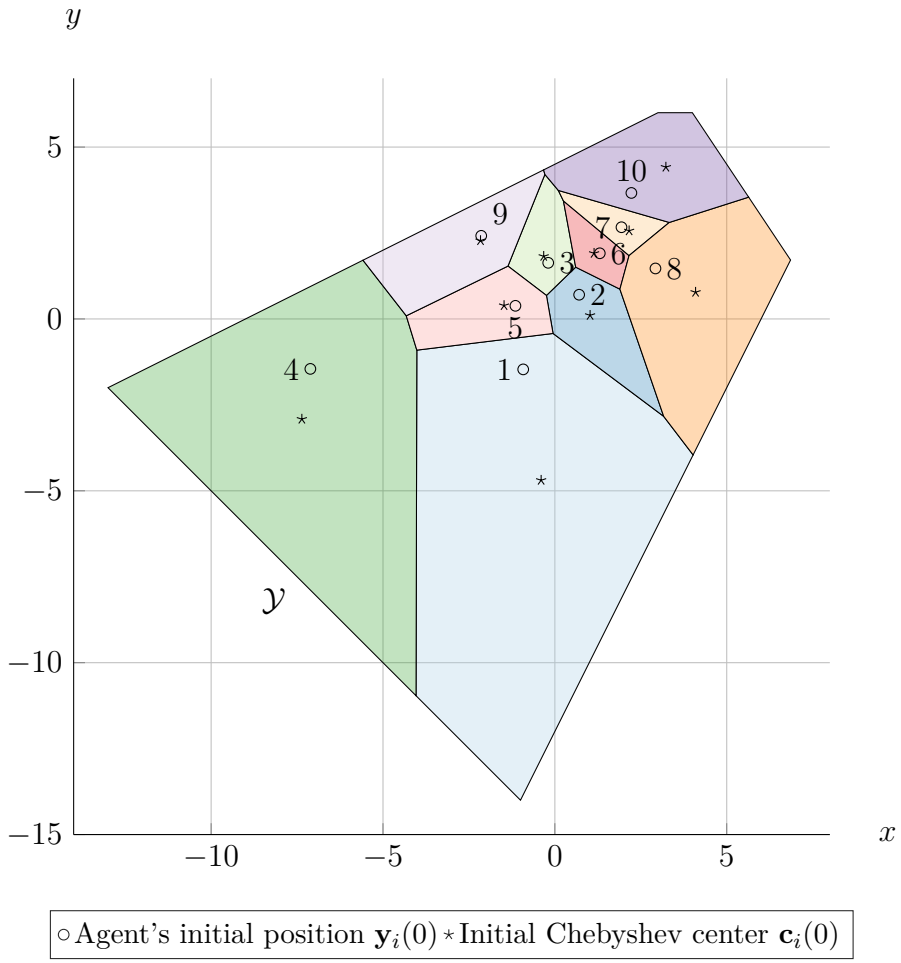
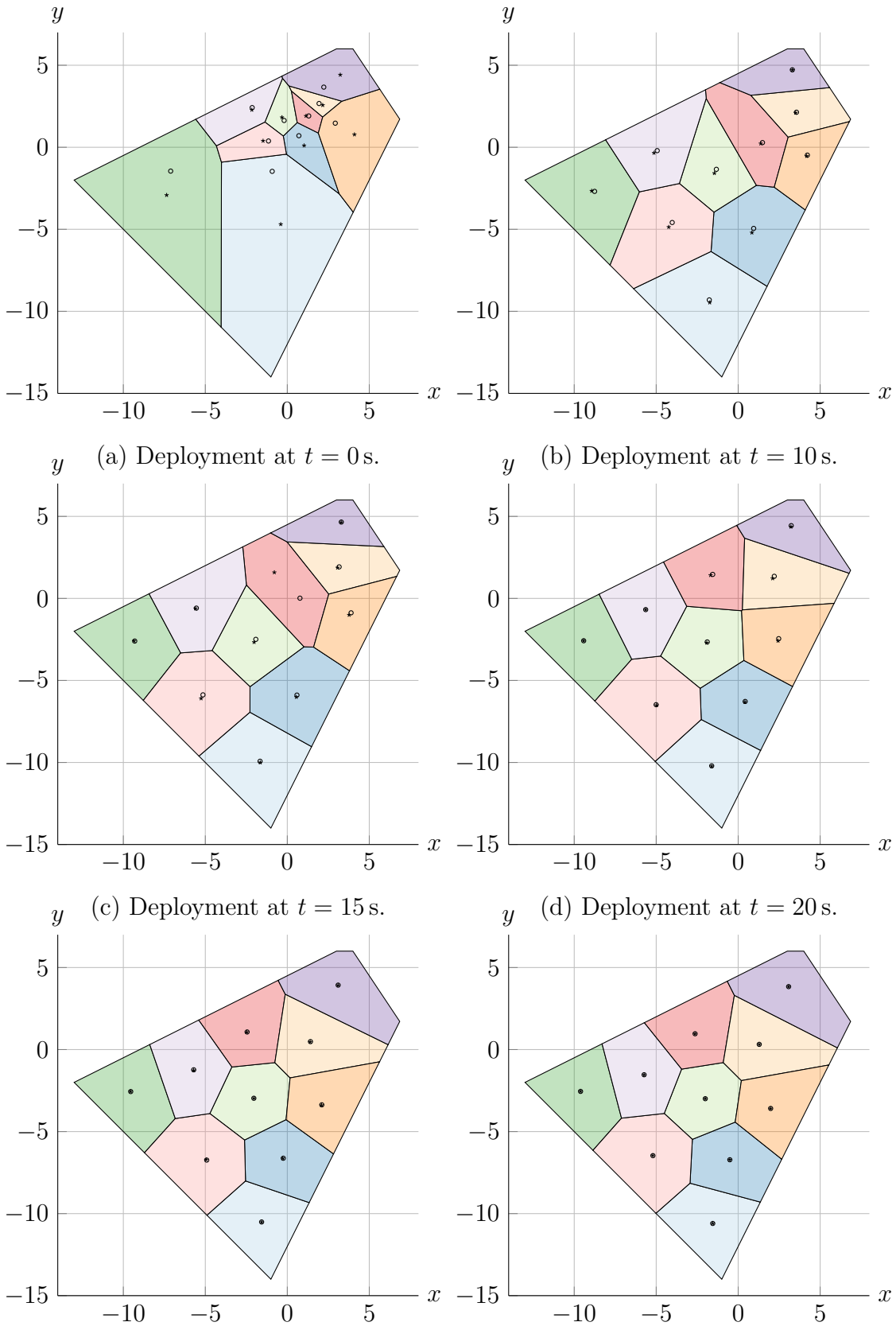


Figure 3.8: Initial position of the agents of Σ in $\mathcal{Y}$.

The agents of Σ follow their Chebyshev centers inside $\mathcal{Y}$. Some snapshots of the deployment are presented in Figure 3.9. These snapshots show the position of the UAV agents, represented by circles, and the Chebyshev centers they are tracking, represented by stars, inside their Voronoi cells.

By examining Figure 3.9 along with Figure 3.10, it is immediate that the behavior of the MAS is close, if not identical, to the behavior observed for single integrator



(a) Deployment at $t = 0\,\mathrm{s}$.

(b) Deployment at $t = 10\,\mathrm{s}$.

(c) Deployment at $t = 15\,\mathrm{s}$.

(d) Deployment at $t = 20\,\mathrm{s}$.

(e) Deployment at $t = 25\,\mathrm{s}$.

(f) Deployment at $t = 50\,\mathrm{s}$.

Figure 3.9: Configuration of Σ at different time instants.

dynamics in Section 3.2.4. As in the last case, the Chebyshev center position can undergo an abrupt change such as what can be seen between Figure 3.9(b) and Figure 3.9(c) where the position of $\mathbf{c}_6$ (in the red Voronoi cell) changes greatly due to the overall movement of the MAS. As in the single integrator case, such changes do not affect the convergence of the algorithm as can be seen in Figure 3.10.

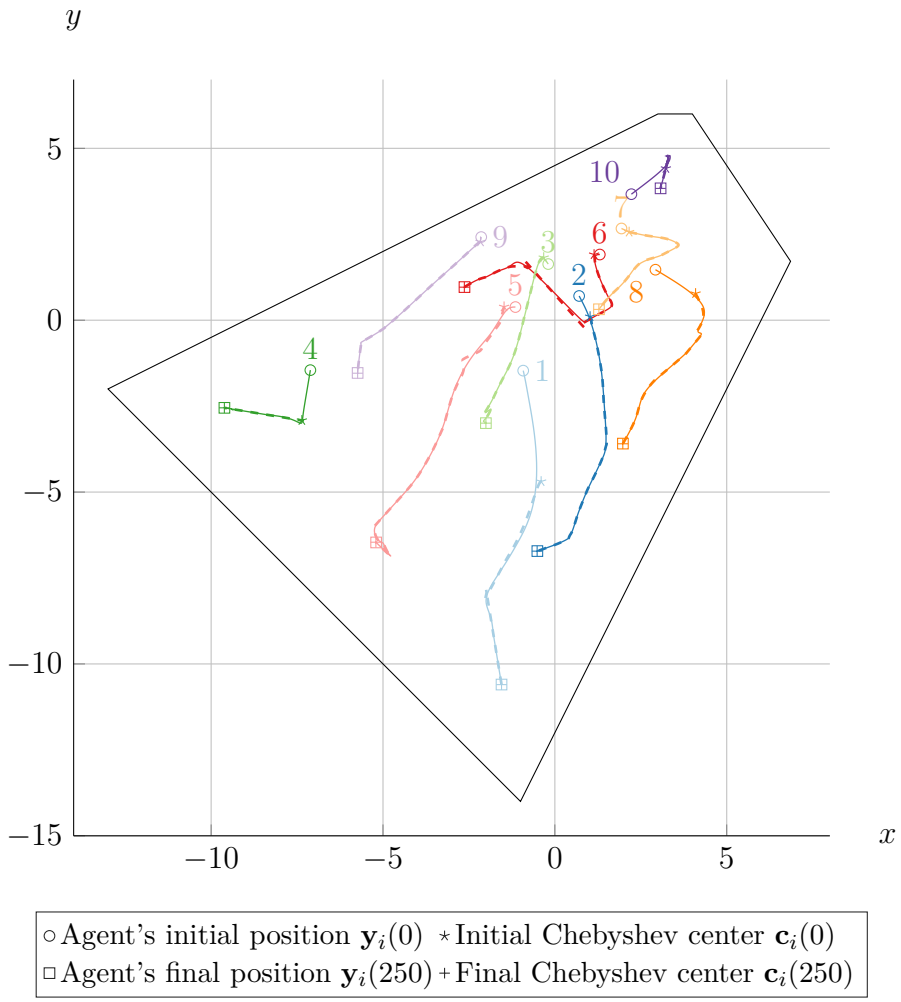


Figure 3.10: Trajectories of the agents of Σ and their associated Chebyshev centers.

As well as for the single integrator case, a measure of the convergence of the system to a static Chebyshev configuration is the distance from the agents' position to their Chebyshev center over time. This distance is displayed in Figure 3.11 for each agent. All distances are calculated as:

$$d_i(k) = \|\mathbf{y}_i(k) - \mathbf{c}_i(k)\|_2$$

for all $i \in \overline{1, N}$. The distances converge to 0 with some occasional pikes due to the sudden change of the Chebyshev center position.

Finally, one of the differences with the single integrator case is the presence of an observer in the control loop. Even though the "real" value of the $\mathbf{y}_i$ was used for the previous plots, it would be interesting to see the difference between the measured position $\mathbf{y}_i$ and the observed position $\widehat{\mathbf{y}}_i$ over time. In Figure 3.12 are presented the value of the norm of the aforementioned difference:

$$\varepsilon_i(k) = \|\mathbf{y}_i(k) - \widehat{\mathbf{y}}_i(k)\|_2$$

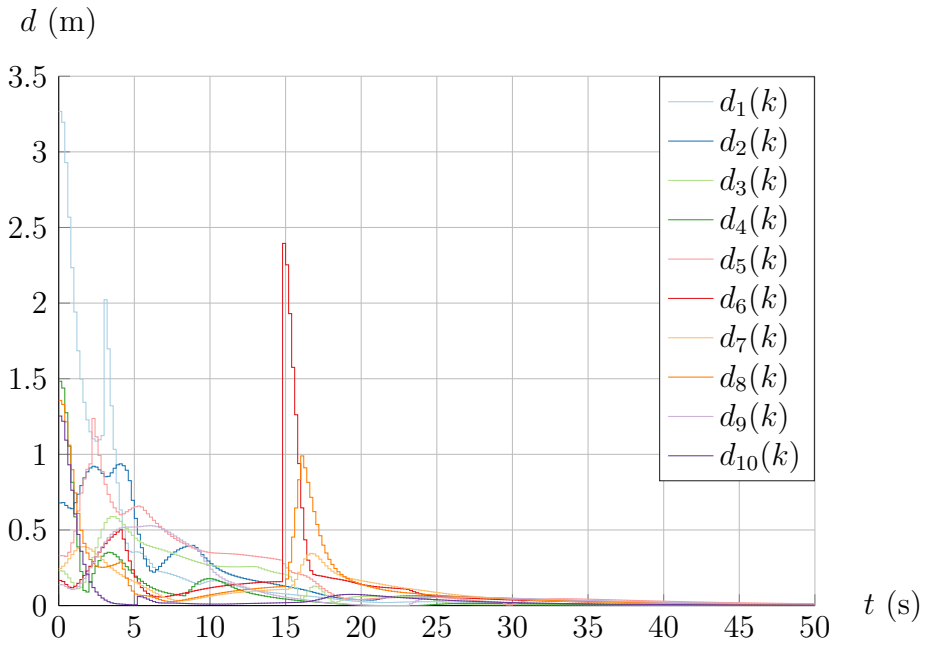


Figure 3.11: Distance of each agent of Σ to its Chebyshev center over time.

for all $i \in \overline{1, N}$ over time. Given that for all agents, ε_i is always less than 0.04 m, it is then reasonable to use the proposed observer in the control loop since the deviation between the measured output and the observed output is negligible.

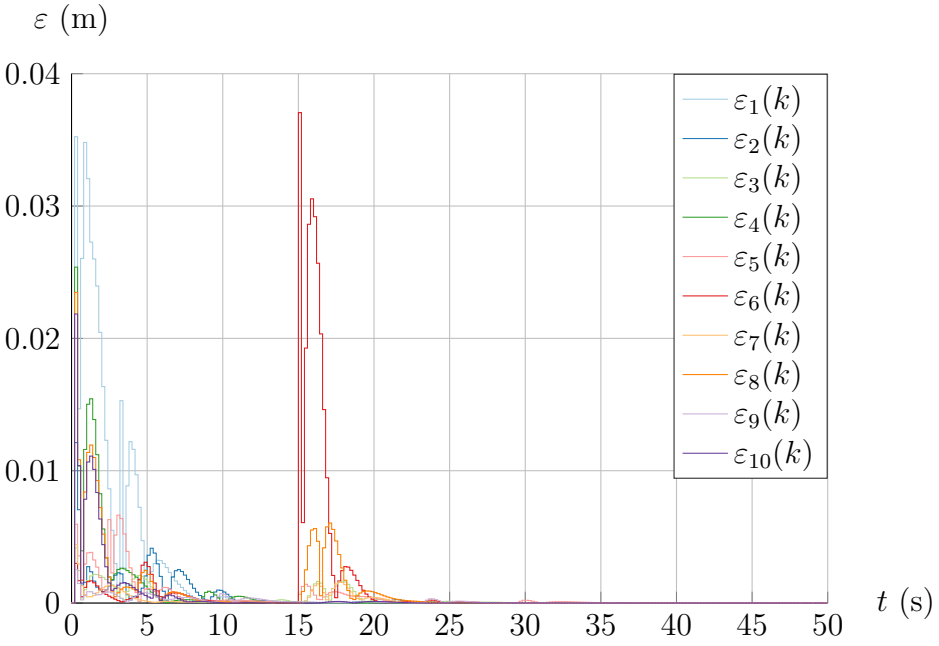


Figure 3.12: Norm of the error between the measured position and the observed position of all agents of Σ.

3.4 Discussion on the stability of the deployment of UAVs

In Section 3.2, a control algorithm based on MPC has been introduced as well as a proof of feasibility of the problem for agents with single integrator dynamics. The convergence proof is not treated since the proposed controller for agents with single integrator dynamics (classical dynamics when dealing with autonomous vehicles) satisfies the assumptions made by Hatleskog (2018) to show convergence of the decentralized deployment problem proposed in Section 3.2. Thus, combining the feasibility result of Theorem 3.1 with the proof of Hatleskog (2018), it is immediate that the proposed algorithm converges.

However, Section 3.3 introduces the idea of using the decentralized algorithm presented in Section 3.2.2 for a different type of dynamics, i.e. double integrator dynamics. While still quite simple and classical in autonomous vehicle modeling, one of the essential assumption of Hatleskog (2018) is not satisfied anymore. Indeed, Assumption 3.3 leads to the fact that the *depth*, i.e. the minimal distance of the position of an agent $\mathbf{y}_i(k)$ to the border of its Voronoi cell $\partial\mathcal{V}_i(k)$, increases with the applied control input, or that $\min(d(\mathbf{y}_i(k), \partial\mathcal{V}_i(k))) < \min(d(\mathbf{y}_i(k+1), \partial\mathcal{V}_i(k)))$. This assumption is essential for the convergence proof in Hatleskog (2018) to hold.

With double integrator dynamics, Assumption 3.3 is not guaranteed anymore. Thus, the convergence is not *a priori* guaranteed. Meanwhile, simulations show that the system still converges towards a static Chebyshev configuration. While only one simulation has been shown in Section 3.3.3, Monte-Carlo simulations with random starting points and random convex shape for the output space $\mathcal{Y}$ still lead to convergence. Thus, it seems that Assumption 3.3 can be relaxed, potentially under some conditions. However, the proof of this statement is difficult to formulate. Such a proof is twofold: first, the fact that the decentralized MPC controller is stable has to be exhibited before dealing with the convergence to a static Chebyshev configuration of the MAS, where each agent obeys double integrator dynamics.

A fundamental ingredient for the stability of MPC is recursive feasibility (Löfberg, 2012), i.e. the fact that finding a feasible solution at time k for the MPC optimization problem implies that a feasible solution exists at time $k+1$ for this same optimization problem. In the literature (Limón et al., 2018, Köhler et al., 2018), proofs of recursive feasibility often (if not always) assume the existence of a feasible sequence of input signals to the MPC optimization problem at time k, e.g. $\{\mathbf{u}(k), \dots, \mathbf{u}(k+N_p-1)\}$, and show that this same sequence shifted by one element, e.g. $\{\mathbf{u}(k+1), \dots, \mathbf{u}(k+N_p-1), \mathbf{u}(k+N_p)\}$, is still feasible at time $k+1$. The last element $\mathbf{u}(k+N_p)$ of the shifted sequence is constructed from the fact that the state of the system is supposed to be inside an invariant set at time $k+N_p-1$, leading to the existence of the element $\mathbf{u}(k+N_p)$.

The deployment problem studied in this chapter is closely related to another kind of control application that can be found in the literature: the *MPC tracking problem* (Dughman and Rossiter, 2015). Indeed, the objective for each agent is to track a time-varying objective in order to finally reach a static and stable configuration. The main complication that arises in such a problem, causing recursive feasibility issues, is the time-varying terminal constraint (3.12f) of the MPC optimization problem (3.12). Indeed, the objective appearing in the cost function (3.12a) has to belong to the set that is used as a terminal constraint. Often, the choice of a

terminal constraint centered on the objective point is made. With the deployment problem, as seen in the simulations of Section 3.2.4 or Section 3.3.3, this objective point can change abruptly from one time step to the other, these changes being impossible to predict with the strategy proposed here. Such a behavior can lead to recursive feasibility issues for the MPC as explained in Dughman and Rossiter (2015). Several strategies exist to deal with time-varying terminal constraints. For example, Limón et al. (2008) and Limón et al. (2018) introduce a new decision variable in the optimization problem used for the MPC that is used as a *virtual reference* that guarantees recursive feasibility of the controller. Another example can be found in Simon et al. (2014), which resembles the algorithm proposed in Section 3.2. Indeed, it uses a scaled and translated version of an invariant set for the dynamics of the considered system (here, the UAV agents), where the scaling factor is a decision variable as well as the translation vector which is close to the concept of *virtual reference* introduced by Limón et al. (2008).

Another element that can make the proof of recursive feasibility difficult is the presence of the time-varying constraints (3.12e) on the output of an agent. Indeed, since the constraint on the output of each agent changes at each time step, it is *a priori* difficult to guarantee that a sequence feasible at time k is still feasible at time $k+1$. In the literature, in order to deal with time-varying constraints, it is often assumed that the evolution of the sets involved in such constraints is known *a priori* as in Köhler et al. (2018). With the problem at hand, constraint (3.12e) would then be replaced by $\mathbf{y}_i(k+l) \in \mathcal{V}_i(k+l)$, with $i \in \overline{1, N}$, for all $l \in \overline{0, N_p - 1}$. The initial constraints $\mathcal{V}_i(l)$ for all $l \in \overline{0, N_p - 1}$ would be obtained by finding an initial trajectory for all the agents and constructing all the Voronoi cells along the prediction horizon. The constraints would then be updated by shifting the trajectory of Voronoi cells by one element and constructing the last one with the knowledge of the last element of the trajectory of each agent. With such constraints, the optimization problem would be recursively feasible if used in combination with the method from Limón et al. (2018) or Simon et al. (2014) which allows to deal with the terminal constraint. However, such a strategy would only work in a centralized or distributed framework, the latter requiring too many communications between the agents. The main advantage of the control algorithm proposed in Section 3.2 is that it is based only on the current position of the other agents to compute the control input of a given agent, thus limiting the need for communications and the complexity of the overall problem.

A proof of stability of the decentralized control algorithm Algorithm 3.2 could start with a strategy similar to the one proposed in Köhler et al. (2018) combined with the *virtual reference* of Limón et al. (2018) or the terminal sets of Simon et al. (2014). The constraints on the output would then be modified to what was described above by computing a so-called trajectory of Voronoi cells for each agent. Then, a proof of stability of the centralized algorithm of Section 3.2.1 would be formulated. It should then be adapted for the decentralized approach of Section 3.2.2 with modified output constraints. The last step of the proof would be to prove that a suboptimal yet stable controller is obtained by restricting the problem introduced above to the one presented in Section 3.2.2. Another possibility would be to find a condition under which the time-varying constraints present in the proposed problem are recursively feasible. Work on recursive feasibility of time-varying constraints has been done by Liu and Stursberg (2019), but this is limited to polyhedral constraints of the form $\boldsymbol{H}\mathbf{x} \leq \boldsymbol{\theta}$, where only $\boldsymbol{\theta}$ is time-varying. In the problem considered in this chapter,

both $\boldsymbol{H}$ and $\boldsymbol{\theta}$ are time-varying, which complicates the problem.

Once the stability of the decentralized MPC algorithm is proven, the convergence of the deployment algorithm is studied. The proof of convergence would then be adapted from the proof of Hatleskog (2018).

However, it can be verified with Monte-Carlo simulations that any initial configuration in any convex set (wide enough for the agents to fit and move inside) converges towards a static Chebyshev configuration when the agents obey quadrotor UAV dynamics. It then must be possible to relax Assumption 3.3 and still obtain convergence of the deployment algorithm.

3.5 Conclusion

This chapter presents a novel model predictive control algorithm to perform a Voronoi-based deployment of a multi-agent system where each agent is tracking the Chebyshev center of the Voronoi cell it belongs to. This algorithm, derived from the one introduced in Nguyen (2016), is presented in both centralized and decentralized forms. A proof of feasibility is proposed in the case of single integrator dynamics which, combined with the proof of convergence of Hatleskog (2018), proves convergence of the deployment algorithm in the case of single integrator dynamics. Simulations then show an example of deployment of a MAS composed of vehicles having single integrator dynamics. Using the decentralized deployment algorithm, the MAS converges to a static Chebyshev configuration.

In a second part, the decentralized deployment algorithm is applied to a fleet of UAVs. The position dynamics used for the MPC are reduced to double integrator and the other dynamics are controlled with a controller obtained from feedback linearization. Simulations performed using nonlinear models for all agents show that the system deploys and converges to a static Chebyshev configuration. The chapter ends on a discussion about said convergence. Indeed, one essential assumption for the proof of converge given in Hatleskog (2018) is not satisfied in the case of dynamics more complex than single integrator. Moreover, time-varying constraints appearing in the MPC algorithm are another drawback. Future work on such a deployment algorithm should tackle the problem of proving convergence for a wider variety of dynamics than single integrator dynamics.

While Chapter 3 deals with a Voronoi-based deployment of a multi-agent system in the nominal case, Chapter 4 considers perturbation and measurement noise on the agent dynamics. Then, it presents robust algorithms to allow the agents to track the Chebyshev center of their Voronoi cell when they are subject to bounded deterministic perturbations as well as when they are subject to unbounded stochastic perturbations.

Chapter 4

Deployment of a multi-vehicle system subject to perturbations

Table of Contents

The deployment problem for a multi-vehicle system (MVS) has been introduced in Chapter 3. However, it has only been approached in the case where the vehicle dynamics are nominal, i.e. their dynamics are of the form (2.40) and the vehicles are not subject to any form of perturbation. This situation is not realistic since this kind of system is meant to operate in real conditions, where the vehicles are affected by noises from the sensors or external perturbations from the environment.

The contribution of this chapter is the deployment of a MVS subject to bounded perturbations in Section 4.1 and then investigates control strategies when the agents in

the MVS are affected by stochastic noises and external perturbations in Section 4.2. In both cases, the model considered for a vehicle is presented along with other dynamics (prediction model, state observer) that are necessary to formulate the model predictive control problem for the Voronoi-based deployment of a MVS. In addition, an overview of existing control methods for systems subject to bounded and unbounded perturbations is developed according to the considered case. Then, in each case, the deployment algorithm is presented. Finally, the deployment algorithm is tested on multi-vehicle systems where each vehicle obeys single integrator and quadrotor unmanned aerial vehicle (UAV) dynamics. As in Chapter 3, the names multi-agent system (MAS) and multi-vehicle system (MVS) are used indifferently since here, the agents are all vehicles.

4.1 Bounded perturbations

In the following section, bounded perturbations are considered on the system model. To this end, Section 4.1.1 introduces the necessary modifications on the dynamics of an agent part of a larger MAS as well as the essential dynamics for the development of a control algorithm. Section 4.1.2 presents an overview of existing methods for control of systems subject to bounded perturbations. The decentralized model predictive control algorithm for Voronoi-based deployment of a MAS presented in Section 3.2.2 is modified using a tube-based MPC approach with guaranteed Voronoi cells to cope with bounded perturbations in Section 4.1.3 before finally being applied in simulation to a MAS composed of agents obeying single integrator or UAV dynamics.

4.1.1 System model

Let Σ be a multi-agent system composed of N agents. Each agent obeys discrete-time linear time-invariant dynamics as in the previous chapter. In addition to (3.1), each agent is now considered to be subject to bounded input and output perturbations. The dynamics of an agent is then:

$$\mathbf{x}_i(k+1) = \boldsymbol{A}\mathbf{x}_i(k) + \boldsymbol{B}\mathbf{u}_i(k) + \mathbf{d}_i(k) \tag{4.1a}$$

$$\mathbf{y}_i(k) = \boldsymbol{C}\mathbf{x}_i(k) + \mathbf{w}_i(k) \tag{4.1b}$$

where $\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^n$, $\mathbf{u}_i \in \mathcal{U} \subset \mathbb{R}^m$, $\mathbf{y}_i \in \mathcal{Y} \subset \mathbb{R}^2$, with $i \in \overline{1,N}$, $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{B} \in \mathbb{R}^{n\times m}$ and $\boldsymbol{C} \in \mathbb{R}^{2\times n}$. With Assumption 2.6, all the agents share the same dynamics and, with Assumption 3.1, all the agents share the same output space. Then the dependency in $i \in \overline{1,N}$ is dropped for $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ as well as for the output space $\mathcal{Y}$ which is a part of the plane $\mathbb{R}^2$. The dependency in $i \in \overline{1,N}$ for $\mathcal{X}$ and $\mathcal{U}$ is also dropped for the sake of simplicity. The signals $\mathbf{d}_i \in \mathcal{D} \subset \mathbb{R}^n$ and $\mathbf{w}_i \in \mathcal{W} \subset \mathbb{R}^2$, with $i \in \overline{1,N}$, are the input and output perturbations, respectively. The sets $\mathcal{D}$ and $\mathcal{W}$ are boxes such that $\mathcal{D} = \mathbb{B}^n(\boldsymbol{\alpha}_d)$ and $\mathcal{W} = \mathbb{B}^2(\boldsymbol{\alpha}_w)$, where $\boldsymbol{\alpha}_d \in \mathbb{R}^n$ and $\boldsymbol{\alpha}_w \in \mathbb{R}^2$.

In order to apply control strategies to such a system, it is necessary to define the nominal dynamics associated to (4.1):

$$\check{\mathbf{x}}_i(k+1) = \boldsymbol{A}\check{\mathbf{x}}_i(k) + \boldsymbol{B}\check{\mathbf{u}}_i(k) \tag{4.2a}$$

$$\check{\mathbf{y}}_i(k) = \boldsymbol{C}\check{\mathbf{x}}_i(k) \tag{4.2b}$$

where $\check{\mathbf{x}}_i \in \mathbb{R}^n$, $\check{\mathbf{u}}_i \in \mathbb{R}^m$ and $\check{\mathbf{y}}_i \in \mathbb{R}^2$. The matrices $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ are the same as in (4.1). This is the dynamics that the system would follow if it was not subject to

any perturbation and is identical to the dynamics used in Chapter 3. It is assumed that Assumption 3.2 holds for the system defined by (4.2) and that the nominal dynamics (4.2) then admits equilibrium points. The dynamics (4.2) is the one used in the model predictive controller presented in Section 4.1.3.

Since, in the general case, there are more states than measured outputs, i.e. $n \geq 2$, the state of system (4.2) is estimated with a Luenberger observer (Luenberger, 1964). The dynamics of the state observer is then:

$$\widehat{\mathbf{x}}_i(k+1) = \boldsymbol{A}\widehat{\mathbf{x}}_i(k) + \boldsymbol{B}\mathbf{u}_i(k) + \boldsymbol{L}(\mathbf{y}_i(k) - \widehat{\mathbf{y}}_i(k)) \tag{4.3a}$$

$$\widehat{\mathbf{y}}_i(k) = \boldsymbol{C}\widehat{\mathbf{x}}_i(k) \tag{4.3b}$$

where $\boldsymbol{L} \in \mathbb{R}^{n\times 2}$ is the observer gain, $\widehat{\mathbf{x}}_i(k) \in \mathbb{R}^n$ is the estimated state vector and $\widehat{\mathbf{y}}_i(k) \in \mathbb{R}^2$ is the estimated output vector.

From the dynamics of the estimated state, it is possible to derive the *estimation error* $\widetilde{\mathbf{x}}_i$, i.e. the difference between the real state $\mathbf{x}_i$ and the estimated state $\widehat{\mathbf{x}}_i$, obeying the dynamics:

$$\begin{aligned}\widetilde{\mathbf{x}}_i(k+1) &= \mathbf{x}_i(k+1) - \widehat{\mathbf{x}}_i(k+1)\\ &= \boldsymbol{A}\mathbf{x}_i(k) + \boldsymbol{B}\mathbf{u}_i(k) + \mathbf{d}_i(k) - \boldsymbol{A}\widehat{\mathbf{x}}_i(k) - \boldsymbol{B}\mathbf{u}_i(k) - \boldsymbol{L}(\mathbf{y}_i(k) - \widehat{\mathbf{y}}_i(k))\\ &= (\boldsymbol{A} - \boldsymbol{LC})(\mathbf{x}_i(k) - \widehat{\mathbf{x}}_i(k)) - \boldsymbol{L}\mathbf{w}_i(k) + \mathbf{d}_i(k)\end{aligned}$$

that can be summarized as:

$$\widetilde{\mathbf{x}}_i(k+1) = (\boldsymbol{A} - \boldsymbol{LC})\widetilde{\mathbf{x}}_i(k) - \boldsymbol{L}\mathbf{w}_i(k) + \mathbf{d}_i(k). \tag{4.4}$$

Another necessary dynamics for the remainder of this section is the *deviation error* $\breve{\mathbf{x}}_i$, i.e. the difference between the estimated state $\widehat{\mathbf{x}}_i$ and the nominal state $\breve{\mathbf{x}}_i$, obeying the dynamics:

$$\begin{aligned}\breve{\mathbf{x}}_i(k+1) &= \widehat{\mathbf{x}}_i(k+1) - \breve{\mathbf{x}}_i(k+1)\\ &= \boldsymbol{A}\widehat{\mathbf{x}}_i(k) + \boldsymbol{B}\mathbf{u}_i(k) + \boldsymbol{L}(\mathbf{y}_i(k) - \widehat{\mathbf{y}}_i(k)) - \boldsymbol{A}\breve{\mathbf{x}}_i(k) - \boldsymbol{B}\breve{\mathbf{u}}_i(k)\\ &= \boldsymbol{A}\breve{\mathbf{x}}_i(k) + \boldsymbol{B}(\mathbf{u}_i(k) - \breve{\mathbf{u}}_i(k)) + \boldsymbol{LC}\widetilde{\mathbf{x}}_i(k) + \boldsymbol{L}\mathbf{w}_i(k).\end{aligned}$$

Since the control algorithm that is presented in Section 4.1.3 is based on classical robust tube-based MPC (Mayne et al., 2006), the control input is chosen such that:

$$\mathbf{u}_i(k) = \breve{\mathbf{u}}_i(k) + \boldsymbol{K}(\widehat{\mathbf{x}}_i(k) - \breve{\mathbf{x}}_i(k)) \tag{4.5}$$

where $\breve{\mathbf{u}}_i$ is obtained by solving a MPC optimization problem and $\boldsymbol{K} \in \mathbb{R}^{m\times n}$ is a gain. Then, the deviation error dynamics can be summarized as:

$$\breve{\mathbf{x}}_i(k+1) = (\boldsymbol{A} + \boldsymbol{BK})\breve{\mathbf{x}}_i(k) + \boldsymbol{LC}\widetilde{\mathbf{x}}_i(k) + \boldsymbol{L}\mathbf{w}_i(k). \tag{4.6}$$

Given the dynamics (4.4) and (4.6), the gain matrices $\boldsymbol{L}$ and $\boldsymbol{K}$ have to be chosen such that $\boldsymbol{A} - \boldsymbol{LC}$ and $\boldsymbol{A} + \boldsymbol{BK}$ have their poles with module lower than one. Moreover, based on the definition of the estimation and deviation error, it is also immediate that:

$$\mathbf{x}_i(k) = \breve{\mathbf{x}}_i(k) + \widetilde{\mathbf{x}}_i(k) + \breve{\mathbf{x}}_i(k). \tag{4.7}$$

4.1.2 Overview of robust tube-based MPC for systems with bounded perturbations

When systems are subject to perturbations, existing control methods have to be adapted to cope with the different behavior that could arise from the action of endogenous or exogenous signals. One of the most widely studied case is when these perturbations are contained in a compact set as for the dynamics (4.1). One of the most famous approaches to deal with perturbations belonging to such a set is based on H_∞ optimization methods (Zames, 1981, Green and Limebeer, 2012). Another widespread approach is based on sliding mode (Drakunov and Utkin, 1992).

However, when dealing with dynamical systems subject to operating constraints, the most popular control technique is tube-based model predictive control (Langson et al., 2004, Mayne et al., 2005, 2006, Kouvaritakis and Cannon, 2016) which has been first studied in the state-feedback case (Langson et al., 2004, Mayne et al., 2005) where only the state equation (4.1a) is used, and later on the output-feedback case (Mayne et al., 2006), where both the state and the output are affected by perturbations.

The tube-based MPC strategy is based on the combination of a model predictive controller and a state-feedback controller such that the control input is given by (4.5). In the state-feedback case, the estimated state $\widehat{\mathbf{x}}_i$ is replaced by the state $\mathbf{x}_i$ itself in (4.5) since there is no need for state estimation. In this case, such a control strategy ensures that the system's state remains inside a robust positively invariant (RPI) set for the dynamics $\mathbf{x}_i(k+1) = (\boldsymbol{A}+\boldsymbol{BK})\mathbf{x}_i(k) + \mathbf{d}_i(k)$ centered on the nominal state $\check{\mathbf{x}}_i(k)$. The goal is then to find the minimal RPI (mRPI) (Blanchini and Miani, 2015), or an approximation of it, for the aforementioned dynamics. Then, it is guaranteed (Mayne et al., 2005) that if $\mathbf{x}_i(0) \in \{\check{\mathbf{x}}_i(0)\} \oplus \mathcal{S}_\mathbf{d}$, where $\mathcal{S}_\mathbf{d}$ is an approximation of the mRPI set for the dynamics of $\mathbf{x}_i(k) - \check{\mathbf{x}}_i(k)$, then one has $\mathbf{x}_i(k) \in \{\check{\mathbf{x}}_i(k)\} \oplus \mathcal{S}_\mathbf{d}$ for all $k > 0$. The way this control strategy works in the output feedback case is quite similar. The controller ensures that the system's state remains inside the sum of the approximation $\mathcal{S}_{\tilde{\mathbf{x}}}$ of the mRPI set for the estimation error dynamics (4.4) and of the approximation $\mathcal{S}_{\check{\mathbf{x}}}$ of the mRPI set for the deviation error dynamics (4.6) centered on the nominal state. Then, it is guaranteed (Mayne et al., 2006) that if $\mathbf{x}_i(0) \in \{\check{\mathbf{x}}_i(0)\} \oplus \mathcal{S}_{\tilde{\mathbf{x}}} \oplus \mathcal{S}_{\check{\mathbf{x}}}$, then, the following expression holds $\mathbf{x}_i(k) \in \{\check{\mathbf{x}}_i(k)\} \oplus \mathcal{S}_{\tilde{\mathbf{x}}} \oplus \mathcal{S}_{\check{\mathbf{x}}}$ for all $k > 0$.

Subsequent work on tube-based MPC has tried to improve its capabilities. For example Alvarado et al. (2007) and Limón et al. (2010) adapted the control strategy from Mayne et al. (2005) for trajectory tracking. Cannon et al. (2012) developed a tube-based MPC when the bounded perturbations have a stochastic nature[1]. Moreover, tube-based strategies have been applied on single vehicles such as ground vehicles (González et al., 2011, Gao et al., 2014, Kayacan et al., 2015, Sun et al., 2018), aerial vehicles (Santos et al., 2018, Chevet et al., 2019, Michel et al., 2019) or even spacecrafts (Mirshams and Khosrojerdi, 2016, Mammarella et al., 2018).

When it comes to multi-agent systems, several schemes of tube-based MPC have been developed. For example, Prodan et al. (2011) use tube-based MPC to drive several agents along a given trajectory, while using RPI sets for their dynamics to derive a safe configuration avoiding collision. Nikou and Dimarogonas (2019) present a decentralized tube-based MPC algorithm for navigation of a multi-agent

[1] In this thesis, the case of unbounded stocahstic perturbations is further investigated in Section 4.2.

system, while Trodden and Richards (2014) propose a distributed scheme for linear agents, the agents being coupled by constraints in both cases. The case of bounded stochastic perturbations has also been studied and it is addressed in Section 4.2.

The deployment problem when the agents are subject to bounded perturbations has been addressed in Chevet et al. (2019). It is based on the work on optimal coverage by systems with uncertain location (Papatheodorou et al., 2016, 2017, Tzes et al., 2018). It starts by considering that the uncertainty on the location of an agent is represented by a perturbation on the output equation (4.1b) and derives a tube-based MPC scheme to drive the agents into a static Chebyshev configuration. Then, in the case of Chevet et al. (2019), the perturbation $\mathbf{d}_i(k)$ appearing in (4.1a) is such that $\mathbf{d}_i(k) = \mathbf{0}_{n\times 1}$ and $\mathbf{w}_i(k) \in \mathcal{W}$ for all $i \in \overline{1,N}$, $k \geq 0$. The remainder of this section builds upon Chevet et al. (2019) to also consider bounded perturbations on the state equation which is a generalization of the strategy already exposed.

4.1.3 Deployment algorithm

As in Chapter 3, the objective of the deployment algorithm that is presented in this paragraph is to drive the MAS defined in Section 4.1.1 into a static configuration in $\mathcal{Y}$. The set $\mathcal{Y}$ is a convex bounded polytope of $\mathbb{R}^2$ as defined in Assumption 3.1. However, an important difference with the algorithm for the nominal case presented in Chapter 3 appears due to the bounded perturbations on the dynamics (4.1). Indeed, the deployment algorithm in the nominal case is based on the Voronoi tessellation of $\mathcal{Y}$, where the positions $\mathbf{y}_i \in \mathcal{Y}$, with $i \in \overline{1,N}$, are the generators of the said tessellation. Here, a bounded additive perturbation exists on the position $\mathbf{y}_i$ of each agent $i \in \overline{1,N}$ and it is not reliable enough for each agent to compute its Voronoi cell.

4.1.3.1 Deployment objective in the perturbed case

Based on Assumption 2.5, a first assumption is made to deal with the case of bounded perturbations.

Assumption 4.1: Knowledge of environment in the perturbed case
Each agent of Σ *knows, at all time, the nominal output of the other agents of* Σ.

With Assumption 4.1 and the knowledge of the output perturbation set $\mathcal{W}$, it is possible for each agent to compute its guaranteed Voronoi cell (GVC) as defined in Paragraph 2.4.2.1.

Remark 4.1: Inhomogeneous perturbations
In the case of inhomogeneous perturbations on the agents of Σ, i.e. in the case where each agent is affected by a different perturbation belonging to a set $\mathcal{W}(i)$, with $i \in \overline{1,N}$, Assumption 4.1 is modified to include the knowledge of all $\mathcal{W}(i)$. $\Diamond$

The deployment objective is modified to cope with the perturbations. At each time instant, each agent $i \in \overline{1,N}$ computes its guaranteed Voronoi cell $\mathcal{V}_i^g(k)$ with the knowledge of $\check{\mathbf{y}}_j(k)$, with $j \in \overline{1,N}$, $j \neq i$. The generator sets of such GVC are the boxes $\mathcal{W}$ centered on the nominal position of each agent $\mathcal{W}_i(k) = \{\check{\mathbf{y}}_i(k)\} \oplus \mathcal{W}$. The construction of the guaranteed Voronoi tessellation generated by rectangles has been extensively presented in Paragraph 2.4.2.1 and the reader can refer to it for more information.

With the knowledge of its guaranteed Voronoi cell $\mathcal{V}_i^g(k)$, each agent is able to compute the Chebyshev center $\mathbf{c}_i(k)$ of $\mathcal{V}_i^g(k)$ by solving the problem (3.5). The objective of the controller is then to drive the agent towards the Chebyshev center $\mathbf{c}_i(k)$ of $\mathcal{V}_i^g(k)$ using a tube-based MPC algorithm. However, while in the nominal case, each agent had to reach the Chebyshev center of its Voronoi cell, i.e. $\lim_{k\to\infty} \mathbf{y}_i(k) = \lim_{k\to\infty} \mathbf{c}_i(k)$, here, the nominal position of each agent has to reach the Chebyshev center of its GVC, i.e. $\lim_{k\to\infty} \breve{\mathbf{y}}_i(k) = \lim_{k\to\infty} \mathbf{c}_i(k)$, while the position $\mathbf{y}_i(k)$ of the agent belongs to a RPI set centered on $\breve{\mathbf{y}}_i(k)$. To do so, the input signal is as presented in (4.5), where $\breve{\mathbf{u}}_i(k)$ is obtained by solving a MPC optimization problem and $\boldsymbol{K}$ is a gain matrix that has to be computed[2]. The following paragraph introduces the optimization problem solved to obtain $\breve{\mathbf{u}}_i(k)$ as well as the associated constraints.

4.1.3.2 Optimization problem for the tube-based MPC

Let $i \in \overline{1, N}$ be an agent of the MAS. The computation of its guaranteed Voronoi cell $\mathcal{V}_i^g(k)$ and its Chebyshev center $\mathbf{c}_i(k)$ have been covered in Paragraph 4.1.3.1. With Assumption 3.2, it is possible to find a couple $(\breve{\mathbf{x}}_{\mathbf{c}_i}(k), \breve{\mathbf{u}}_{\mathbf{c}_i}(k))$ such that $(\breve{\mathbf{x}}_{\mathbf{c}_i}(k), \breve{\mathbf{u}}_{\mathbf{c}_i}(k), \mathbf{c}_i(k))$ is an equilibrium point of (4.2) satisfying:

$$\begin{aligned} \breve{\mathbf{x}}_{\mathbf{c}_i}(k) &= \boldsymbol{A}\breve{\mathbf{x}}_{\mathbf{c}_i}(k) + \boldsymbol{B}\breve{\mathbf{u}}_{\mathbf{c}_i}(k) \\ \mathbf{c}_i(k) &= \boldsymbol{C}\breve{\mathbf{x}}_{\mathbf{c}_i}(k). \end{aligned} \tag{4.8}$$

With all the elements computed above, the nominal input $\breve{\mathbf{u}}_i(k)$ for agent $i \in \overline{1, N}$ is computed by finding the solution of the optimization problem:

$$\underset{\substack{\breve{\mathbf{u}}_i(k+l),\\ \forall l \in \overline{0, N_p-1}}}{\text{minimize}} \quad \sum_{l=0}^{N_p-1} \ell(\breve{\mathbf{x}}_i(k+l), \breve{\mathbf{u}}_i(k+l), \breve{\mathbf{x}}_{\mathbf{c}_i}(k), \breve{\mathbf{u}}_{\mathbf{c}_i}(k)) + V(\breve{\mathbf{x}}_i(k+N_p), \breve{\mathbf{x}}_{\mathbf{c}_i}(k)) \tag{4.9a}$$

subject to

$$\breve{\mathbf{x}}_i(k+l+1) = \boldsymbol{A}\breve{\mathbf{x}}_i(k+l) + \boldsymbol{B}\breve{\mathbf{u}}_i(k+l), \qquad \forall l \in \overline{0, N_p-1}, \tag{4.9b}$$

$$\breve{\mathbf{x}}_i(k+l) \in \breve{\mathcal{X}}, \qquad \forall l \in \overline{0, N_p-1}, \tag{4.9c}$$

$$\breve{\mathbf{u}}_i(k+l) \in \breve{\mathcal{U}}, \qquad \forall l \in \overline{0, N_p-1}, \tag{4.9d}$$

$$\boldsymbol{C}\breve{\mathbf{x}}_i(k+l) \in \breve{\mathcal{V}}_i^g(k), \qquad \forall l \in \overline{0, N_p-1}, \tag{4.9e}$$

$$\boldsymbol{C}\breve{\mathbf{x}}_i(k+N_p) \in \breve{\Omega}_i(k) \tag{4.9f}$$

where $\ell(\breve{\mathbf{x}}_i(k+l), \breve{\mathbf{u}}_i(k+l), \breve{\mathbf{x}}_{\mathbf{c}_i}(k), \breve{\mathbf{u}}_{\mathbf{c}_i}(k))$, with $l \in \overline{0, N_p-1}$, is the stage cost:

$$\begin{aligned} &\ell(\breve{\mathbf{x}}_i(k+l), \breve{\mathbf{u}}_i(k+l), \breve{\mathbf{x}}_{\mathbf{c}_i}(k), \breve{\mathbf{u}}_{\mathbf{c}_i}(k)) \\ &\qquad = \|\breve{\mathbf{x}}_i(k+l) - \breve{\mathbf{x}}_{\mathbf{c}_i}(k)\|_{\boldsymbol{Q}_i}^2 + \|\breve{\mathbf{u}}_i(k+l) - \breve{\mathbf{u}}_{\mathbf{c}_i}(k)\|_{\boldsymbol{R}_i}^2 \end{aligned} \tag{4.10}$$

and $V(\breve{\mathbf{x}}_i(k+N_p), \breve{\mathbf{x}}_{\mathbf{c}_i}(k))$ is the terminal cost:

$$V(\breve{\mathbf{x}}_i(k+N_p), \breve{\mathbf{x}}_{\mathbf{c}_i}(k)) = \|\breve{\mathbf{x}}_i(k+N_p) - \breve{\mathbf{x}}_{\mathbf{c}_i}(k)\|_{\boldsymbol{P}_i}^2. \tag{4.11}$$

[2]The computation of the gain matrix $\boldsymbol{K}$ is detailed in Paragraph 4.1.3.3.

The weighting matrices $\boldsymbol{Q}_i, \boldsymbol{P}_i \in \mathbb{R}^{n\times n}$ and $\boldsymbol{R}_i \in \mathbb{R}^{m\times m}$ in (4.10) and (4.11) are chosen such that $\boldsymbol{Q}_i = \boldsymbol{Q}_i^\top \succ 0$, $\boldsymbol{P}_i = \boldsymbol{P}_i^\top \succ 0$ and $\boldsymbol{R}_i = \boldsymbol{R}_i^\top \succ 0$. The prediction horizon N_p is a positive integer.

The decentralized optimization problem (4.9) is similar to the decentralized problem (3.12) used in the nominal case. However, contrary to the nominal case, as was explained in the previous paragraphs, the model predictive controller runs for the nominal system (4.2), hence the presence of the nominal state and nominal input in the cost function as well as in the constraints of (4.9). Moreover, the constraints have to be heavily modified with respect to their nominal counterpart to be adapted to the tube-based control law that is applied in the case of bounded perturbations.

The constraint (4.9b) predicts the future value of the nominal state $\breve{\mathbf{x}}_i(k+l+1)$ for all $l \in \overline{0, N_p-1}$ of the agent i given the value of the decision variables, i.e. the nominal input sequence $\breve{\mathbf{u}}_i(k+l)$ for all $l \in \overline{0, N_p-1}$.

In constraints (4.9c)-(4.9f), the notation $\breve{\cdot}$ over a set denotes a set to which an invariant set has been "subtracted". For example, the constraint (4.9c) ensures that, over the prediction horizon N_p, the nominal state of agent i lies inside the state space $\mathcal{X}$, a convex polytope, to which the mRPI set for the estimation error $\mathcal{S}_{\tilde{\mathbf{x}}}$ and the mRPI set for the deviation error $\mathcal{S}_{\check{\mathbf{x}}}$ are subtracted. Then, the nominal state is constrained inside $\breve{\mathcal{X}} = \mathcal{X} \ominus \mathcal{S}_{\tilde{\mathbf{x}}} \ominus \mathcal{S}_{\check{\mathbf{x}}}$. Indeed, the final objective of the control law is to have $\mathbf{x}_i \in \mathcal{X}$ and the expression of $\breve{\mathcal{X}}$ is obtained from the relation (4.7).

The constraint (4.9d) ensures that the nominal input of agent i lies inside the input set $\breve{\mathcal{U}} = \mathcal{U} \ominus \boldsymbol{K}\mathcal{S}_{\check{\mathbf{x}}}$ over the prediction horizon N_p, where $\mathcal{U}$ is a convex polytope. This relation comes from the expression of the full control input (4.5) and the fact that, as for the state, the goal is to have $\mathbf{u}_i \in \mathcal{U}$.

The constraint (4.9e) constrains the agent's nominal output to belong to a restricted version of the agent's guaranteed Voronoi cell $\breve{\mathcal{V}}_i^g(k) = \mathcal{V}_i^g(k) \ominus \boldsymbol{C}\mathcal{S}_{\tilde{\mathbf{x}}} \ominus \boldsymbol{C}\mathcal{S}_{\check{\mathbf{x}}}$ over the prediction horizon. The expression of $\breve{\mathcal{V}}_i^g(k)$ has the same origin as the expression of $\breve{\mathcal{X}}$. For the same reason as in Section 3.2.2, this constraint is more restrictive on the states participating in the output than the state constraint (4.9e).

Finally, the terminal set appearing in the constraint (4.9f) is defined as $\breve{\Omega}_i(k) = \Omega_i(k) \ominus \boldsymbol{C}(\mathcal{S}_{\tilde{\mathbf{x}}} \oplus \mathcal{S}_{\check{\mathbf{x}}})$ where $\Omega_i(k)$ is defined in Assumption 3.5. Indeed, if $\breve{\Omega}_i(k) = \{\mathbf{c}_i(k)\} \oplus \lambda_i(k)(\mathcal{V}_i^g(k) \oplus \{-\mathbf{c}_i(k)\}) \ominus \boldsymbol{C}(\mathcal{S}_{\tilde{\mathbf{x}}} \oplus \mathcal{S}_{\check{\mathbf{x}}})$, it is ensured that $\breve{\Omega}_i(k)$ is controlled invariant by Theorem 3.2 for the nominal dynamics (4.2).

By solving the optimization problem (4.9), the first element $\breve{\mathbf{u}}_i(k)$ of the control input (4.5) to be applied to the system (4.3) is obtained. However, for the second part of the control input as well as for solving the optimization problem, it is necessary to find a gain matrix $\boldsymbol{K}$ for the state-feedback part of the input and invariant sets $\mathcal{S}_{\tilde{\mathbf{x}}}$ and $\mathcal{S}_{\check{\mathbf{x}}}$ for the error dynamics (4.4) and (4.6), respectively. The proposed tuning procedure to obtain the state-feedback and observer gain matrices $\boldsymbol{K}$ and $\boldsymbol{L}$ and the method proposed to compute the invariant sets $\mathcal{S}_{\tilde{\mathbf{x}}}$ and $\mathcal{S}_{\check{\mathbf{x}}}$ are presented in the next paragraph.

4.1.3.3 Tube envelope

Remark 4.2: Index
For this entire paragraph, the index $i \in \overline{1, N}$ is used to designate an agent of Σ. $\Diamond$

When designing a tube-based MPC scheme, the gain matrix $\boldsymbol{K}$ appearing in the control input (4.5) has an important role as well as the observer gain matrix

$\boldsymbol{L}$ appearing in the estimation error dynamics (4.4). The design method for both gain matrices $\boldsymbol{K}$ and $\boldsymbol{L}$ is based on the method proposed by Limón et al. (2010). However, in Limón et al. (2010), the tuning is proposed in the state-feedback case (i.e. only the gain matrix $\boldsymbol{K}$ is obtained). Given the dynamics of the estimation and deviation error (4.4) and (4.6), both gain matrices $\boldsymbol{K}$ and $\boldsymbol{L}$ are linked and could be obtained from one single optimization problem. However, such a procedure would be complex and lead to polynomial matrix inequalities while the procedures proposed in the following lead to bilinear matrix inequalities (BMI) that can be simplified, and even relaxed into linear matrix inequalities (LMI). Then, the procedures proposed in this thesis consider that the gains are independent and are tuned sequentially. A discussion on the robustification of such a procedure is held in Section 4.3.

The tuning procedure for the control gain matrix $\boldsymbol{K}$ in Limón et al. (2010) is based on three elements. For the present problem, i.e. input and output perturbations on the system, these elements are:

(i) the existence of a RPI set $\breve{\mathcal{S}}$ (respectively $\widetilde{\mathcal{S}}$) for the dynamics (4.6) (respectively (4.4));

(ii) the RPI set is such that the sets $\mathcal{X} \ominus \breve{\mathcal{S}}$ and $\mathcal{U} \ominus \boldsymbol{K}\breve{\mathcal{S}}$ (respectively $\mathcal{X} \ominus \widetilde{\mathcal{S}}$) are not empty and large enough for the MPC to have enough degrees of freedom to optimize the performance of the system ;

(iii) the size of the RPI set $\breve{\mathcal{S}}$ (respectively $\widetilde{\mathcal{S}}$) is minimal to reduce the impact of perturbations on the closed-loop system.

For this procedure, the RPI sets are considered to be centered and normalized ellipsoidal sets $\widetilde{\mathcal{S}} = \mathscr{E}(\widetilde{\boldsymbol{P}})$ and $\breve{\mathcal{S}} = \mathscr{E}(\breve{\boldsymbol{P}})$, with $\widetilde{\boldsymbol{P}}, \breve{\boldsymbol{P}} \in \mathbb{R}^{n\times n}$ such that $\widetilde{\boldsymbol{P}} = \widetilde{\boldsymbol{P}}^\top \succ 0$ and $\breve{\boldsymbol{P}} = \breve{\boldsymbol{P}}^\top \succ 0$. In (ii), the sets $\mathcal{X} \ominus \widetilde{\mathcal{S}}$ and $\mathcal{X} \ominus \breve{\mathcal{S}}$ are preferred to the set $\mathcal{X} \ominus \widetilde{\mathcal{S}} \ominus \breve{\mathcal{S}}$ since the use of the latter would lead, as mentioned earlier, to polynomial matrix inequalities in the tuning procedure.

Remark 4.3: Invariant sets for the tuning procedures
The RPI sets $\widetilde{\mathcal{S}}$ and $\breve{\mathcal{S}}$ defined for the tuning procedures are not the same sets as $\mathcal{S}_{\widetilde{\mathbf{x}}}$ and $\mathcal{S}_{\breve{\mathbf{x}}}$ used for control. Indeed, the sets $\widetilde{\mathcal{S}}$ and $\breve{\mathcal{S}}$ are outer ellipsoidal approximations of the mRPI sets $\mathcal{S}_{\widetilde{\mathbf{x}}}$ and $\mathcal{S}_{\breve{\mathbf{x}}}$ for the dynamics (4.4) and (4.6), respectively. ◊

The elements (i)-(iii) can be translated mathematically for both procedures. The fact that $\widetilde{\mathcal{S}}$ is RPI for (4.4) can be translated into:

$$\widetilde{\mathbf{x}}_i(k+1)^\top \widetilde{\boldsymbol{P}} \widetilde{\mathbf{x}}_i(k+1) \leq 1,\ \forall \widetilde{\mathbf{x}}_i(k) \in \widetilde{\mathcal{S}},\ \forall \mathbf{d}_i(k) \in \mathcal{D},\ \forall \mathbf{w}_i(k) \in \mathcal{W}.$$

Taking into account that $\widetilde{\mathbf{x}}_i(k+1)$ is convex with respect to $\mathbf{d}_i(k)$ and $\mathbf{w}_i(k)$, by applying the S-procedure (Theorem 2.1), the previous equation is satisfied if there exists a real $\tau \geq 0$ such that:

$$\begin{aligned}((\boldsymbol{A}-\boldsymbol{LC})\widetilde{\mathbf{x}}_i(k) - \boldsymbol{L}\mathbf{w} + \mathbf{d})^\top \widetilde{\boldsymbol{P}}((\boldsymbol{A}-\boldsymbol{LC})\widetilde{\mathbf{x}}_i(k) - \boldsymbol{L}\mathbf{w} + \mathbf{d}) \\ + \tau\Big(1 - \widetilde{\mathbf{x}}_i(k)^\top \widetilde{\boldsymbol{P}} \widetilde{\mathbf{x}}_i(k)\Big) \leq 1,\ \forall \mathbf{d} \in \mathrm{vert}(\mathcal{D}),\ \forall \mathbf{w} \in \mathrm{vert}(\mathcal{W})\end{aligned}$$

where $\mathrm{vert}(\mathcal{D})$ and $\mathrm{vert}(\mathcal{W})$ are the sets of all vertices of the sets $\mathcal{D}$ and $\mathcal{W}$, respectively. Such an inequality can be rewritten:

$$\begin{bmatrix}\widetilde{\mathbf{x}}_i(k) \\ 1\end{bmatrix}^\top \begin{bmatrix}\tau\widetilde{\boldsymbol{P}} - \boldsymbol{A}_{\boldsymbol{L}}^\top \widetilde{\boldsymbol{P}} \boldsymbol{A}_{\boldsymbol{L}} & \boldsymbol{A}_{\boldsymbol{L}}^\top \widetilde{\boldsymbol{P}}(\boldsymbol{L}\mathbf{w} - \mathbf{d}) \\ (\boldsymbol{L}\mathbf{w} - \mathbf{d})^\top \widetilde{\boldsymbol{P}} \boldsymbol{A}_{\boldsymbol{L}} & 1 - \tau - (\boldsymbol{L}\mathbf{w} - \mathbf{d})^\top \widetilde{\boldsymbol{P}}(\boldsymbol{L}\mathbf{w} - \mathbf{d})\end{bmatrix} \begin{bmatrix}\widetilde{\mathbf{x}}_i(k) \\ 1\end{bmatrix} \geq 0 \quad (4.12)$$

for all $\mathbf{d} \in \text{vert}(\mathcal{D})$ and $\mathbf{w} \in \text{vert}(\mathcal{W})$, where $\boldsymbol{A_L} = \boldsymbol{A} - \boldsymbol{LC}$. The matrix appearing in (4.12) can be rewritten $\forall \widetilde{\mathbf{x}}_i(k) \in \widetilde{\mathcal{S}}$, $\forall \mathbf{d} \in \text{vert}(\mathcal{D})$, $\forall \mathbf{w} \in \text{vert}(\mathcal{W})$, as the BMI:

$$\begin{bmatrix} \tau\widetilde{\boldsymbol{P}} - \boldsymbol{A_L}^\top\widetilde{\boldsymbol{P}}\boldsymbol{A_L} & \boldsymbol{A_L}^\top\widetilde{\boldsymbol{P}}(\boldsymbol{L}\mathbf{w} - \mathbf{d}) \\ (\boldsymbol{L}\mathbf{w} - \mathbf{d})^\top\widetilde{\boldsymbol{P}}\boldsymbol{A_L} & 1 - \tau - (\boldsymbol{L}\mathbf{w} - \mathbf{d})^\top\widetilde{\boldsymbol{P}}(\boldsymbol{L}\mathbf{w} - \mathbf{d}) \end{bmatrix} \succeq 0$$

which can be separated into:

$$\begin{bmatrix} \tau\widetilde{\boldsymbol{P}} & \mathbf{0}_{n\times 1} \\ \mathbf{0}_{1\times n} & 1-\tau \end{bmatrix} - \begin{bmatrix} \boldsymbol{A_L}^\top \\ (\mathbf{d} - \boldsymbol{L}\mathbf{w})^\top \end{bmatrix} \widetilde{\boldsymbol{P}} \begin{bmatrix} \boldsymbol{A_L} & \mathbf{d} - \boldsymbol{L}\mathbf{w} \end{bmatrix} \succeq 0,$$

$\forall \mathbf{d} \in \text{vert}(\mathcal{D})$, $\forall \mathbf{w} \in \text{vert}(\mathcal{W})$, or, equivalently, into:

$$\begin{bmatrix} \tau\widetilde{\boldsymbol{P}} & \mathbf{0}_{n\times 1} \\ \mathbf{0}_{1\times n} & 1-\tau \end{bmatrix} - \begin{bmatrix} \boldsymbol{A_L}^\top\widetilde{\boldsymbol{P}} \\ (\mathbf{d} - \boldsymbol{L}\mathbf{w})^\top\widetilde{\boldsymbol{P}} \end{bmatrix} \widetilde{\boldsymbol{P}}^{-1} \begin{bmatrix} \widetilde{\boldsymbol{P}}\boldsymbol{A_L} & \widetilde{\boldsymbol{P}}(\mathbf{d} - \boldsymbol{L}\mathbf{w}) \end{bmatrix} \succeq 0, \tag{4.13}$$

$\forall \mathbf{d} \in \text{vert}(\mathcal{D})$, $\forall \mathbf{w} \in \text{vert}(\mathcal{W})$. By using the Schur complement (2.7), (4.13) is equivalent to:

$$\begin{bmatrix} \tau\widetilde{\boldsymbol{P}} & \mathbf{0}_{n\times 1} & \boldsymbol{A_L}^\top\widetilde{\boldsymbol{P}} \\ \mathbf{0}_{1\times n} & 1-\tau & (\mathbf{d} - \boldsymbol{L}\mathbf{w})^\top\widetilde{\boldsymbol{P}} \\ \widetilde{\boldsymbol{P}}\boldsymbol{A_L} & \widetilde{\boldsymbol{P}}(\mathbf{d} - \boldsymbol{L}\mathbf{w}) & \widetilde{\boldsymbol{P}} \end{bmatrix} \succeq 0, \forall \mathbf{d} \in \text{vert}(\mathcal{D}), \forall \mathbf{w} \in \text{vert}(\mathcal{W}).$$

Thus, the inequality (4.12) is equivalent to the BMI:

$$\begin{bmatrix} \tau\widetilde{\boldsymbol{P}} & \star & \star \\ \mathbf{0}_{1\times n} & 1-\tau & \star \\ \widetilde{\boldsymbol{P}}\boldsymbol{A} - \widetilde{\boldsymbol{Y}}\boldsymbol{C} & \widetilde{\boldsymbol{P}}\mathbf{d} - \widetilde{\boldsymbol{Y}}\mathbf{w} & \widetilde{\boldsymbol{P}} \end{bmatrix} \succeq 0, \forall \mathbf{d} \in \text{vert}(\mathcal{D}), \forall \mathbf{w} \in \text{vert}(\mathcal{W}) \tag{4.14}$$

with the decision variables $\tau \in \mathbb{R}_+$, $\widetilde{\boldsymbol{P}} \in \mathbb{R}^{n\times n}$ and $\widetilde{\boldsymbol{Y}} = \widetilde{\boldsymbol{P}}\boldsymbol{L} \in \mathbb{R}^{n\times 2}$. The placeholder $\star$ designates symmetrical terms. The attentive reader notices that in (4.14), the only bilinear term is the product between the scalar τ and the matrix $\widetilde{\boldsymbol{P}}$. Then, the BMI (4.14) can be relaxed into a LMI if τ is fixed arbitrarily.

To ensure that $\widetilde{\mathcal{S}}$ is of minimal size as per (iii), the measure of the size of the RPI set chosen in Limón et al. (2010) is also chosen here. It consists in a parameter $\widetilde{\eta}$ such that $\widetilde{\mathcal{S}} \subseteq \sqrt{\widetilde{\eta}}\mathcal{X}$. Thus, minimizing the size of $\widetilde{\mathcal{S}}$ can be transformed into the problem of the minimization of the value of the parameter $\widetilde{\eta}$ subject to the constraint $\widetilde{\mathcal{S}} \subseteq \sqrt{\widetilde{\eta}}\mathcal{X}$. Considering that $\mathcal{X}$ is a polytope induced by $\boldsymbol{H}_\mathcal{X} \in \mathbb{R}^{r_x\times n}$ and $\boldsymbol{\theta}_\mathcal{X} \in \mathbb{R}^{r_x}$, the constraint $\widetilde{\mathcal{S}} \subseteq \sqrt{\widetilde{\eta}}\mathcal{X}$ can be rewritten as a LMI (Boyd et al., 1994):

$$\begin{bmatrix} \widetilde{\eta}\theta_{\mathcal{X},j}^2 & \boldsymbol{h}_{\mathcal{X},j} \\ \boldsymbol{h}_{\mathcal{X},j}^\top & \widetilde{\boldsymbol{P}} \end{bmatrix} \succeq 0, \forall j \in \overline{1, r_x} \tag{4.15}$$

where $\boldsymbol{h}_{\mathcal{X},j}^\top$ and $\theta_{\mathcal{X},j}$ are the j-th rows of $\boldsymbol{H}_\mathcal{X}$ and $\boldsymbol{\theta}_\mathcal{X}$, with $j \in \overline{1, r_x}$.

With all these elements, the BMI constrained optimization problem to be solved to find the observer gain matrix $\boldsymbol{L}$ while minimizing the size of the set $\widetilde{\mathcal{S}}$ is:

$$
\begin{aligned}
&\underset{\tau,\widetilde{\eta},\widetilde{\boldsymbol{P}},\widetilde{\boldsymbol{Y}}}{\text{minimize}} \quad \widetilde{\eta} \\
&\text{subject to} \\
&\begin{bmatrix} \tau\widetilde{\boldsymbol{P}} & \star & \star \\ \boldsymbol{0}_{1\times n} & 1-\tau & \star \\ \widetilde{\boldsymbol{P}}\boldsymbol{A}-\widetilde{\boldsymbol{Y}}\boldsymbol{C} & \widetilde{\boldsymbol{P}}\mathbf{d}-\widetilde{\boldsymbol{Y}}\mathbf{w} & \widetilde{\boldsymbol{P}} \end{bmatrix} \succeq 0, \qquad \forall \mathbf{d}\in \mathrm{vert}(\mathcal{D}),\ \forall \mathbf{w}\in \mathrm{vert}(\mathcal{W}), \\
&\qquad\qquad \begin{bmatrix} \widetilde{\eta}\theta_{\mathcal{X},j}^2 & \star \\ \boldsymbol{h}_{\mathcal{X},j}^\top & \widetilde{\boldsymbol{P}} \end{bmatrix} \succeq 0, \qquad \forall j\in\overline{1,r_x}, \\
&\qquad\qquad \tau \geq 0, \\
&\qquad\qquad \widetilde{\eta}\in[0,1).
\end{aligned}
\tag{4.16}
$$

This problem has $n^2/2+5n/2+2$ scalar decision variables and $|\mathrm{vert}(\mathcal{D})|\cdot|\mathrm{vert}(\mathcal{W})|+r_x+3$ constraints.

By solving (4.16), it is then possible to obtain $\boldsymbol{L}=\widetilde{\boldsymbol{P}}^{-1}\widetilde{\boldsymbol{Y}}$. Then, using $\boldsymbol{L}$ and the perturbation set $\mathcal{D}\oplus(-\boldsymbol{L}\mathcal{W})$, Algorithm 2.1 is used to obtain an approximation of the mRPI set $\mathcal{S}_{\widetilde{\mathbf{x}}}$ for the dynamics (4.4). However, since this set is used in the tuning procedure for the control gain matrix $\boldsymbol{K}$ as well as to obtain an approximation of the mRPI set $\mathcal{S}_{\check{\mathbf{x}}}$ for the dynamics (4.6), the combination of Algorithm 2.1 and (2.39) can be used to keep a reasonable number of vertices for $\mathcal{S}_{\widetilde{\mathbf{x}}}$. For another objective than the deployment or to keep a reasonable complexity, other methods can be used to compute the approximation of the mRPI set such as the ones proposed in Raković et al. (2005) or Alvarado (2007).

When the observer gain matrix $\boldsymbol{L}$ and the approximation of the mRPI set $\mathcal{S}_{\widetilde{\mathbf{x}}}$ for the dynamics (4.4) are obtained, the tuning procedure for the control gain matrix $\boldsymbol{K}$ is run. This procedure is quite similar to the tuning procedure to obtain the observer gain matrix $\boldsymbol{L}$. The set $\check{\mathcal{S}}$ is RPI for (4.6) if:

$$
\check{\mathbf{x}}_i(k+1)^\top\check{\boldsymbol{P}}\check{\mathbf{x}}_i(k+1)\leq 1,\ \forall\check{\mathbf{x}}_i(k)\in\check{\mathcal{S}},\ \forall\widetilde{\mathbf{x}}_i(k)\in\mathcal{S}_{\widetilde{\mathbf{x}}},\ \forall\mathbf{w}_i(k)\in\mathcal{W}.
$$

Taking into account that $\check{\mathbf{x}}_i(k+1)$ is convex with respect to $\widetilde{\mathbf{x}}_i(k)$ and $\mathbf{w}_i(k)$, by applying the S-procedure (Theorem 2.1), the previous equation is satisfied if there exists a real scalar $\tau\geq 0$ such that:

$$
\begin{aligned}
&((\boldsymbol{A}+\boldsymbol{B}\boldsymbol{K})\check{\mathbf{x}}_i(k)+\boldsymbol{L}\boldsymbol{C}\widetilde{\mathbf{x}}+\boldsymbol{L}\mathbf{w})^\top\check{\boldsymbol{P}}((\boldsymbol{A}+\boldsymbol{B}\boldsymbol{K})\check{\mathbf{x}}_i(k)+\boldsymbol{L}\boldsymbol{C}\widetilde{\mathbf{x}}+\boldsymbol{L}\mathbf{w}) \\
&\qquad\qquad +\tau\Big(1-\check{\mathbf{x}}_i(k)^\top\check{\boldsymbol{P}}\check{\mathbf{x}}_i(k)\Big)\leq 1,\ \forall\widetilde{\mathbf{x}}\in\mathrm{vert}(\mathcal{S}_{\widetilde{\mathbf{x}}}),\ \forall\mathbf{w}\in\mathrm{vert}(\mathcal{W})
\end{aligned}
$$

where $\mathrm{vert}(\mathcal{S}_{\widetilde{\mathbf{x}}})$ and $\mathrm{vert}(\mathcal{W})$ are the sets of all vertices of the sets $\mathcal{S}_{\widetilde{\mathbf{x}}}$ and $\mathcal{W}$, respectively. Such an inequality can be rewritten:

$$
\begin{bmatrix}\check{\mathbf{x}}_i(k)\\1\end{bmatrix}^\top\begin{bmatrix}\tau\check{\boldsymbol{P}}-\boldsymbol{A}_{\boldsymbol{K}}^\top\check{\boldsymbol{P}}\boldsymbol{A}_{\boldsymbol{K}} & -\boldsymbol{A}_{\boldsymbol{K}}^\top\check{\boldsymbol{P}}\boldsymbol{L}(\boldsymbol{C}\widetilde{\mathbf{x}}+\mathbf{w}) \\ -(\boldsymbol{C}\widetilde{\mathbf{x}}+\mathbf{w})^\top\boldsymbol{L}^\top\check{\boldsymbol{P}}\boldsymbol{A}_{\boldsymbol{K}} & 1-\tau-(\boldsymbol{C}\widetilde{\mathbf{x}}+\mathbf{w})^\top\boldsymbol{L}^\top\check{\boldsymbol{P}}\boldsymbol{L}(\boldsymbol{C}\widetilde{\mathbf{x}}+\mathbf{w})\end{bmatrix}\begin{bmatrix}\check{\mathbf{x}}_i(k)\\1\end{bmatrix}\geq 0
\tag{4.17}
$$

for all $\widetilde{\mathbf{x}}\in\mathrm{vert}(\mathcal{S}_{\widetilde{\mathbf{x}}})$ and $\mathbf{w}\in\mathrm{vert}(\mathcal{W})$, where $\boldsymbol{A}_{\boldsymbol{K}}=\boldsymbol{A}+\boldsymbol{B}\boldsymbol{K}$. The matrix appearing in (4.17) can be rewritten $\forall\check{\mathbf{x}}_i(k)\in\check{\mathcal{S}}$, $\forall\widetilde{\mathbf{x}}\in\mathrm{vert}(\mathcal{S}_{\widetilde{\mathbf{x}}})$, $\forall\mathbf{w}\in\mathrm{vert}(\mathcal{W})$, as the BMI:

$$
\begin{bmatrix}\tau\check{\boldsymbol{P}}-\boldsymbol{A}_{\boldsymbol{K}}^\top\check{\boldsymbol{P}}\boldsymbol{A}_{\boldsymbol{K}} & -\boldsymbol{A}_{\boldsymbol{K}}^\top\check{\boldsymbol{P}}\boldsymbol{L}(\boldsymbol{C}\widetilde{\mathbf{x}}+\mathbf{w}) \\ -(\boldsymbol{C}\widetilde{\mathbf{x}}+\mathbf{w})^\top\boldsymbol{L}^\top\check{\boldsymbol{P}}\boldsymbol{A}_{\boldsymbol{K}} & 1-\tau-(\boldsymbol{C}\widetilde{\mathbf{x}}+\mathbf{w})^\top\boldsymbol{L}^\top\check{\boldsymbol{P}}\boldsymbol{L}(\boldsymbol{C}\widetilde{\mathbf{x}}+\mathbf{w})\end{bmatrix}\succeq 0
$$

which is equivalent, by using the Schur complement (2.7), to:

$$\begin{bmatrix} \tau\breve{\boldsymbol{P}} & \mathbf{0}_{n\times 1} & \boldsymbol{A}_{\boldsymbol{K}}^{\top} \\ \mathbf{0}_{1\times n} & 1-\tau & (\boldsymbol{C}\widetilde{\mathbf{x}}+\mathbf{w})^{\top}\boldsymbol{L}^{\top} \\ \boldsymbol{A}_{\boldsymbol{K}} & \boldsymbol{L}(\boldsymbol{C}\widetilde{\mathbf{x}}+\mathbf{w}) & \breve{\boldsymbol{P}}^{-1} \end{bmatrix} \succeq 0,\ \forall \widetilde{\mathbf{x}} \in \mathrm{vert}(\mathcal{S}_{\widetilde{\mathbf{x}}}),\ \forall \mathbf{w} \in \mathrm{vert}(\mathcal{W})$$

which can be decomposed, using the Schur complement (2.6) $\forall \widetilde{\mathbf{x}} \in \mathrm{vert}(\mathcal{S}_{\widetilde{\mathbf{x}}})$, $\forall \mathbf{w} \in \mathrm{vert}(\mathcal{W})$, into:

$$\begin{bmatrix} 1-\tau & (\boldsymbol{C}\widetilde{\mathbf{x}}+\mathbf{w})^{\top}\boldsymbol{L}^{\top} \\ \boldsymbol{L}(\boldsymbol{C}\widetilde{\mathbf{x}}+\mathbf{w}) & \breve{\boldsymbol{P}}^{-1} \end{bmatrix} - \begin{bmatrix} \mathbf{0}_{1\times n} \\ \boldsymbol{A}_{\boldsymbol{K}}\breve{\boldsymbol{P}}^{-1} \end{bmatrix} \frac{1}{\tau}\breve{\boldsymbol{P}} \begin{bmatrix} \mathbf{0}_{n\times 1} & \breve{\boldsymbol{P}}^{-1}\boldsymbol{A}_{\boldsymbol{K}}^{\top} \end{bmatrix} \succeq 0$$

which is equivalent, by using the Schur complement (2.6), to:

$$\begin{bmatrix} \tau\breve{\boldsymbol{P}}^{-1} & \mathbf{0}_{n\times 1} & \breve{\boldsymbol{P}}^{-1}\boldsymbol{A}_{\boldsymbol{K}}^{\top} \\ \mathbf{0}_{1\times n} & 1-\tau & (\boldsymbol{C}\widetilde{\mathbf{x}}+\mathbf{w})^{\top}\boldsymbol{L}^{\top} \\ \boldsymbol{A}_{\boldsymbol{K}}\breve{\boldsymbol{P}}^{-1} & \boldsymbol{L}(\boldsymbol{C}\widetilde{\mathbf{x}}+\mathbf{w}) & \breve{\boldsymbol{P}}^{-1} \end{bmatrix} \succeq 0,\ \forall \widetilde{\mathbf{x}} \in \mathrm{vert}(\mathcal{S}_{\widetilde{\mathbf{x}}}),\ \forall \mathbf{w} \in \mathrm{vert}(\mathcal{W}).$$

Thus, the inequality (4.17) is equivalent to the BMI:

$$\begin{bmatrix} \tau\breve{\boldsymbol{Z}} & \star & \star \\ \mathbf{0}_{1\times n} & 1-\tau & \star \\ \boldsymbol{A}\breve{\boldsymbol{Z}}+\boldsymbol{B}\breve{\boldsymbol{Y}} & \boldsymbol{L}(\boldsymbol{C}\widetilde{\mathbf{x}}+\mathbf{w}) & \breve{\boldsymbol{Z}} \end{bmatrix} \succeq 0,\ \forall \widetilde{\mathbf{x}} \in \mathrm{vert}(\mathcal{S}_{\widetilde{\mathbf{x}}}),\ \forall \mathbf{w} \in \mathrm{vert}(\mathcal{W}) \tag{4.18}$$

with the decision variables $\tau \in \mathbb{R}_+$, $\breve{\boldsymbol{Z}} = \breve{\boldsymbol{P}}^{-1} \in \mathbb{R}^{n\times n}$ and $\breve{\boldsymbol{Y}} = \boldsymbol{K}\breve{\boldsymbol{Z}} \in \mathbb{R}^{m\times n}$. As for (4.14), the attentive reader notices that in (4.18), the only bilinear term is the product between the scalar τ and the matrix $\breve{\boldsymbol{Z}}$. Then, the BMI (4.18) can be relaxed into a LMI if τ is fixed arbitrarily.

Using the same measure of the size of $\breve{\mathcal{S}}$ as for $\widetilde{\mathcal{S}}$, minimizing the size of $\breve{\mathcal{S}}$ consists in minimizing the value of a parameter $\breve{\eta}$ subject to the constraint $\breve{\mathcal{S}} \subseteq \sqrt{\breve{\eta}}\mathcal{X}$ that can be written, by applying the Schur complement (2.7), as the LMI:

$$\begin{bmatrix} \breve{\eta}\theta_{\mathcal{X},j}^2 & \boldsymbol{h}_{\mathcal{X},j}\breve{\boldsymbol{P}}^{-1} \\ \breve{\boldsymbol{P}}^{-1}\boldsymbol{h}_{\mathcal{X},j}^{\top} & \breve{\boldsymbol{P}}^{-1} \end{bmatrix} \succeq 0,\ \forall j \in \overline{1, r_x} \tag{4.19}$$

or:

$$\begin{bmatrix} \breve{\eta}\theta_{\mathcal{X},j}^2 & \star \\ \breve{\boldsymbol{Z}}\boldsymbol{h}_{\mathcal{X},j}^{\top} & \breve{\boldsymbol{Z}} \end{bmatrix} \succeq 0,\ \forall j \in \overline{1, r_x} \tag{4.20}$$

where $\breve{\boldsymbol{Z}}$ is defined as for (4.18).

For the tuning of the control gain matrix $\boldsymbol{K}$, Limón et al. (2010) proposes a last constraint. Indeed, this gain $\boldsymbol{K}$ has an impact on the size of the input constraint set $\breve{\mathcal{U}} = \mathcal{U} \ominus \boldsymbol{K}\mathcal{S}_{\breve{\mathbf{x}}}$ in the MPC problem (4.9). Thus, a constraint to restrict the size of the set $\boldsymbol{K}\mathcal{S}_{\breve{\mathbf{x}}}$ of admissible control inputs for the state-feedback part of the controller would be useful to increase the control range of its MPC part. By denoting $\boldsymbol{H}_{\mathcal{U}} \in \mathbb{R}^{r_u\times m}$ and $\boldsymbol{\theta}_{\mathcal{U}} \in \mathbb{R}^{r_u}$ the matrices inducing $\mathcal{U}$ in $\mathbb{R}^m$ and $\boldsymbol{h}_{\mathcal{U},j}$ and $\theta_{\mathcal{U},j}$, with $j \in \overline{1, r_u}$, their rows, such a constraint would have the form:

$$\boldsymbol{h}_{\mathcal{U},j}\boldsymbol{K}\mathbf{x} \leq \sqrt{\rho_j}\theta_{\mathcal{U},j},\ \forall j \in \overline{1, r_u},\ \forall \mathbf{x} \in \breve{\mathcal{S}}$$

where $\rho_j \in (0, 1]$ for all $j \in \overline{1, r_u}$, which can be rewritten as the LMI:

$$\begin{bmatrix} \rho_j \theta_{\mathcal{U},j}^2 & \boldsymbol{h}_{\mathcal{U},j} \boldsymbol{K} \\ \boldsymbol{K}^\top \boldsymbol{h}_{\mathcal{U},j}^\top & \breve{\boldsymbol{P}} \end{bmatrix} \succeq 0, \, \forall j \in \overline{1, r_u} \tag{4.21}$$

By applying the Schur complement (2.7), the LMI (4.21) is equivalent to the LMI:

$$\begin{bmatrix} \rho_j \theta_{\mathcal{U},j}^2 & \star \\ \breve{\boldsymbol{Y}}^\top \boldsymbol{h}_{\mathcal{U},j}^\top & \breve{\boldsymbol{Z}} \end{bmatrix} \succeq 0, \, \forall j \in \overline{1, r_u} \tag{4.22}$$

where $\breve{\boldsymbol{Z}}$ and $\breve{\boldsymbol{Y}}$ are defined as for (4.18).

With all these elements, the BMI constrained optimization problem to be solved to find the controller gain matrix $\boldsymbol{K}$ is:

$$\begin{aligned}
&\underset{\substack{\tau, \breve{\eta}, \breve{\boldsymbol{Z}}, \breve{\boldsymbol{Y}}, \\ \rho_j, \forall j \in \overline{1, r_u}}}{\text{minimize}} \quad \alpha \breve{\eta} + \beta \sum_{j=1}^{r} \rho_j \\
&\text{subject to} \\
&\begin{bmatrix} \tau \breve{\boldsymbol{Z}} & \star & \star \\ \boldsymbol{0}_{1 \times n} & 1 - \tau & \star \\ \boldsymbol{A} \breve{\boldsymbol{Z}} + \boldsymbol{B} \breve{\boldsymbol{Y}} & \boldsymbol{L}(\boldsymbol{C} \widetilde{\mathbf{x}} + \mathbf{w}) & \breve{\boldsymbol{Z}} \end{bmatrix} \succeq 0, \qquad \forall \widetilde{\mathbf{x}} \in \text{vert}(\mathcal{S}_{\widetilde{\mathbf{x}}}), \, \forall \mathbf{w} \in \text{vert}(\mathcal{W}), \\
&\qquad\qquad \begin{bmatrix} \breve{\eta} \theta_{\mathcal{X},j}^2 & \star \\ \breve{\boldsymbol{Z}} \boldsymbol{h}_{\mathcal{X},j}^\top & \breve{\boldsymbol{Z}} \end{bmatrix} \succeq 0, \qquad \forall j \in \overline{1, r_x}, \\
&\qquad\qquad \begin{bmatrix} \rho_j \theta_{\mathcal{U},j}^2 & \star \\ \breve{\boldsymbol{Y}}^\top \boldsymbol{h}_{\mathcal{U},j}^\top & \breve{\boldsymbol{Z}} \end{bmatrix} \succeq 0, \qquad \forall j \in \overline{1, r_u}, \\
&\qquad\qquad\qquad \rho_j \in (0, 1], \quad \forall j \in \overline{1, r_u}, \\
&\qquad\qquad\qquad \tau \geq 0, \\
&\qquad\qquad\qquad \breve{\eta} \in [0, 1)
\end{aligned} \tag{4.23}$$

with $\alpha, \beta \in \mathbb{R}_+$ two weights. This problem has $n^2/2 + (2m + 1)n/2 + r_u + 2$ scalar decision variables and $|\text{vert}(\mathcal{S}_{\widetilde{\mathbf{x}}})| \cdot |\text{vert}(\mathcal{W})| + 3r_u + r_x + 3$.

As for (4.16), the only bilinear term is the product between the scalar τ and the matrix $\breve{\boldsymbol{Z}}$. Then, the problem (4.23) can be relaxed into a LMI constrained problem if τ is fixed arbitrarily. Moreover, the values of the decision variables ρ_j, with $j \in \overline{1, r_u}$, can be chosen beforehand to impose a given behavior as in Limón et al. (2010). In this last case, the cost function of (4.23) is changed simply into $\breve{\eta}$.

By solving (4.23), it is then possible to obtain $\boldsymbol{K} = \breve{\boldsymbol{Y}} \breve{\boldsymbol{Z}}^{-1}$. Then, using $\boldsymbol{K}$ and the set $\boldsymbol{L}(\boldsymbol{C} \mathcal{S}_{\widetilde{\mathbf{x}}} \oplus \mathcal{W})$ containing the signal $\boldsymbol{L}(\boldsymbol{C} \widetilde{\mathbf{x}}_i(k) + \mathbf{w}_i(k))$, the same procedure used to obtain $\mathcal{S}_{\widetilde{\mathbf{x}}}$ is used for $\mathcal{S}_{\breve{\mathbf{x}}}$.

For both tuning procedures, no constraint relative to the output set has been formulated. This is due to the fact that such constraints would be linked to the time varying guaranteed Voronoi cells that are not available *a priori*. The tuning procedures presented above are meant to be run before using the gains and the invariant sets in the MPC problem (4.9). Thus, constraints on the output cannot be formulated to tune the observer and controller gain matrices $\boldsymbol{L}$ and $\boldsymbol{K}$.

As mentioned in Section 4.1.2, Mayne et al. (2006) show that, with the control input defined in (4.5), as long as $\widetilde{\mathbf{x}}_i(0) \in \mathcal{S}_{\widetilde{\mathbf{x}}}$ and $\breve{\mathbf{x}}_i(0) \in \mathcal{S}_{\breve{\mathbf{x}}}$, it is guaranteed that $\mathbf{x}_i(k) \in \{\breve{\mathbf{x}}_i(k)\} \oplus \mathcal{S}_{\widetilde{\mathbf{x}}} \oplus \mathcal{S}_{\breve{\mathbf{x}}}$ for all $k \geq 0$. Then, the set $\mathcal{S}_{\widetilde{\mathbf{x}}} \oplus \mathcal{S}_{\breve{\mathbf{x}}}$ defines the envelope of a tube centered on the trajectory $\breve{\mathbf{x}}_i(k)$ in which the state $\mathbf{x}_i(k)$ of the agent $i \in \overline{1, N}$ evolves.

4.1.4 Proposed deployment results for MAS with bounded perturbations

In this paragraph the tube-based MPC strategy proposed in Section 4.1.3 is applied to two multi-vehicle systems. These two MVS are composed of vehicles obeying single integrator dynamics for the first case and of quadrotor UAVs as described in Section 3.3 for the second case.

4.1.4.1 Single integrator dynamics

Let Σ be a multi-vehicle system composed of $N = 6$ agents deployed into the output space:

$$\mathcal{Y} = \mathcal{X} = \left\{ \mathbf{x} \in \mathbb{R}^2 \;\middle|\; \begin{bmatrix} -1 & 4 \\ -3 & -2 \\ 3 & -7 \\ 5 & 1 \end{bmatrix} \mathbf{x} \leq \begin{bmatrix} 24 \\ 20 \\ 44 \\ 48 \end{bmatrix} \right\}. \tag{4.24}$$

Each agent $i \in \overline{1, N}$ obeys the discrete-time single integrator dynamics:

$$\begin{aligned} \mathbf{x}_i(k+1) &= \mathbf{x}_i(k) + T_s \mathbf{u}_i(k) + \mathbf{d}_i(k) \\ \mathbf{y}_i(k) &= \mathbf{x}_i(k) + \mathbf{w}_i(k) \end{aligned} \tag{4.25}$$

where $\mathbf{x}_i(k) \in \mathbb{R}^2$, $\mathbf{u}_i(k) \in \mathcal{U}$ with the input set $\mathcal{U} = \mathbb{B}^2(2 \cdot \mathbf{1}_{2\times 1})$, $\mathbf{d}_i(k) \in \mathcal{D}$ with the input perturbation set $\mathcal{D} = \mathbb{B}^2(0.05 \cdot \mathbf{1}_{2\times 1})$ and $\mathbf{w}_i(k) \in \mathcal{W}$ with the output perturbation set $\mathcal{W} = \mathbb{B}^2(0.1 \cdot \mathbf{1}_{2\times 1})$. A value of $T_s = 0.2\,\mathrm{s}$ is chosen for the sampling period.

Using the notations from (4.1), in (4.25), $\boldsymbol{A} = \boldsymbol{C} = \boldsymbol{I}_2$ and $\boldsymbol{B} = T_s \boldsymbol{I}_2$. As stated in Raković et al. (2005), the mRPI set for the dynamics (4.4) is defined as:

$$\widetilde{\mathcal{S}}_\infty = \bigoplus_{j=0}^{+\infty} (\boldsymbol{A} - \boldsymbol{L}\boldsymbol{C})^j (\mathcal{D} \oplus (-\boldsymbol{L}\mathcal{W})).$$

Given the simplicity of $\boldsymbol{A}$ and $\boldsymbol{C}$, it is immediate that $\widetilde{\mathcal{S}}_\infty$ is finitely determined and as small as possible if $\boldsymbol{L} = \boldsymbol{I}_2$. Moreover, with $\boldsymbol{L} = \boldsymbol{I}_2$, the dynamics (4.4) is stable. Using $\boldsymbol{L} = \boldsymbol{I}_2$ for the observer gain as in Chevet et al. (2019), $(\boldsymbol{A} - \boldsymbol{L}\boldsymbol{C})^j = \mathbf{0}_2$ for all $j > 0$ and $(\boldsymbol{A} - \boldsymbol{L}\boldsymbol{C})^0 = \boldsymbol{I}_2$, thus:

$$\mathcal{S}_{\widetilde{\mathbf{x}}} = \widetilde{\mathcal{S}}_\infty = \mathcal{D} \oplus (-\boldsymbol{I}_2 \mathcal{W}). \tag{4.26}$$

Remark 4.4: Value of the observer gain matrix
With the tuning procedure described in Paragraph 4.1.3.3 for the dynamics (4.25), the optimization problem (4.16) also gives $\boldsymbol{L} = \boldsymbol{I}_2$. ♢

Solving the BMI constrained optimization problem (4.23), it is then possible to obtain the control gain matrix $\boldsymbol{K}$. With the weights $\alpha = \beta = 1$, the control gain matrix obtained with the optimization problem (4.23) is then:

$$\boldsymbol{K} = -5\boldsymbol{I}_2.$$

Remark 4.5: Value of the control gain matrix
The arguments used to chose the value of $\boldsymbol{L}$ could have been used here to chose $\boldsymbol{K} = -5\boldsymbol{I}_2$ such that $\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K} = \boldsymbol{0}_2$ to have a finitely determined mRPI set $\breve{\mathcal{S}}_\infty$. $\Diamond$

As for the dynamics (4.4), the mRPI set for the dynamics (4.6) is defined as:

$$\breve{\mathcal{S}}_\infty = \bigoplus_{j=0}^{+\infty} (\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K})^j \boldsymbol{L}(\boldsymbol{C}\mathcal{S}_{\tilde{\mathbf{x}}} \oplus \mathcal{W})$$

Since the gain matrix $\boldsymbol{K}$ obtained via the tuning procedure is such that $\boldsymbol{A}+\boldsymbol{B}\boldsymbol{K} = \boldsymbol{0}_2$, the mRPI set $\mathcal{S}_{\breve{\mathbf{x}}}$ for the dynamics (4.6) is:

$$\mathcal{S}_{\breve{\mathbf{x}}} = \breve{\mathcal{S}}_\infty = \mathcal{S}_{\tilde{\mathbf{x}}} \oplus \mathcal{W}. \tag{4.27}$$

The sets $\mathcal{S}_{\tilde{\mathbf{x}}}$, $\mathcal{S}_{\breve{\mathbf{x}}}$ and $\boldsymbol{K}\mathcal{S}_{\breve{\mathbf{x}}}$ are presented in Figure 4.1.

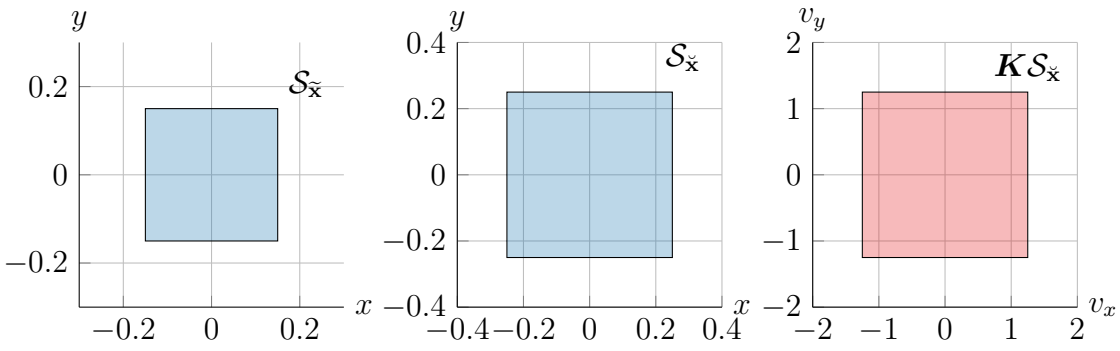


Figure 4.1: Invariant sets for the tube-based MPC controller in the single integrator dynamics case.

The nominal input $\breve{\mathbf{u}}_i(k)$ from (4.5) is computed by solving the optimization problem (4.9). As in Section 3.2.4, the weighting matrices are chosen such that $\boldsymbol{Q} = \boldsymbol{R} = \boldsymbol{I}_2$ and $\boldsymbol{P}$ is the solution of the algebraic Riccati equation:

$$\boldsymbol{A}^\top \boldsymbol{P}\boldsymbol{A} - \boldsymbol{P} - \boldsymbol{A}^\top \boldsymbol{P}\boldsymbol{B}\big(\boldsymbol{B}^\top \boldsymbol{P}\boldsymbol{B} + \boldsymbol{R}\big)^{-1}\boldsymbol{B}^\top \boldsymbol{P}\boldsymbol{A} + \boldsymbol{Q} = \boldsymbol{0}_2$$

with $\boldsymbol{A} = \boldsymbol{I}_2$ and $\boldsymbol{B} = T_s\boldsymbol{I}_2$. The prediction horizon is chosen to be $N_p = 10$. The scaling factor used in the terminal constraint is chosen to be $\lambda_i = 0.9$, with $i \in \overline{1, N}$.

The initial value of $\breve{\mathbf{x}}_i(0)$ for all $i \in \overline{1, N}$ is chosen randomly such that the sets $\{\breve{\mathbf{x}}_i(0)\} \oplus \mathcal{W}$ do not overlap. The initial value of the estimated state $\widehat{\mathbf{x}}_i$ for all $i \in \overline{1, N}$ is initialized such that $\widehat{\mathbf{x}}_i(0) = \breve{\mathbf{x}}_i(0)$ and the state $\mathbf{x}_i$ is initialized to a random value in $\{\breve{\mathbf{x}}_i(0)\} \oplus \mathcal{S}_{\tilde{\mathbf{x}}}$. During the entire simulation, the input and output perturbation signals $\mathbf{d}_i(k)$ and $\mathbf{w}_i(k)$ take random values in $\mathcal{D}$ and $\mathcal{W}$ for all $i \in \overline{1, N}$, this value changing every 10 s.

The initial configuration of the MAS Σ in $\mathcal{Y}$ is displayed in Figure 4.2. The initial nominal position $\breve{\mathbf{x}}_i(0)$, with $i \in \overline{1, N}$, of the agents is represented by circles,

the real position of the agents $\mathbf{x}_i(0)$, by diamonds and the initial Chebyshev center position $\mathbf{c}_i(0)$, by stars. In addition to the guaranteed Voronoi tessellation of the MAS, are also presented the output constraint sets $\breve{\mathcal{V}}_i^g(0)$ for the MPC problem (4.9) with dashed frontiers. The sets $\{\breve{\mathbf{x}}_i(0)\} \oplus \mathcal{S}_{\tilde{\mathbf{x}}} \oplus \mathcal{S}_{\breve{\mathbf{x}}}$ are shown to ensure that $\mathbf{x}_i(0)$ is correctly initialized.

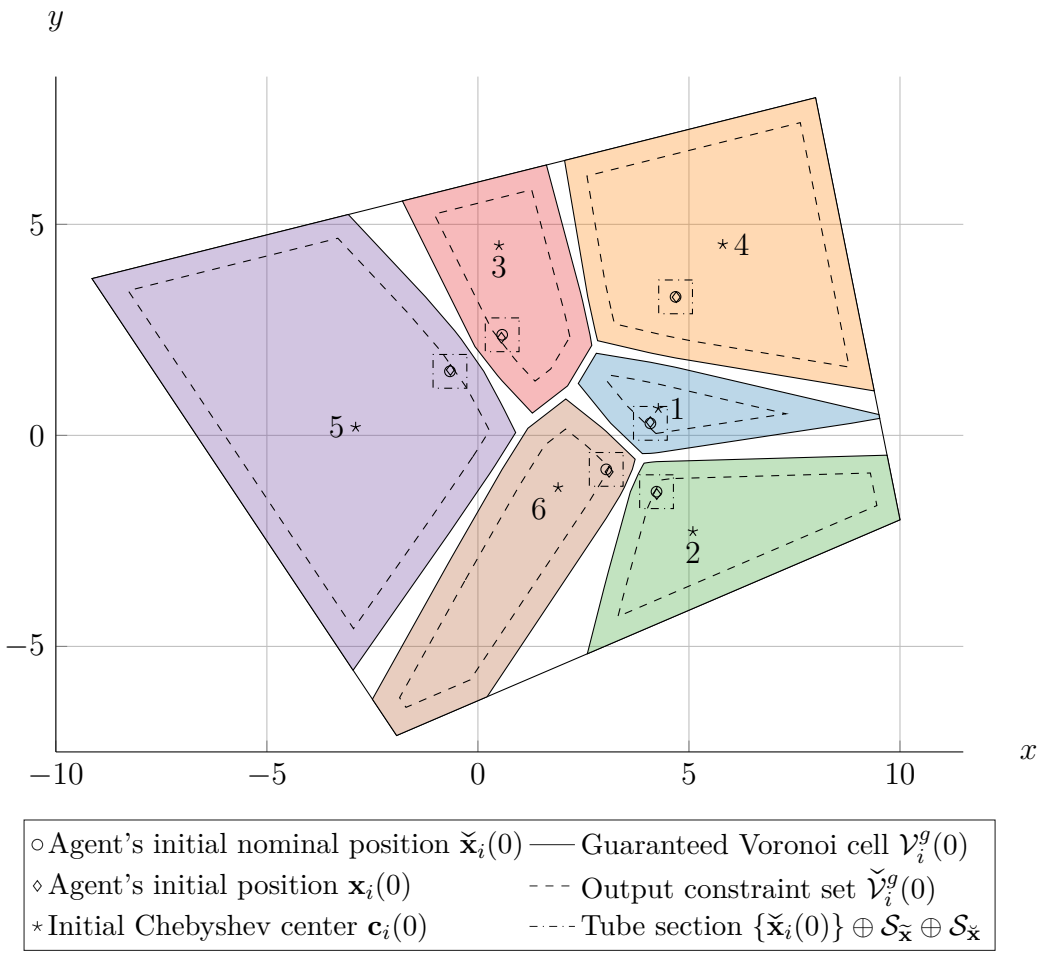


Figure 4.2: Initial position of the MVS Σ in the output space.

The agents of Σ then follow their Chebyshev centers inside $\mathcal{Y}$. The final configuration is shown in Figure 4.3 with the same elements as in Figure 4.2. It is obvious from Figure 4.3 that the nominal position $\breve{\mathbf{x}}_i$, with $i \in \overline{1, N}$ has reached the Chebyshev center $\mathbf{c}_i$ of its guaranteed Voronoi cell $\mathcal{V}_i^g$. The real position $\mathbf{x}_i$, with $i \in \overline{1, N}$, of the agents is guaranteed to belong to the mRPI set centered on the nominal position $\breve{\mathbf{x}}_i$ drawn with dash dotted frontiers.

Finally, Figure 4.4 presents the trajectory of the Chebyshev centers $\mathbf{c}_i(k)$ and of the nominal positions $\breve{\mathbf{x}}_i(k)$ over the entire simulation. This confirms the fact that the nominal position $\breve{\mathbf{x}}_i$ converges to its objective $\mathbf{c}_i$.

Another measure of this convergence is presented in Figure 4.5. This figure presents the distance over time between the Chebyshev center and the nominal position such that:

$$d_i(k) = \|\breve{\mathbf{x}}_i(k) - \mathbf{c}_i(k)\|_2$$

for all $i \in \overline{1, N}$. As expected from the nominal case, the distances converge to 0. The plot is cut after $t = 25\,\mathrm{s}$ since the distance stays at zero for the remainder of

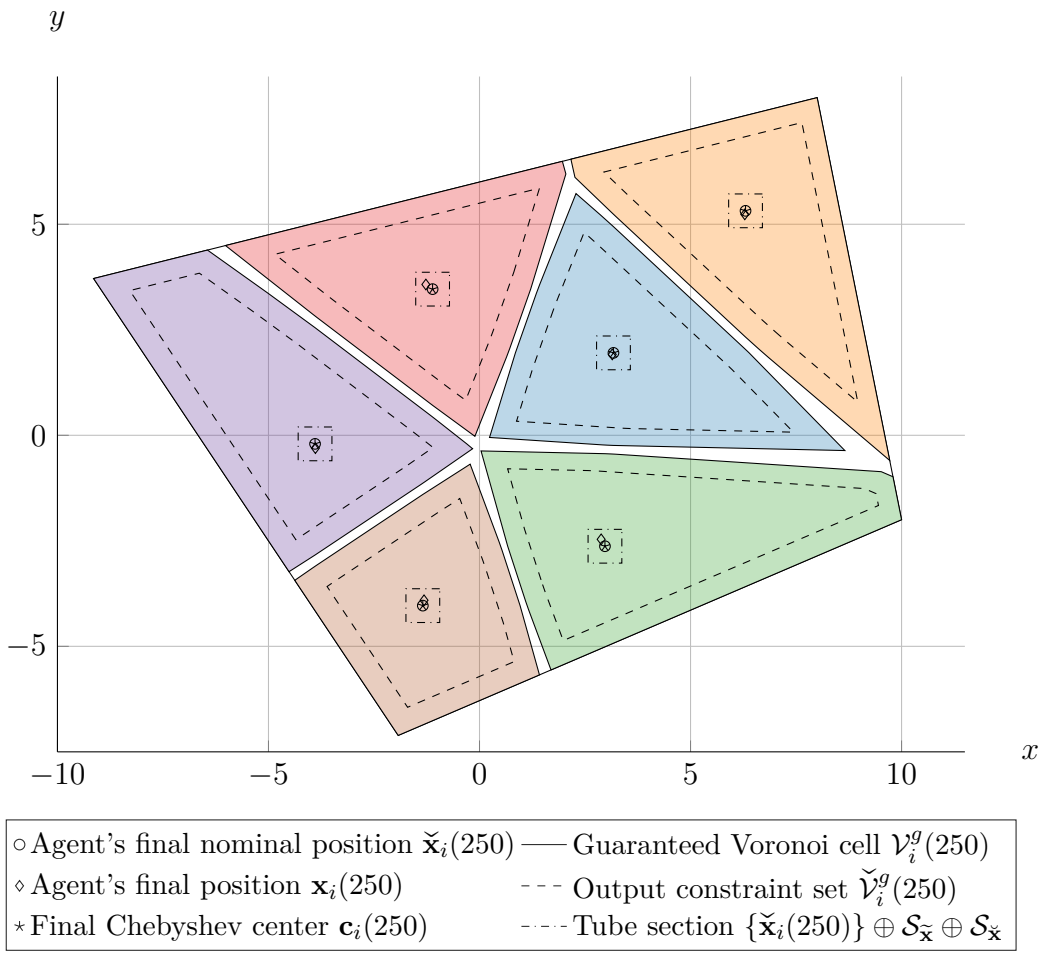


Figure 4.3: Final position of the MVS Σ in the output space at $t = 50\,\mathrm{s}$.

the simulation.

An interesting measure of the efficiency of the proposed control algorithm is the norm of the estimation error $\widetilde{\mathbf{x}}_i$ as well as the norm of the deviation error $\breve{\mathbf{x}}_i$. Indeed, these two quantities quantify the fact that the state of the agent $i \in \overline{1, N}$ remains inside the invariant set centered on $\breve{\mathbf{x}}_i(k)$ and bounded by $\mathcal{S}_{\widetilde{\mathbf{x}}} \oplus \mathcal{S}_{\breve{\mathbf{x}}}$. Figure 4.6 and Figure 4.7 show the evolution of these two norms over time. While the signals in Figure 4.6 might be difficult to understand, one element is of importance. Indeed, the norm remains bounded by the maximum norm of the vectors of $\mathcal{S}_{\widetilde{\mathbf{x}}}$, i.e. $\|\widetilde{\mathbf{x}}_{\max}\|_2 = \max_{\mathbf{x} \in \mathcal{S}_{\widetilde{\mathbf{x}}}} \|\mathbf{x}\|_2$. Given the shape of $\mathcal{S}_{\widetilde{\mathbf{x}}}$, the maximum norm is $\|\widetilde{\mathbf{x}}_{\max}\|_2 = 0.21\,\mathrm{m}$. Every plot in Figure 4.6 is below the value of $\|\widetilde{\mathbf{x}}_{\max}\|_2$. The shape of the plots comes from the fact that, given (4.4), the estimation error converges quickly to $\mathbf{d}_i(k) - \boldsymbol{L}\mathbf{w}_i(k)$. Since the perturbation signals $\mathbf{d}_i(k)$ and $\mathbf{w}_i(k)$ take random values every 10 s, the norm of the estimation error $\|\widetilde{\mathbf{x}}_i(k)\|_2$ converges to the norm $\|\mathbf{d}_i(k) - \boldsymbol{L}\mathbf{w}_i(k)\|_2$. The same remark concerns Figure 4.7: every plot remains below the maximum norm of the vectors of $\mathcal{S}_{\breve{\mathbf{x}}}$, i.e. $\|\breve{\mathbf{x}}_{\max}\|_2 = \max_{\mathbf{x} \in \mathcal{S}_{\breve{\mathbf{x}}}} \|\mathbf{x}\|_2$. Given the shape of $\mathcal{S}_{\breve{\mathbf{x}}}$, the maximum norm is $\|\breve{\mathbf{x}}_{\max}\|_2 = 0.35\,\mathrm{m}$, which is indeed an upper bound of the plots in Figure 4.7. The pikes that appear in this figure occur for every change of the perturbation signals $\mathbf{d}_i(k)$ and $\mathbf{w}_i(k)$. After every change, the norm of the deviation error $\|\breve{\mathbf{x}}_i(k)\|_2$ converges to $\|\boldsymbol{LC}\widetilde{\mathbf{x}}_i(k) + \boldsymbol{L}\mathbf{w}_i(k)\|_2$ since the control gain matrix $\boldsymbol{K}$ is chosen such that the dynamics (4.6) is stable.

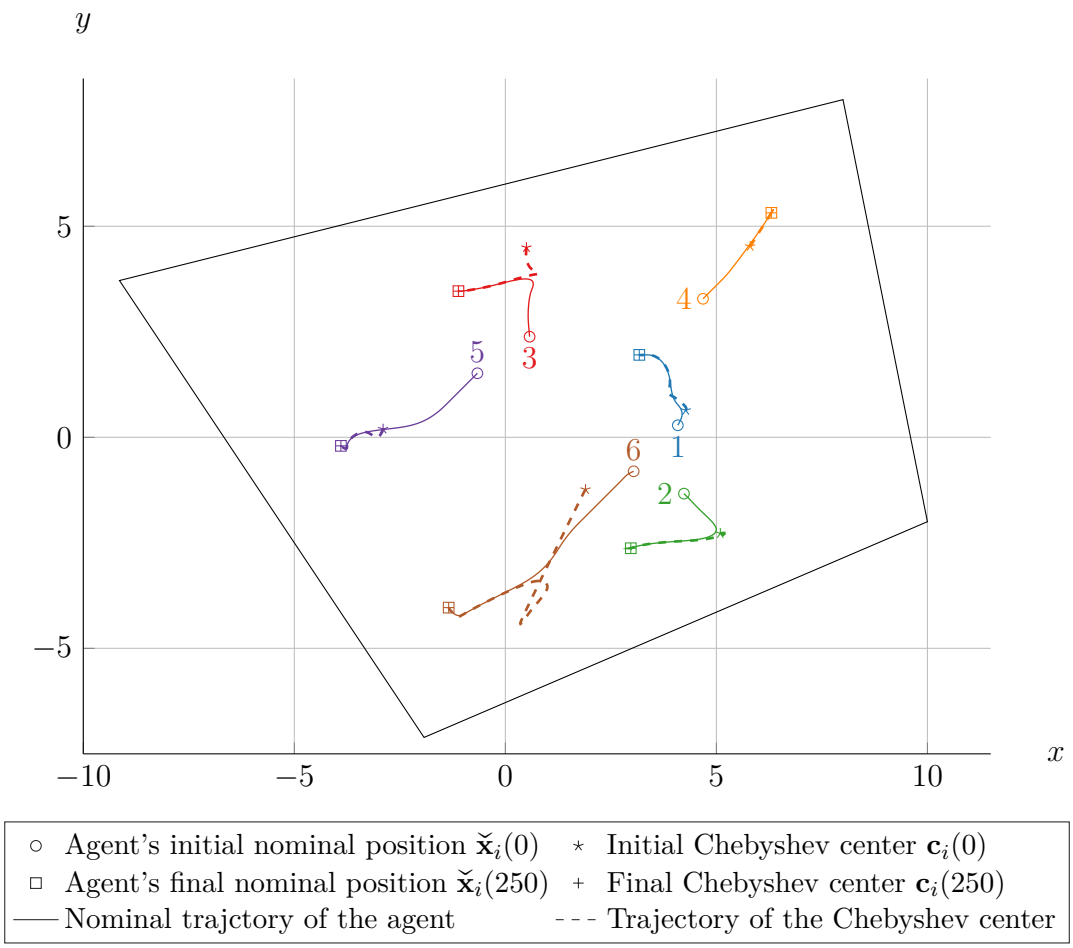


Figure 4.4: Nominal trajectories of the agents of Σ and their associated Chebyshev centers.

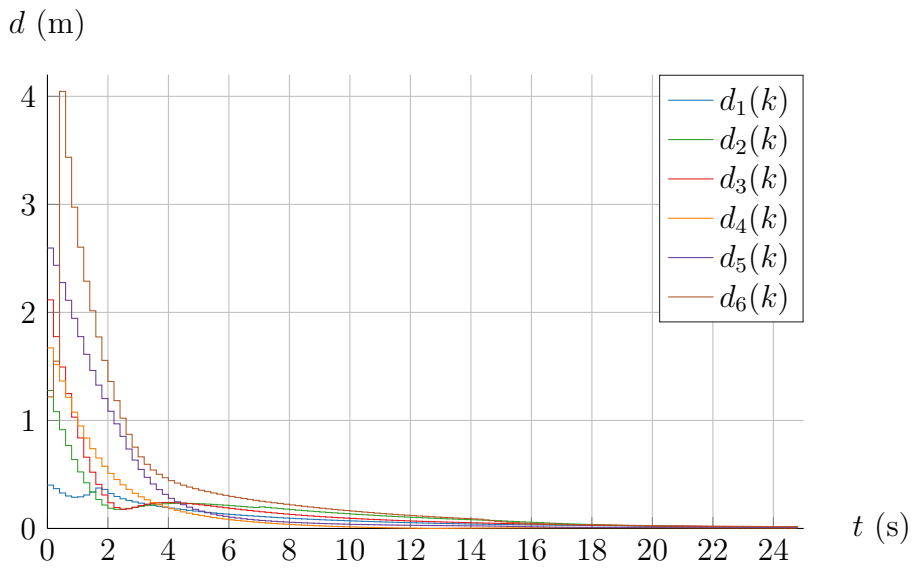


Figure 4.5: Distance of the nominal position of each agent of Σ to its Chebyshev center over time.

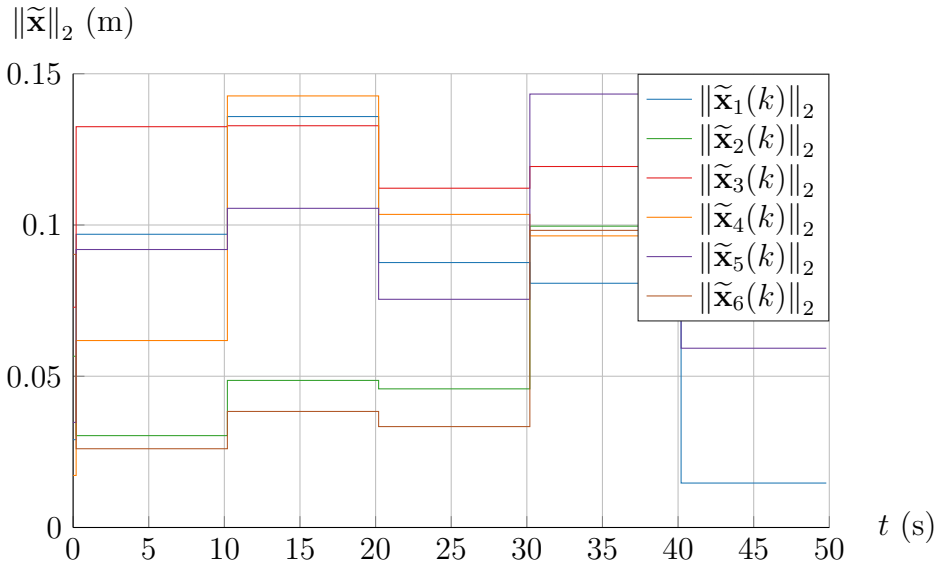


Figure 4.6: Norm of the estimation error of each agent of Σ.

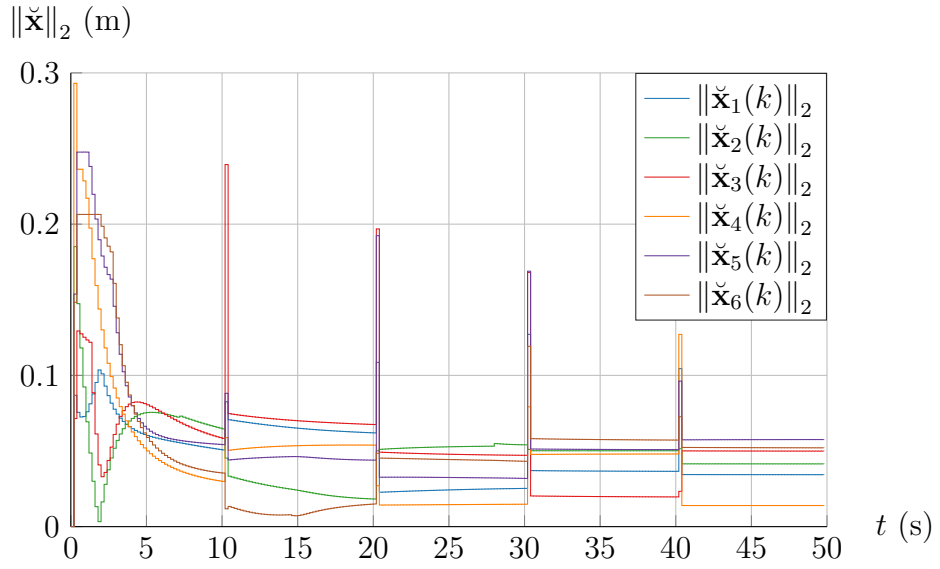


Figure 4.7: Norm of the deviation error of each agent of Σ.

Since there is no offset cancellation strategy (Limón et al., 2010), the estimation and deviation errors never go back to zero. Indeed, the goal of the presented control strategy is to drive the MVS to a static Chebyshev configuration without collisions despite the presence of bounded perturbations on the agents' dynamics. Then, the control strategy proves to be efficient since the MVS is deployed into a static Chebyshev configuration. Moreover, the state of the agents, despite the perturbations, is guaranteed to remain in a given set known beforehand which, coupled with the guaranteed Voronoi tessellation, guarantees, by construction, that no collision can occur between two vehicles.

4.1.4.2 UAV dynamics

Let Σ be a multi-vehicle system composed of $N = 6$ agents deployed in the output space:

$$\mathcal{Y} = \mathbb{B}^2(10 \cdot \mathbf{1}_{2\times 1}). \tag{4.28}$$

Each agent $i \in \overline{1, N}$ is a quadrotor UAV obeying the dynamics and the control strategy presented in Section 3.3. The predictive controller used for the outer loop in Paragraph 3.3.2.3 is replaced by the tube-based MPC described in Section 4.1.3.

The dynamics (3.31) is replaced by:

$$\begin{aligned} \mathbf{x}_i(k+1) &= \boldsymbol{A}\mathbf{x}_i(k) + \boldsymbol{B}\mathbf{u}_i(k) + \mathbf{d}_i(k) \\ \mathbf{y}_i(k) &= \boldsymbol{C}\mathbf{x}_i(k) + \mathbf{w}_i(k) \end{aligned} \tag{4.29}$$

where $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ are defined in (3.39), $\mathbf{u}_i(k) \in \mathcal{U} = \mathbb{B}^2\left(\frac{\pi}{6} \cdot \mathbf{1}_{2\times 1}\right)$, $\mathbf{d}_i(k) \in \mathcal{D} = \mathbb{B}^4\left(10^{-2} \cdot \begin{bmatrix} 0.5 & 2 & 0.5 & 2 \end{bmatrix}^\top\right)$, $\mathbf{w}_i(k) \in \mathcal{W} = \mathbb{B}^2(0.01 \cdot \mathbf{1}_{2\times 1})$ and $\mathbf{x}_i(k) \in \mathcal{X} = \mathbb{B}^4\left(\begin{bmatrix} 10 & 3 & 10 & 3 \end{bmatrix}^\top\right)$. The definition of the sets $\mathcal{D}$ and $\mathcal{W}$ is based on Michel et al. (2019).

The tuning procedures for the observer and controller gain matrices $\boldsymbol{L}$ and $\boldsymbol{K}$ are computationally heavy if the four state dynamics (4.29) is used. Given the state matrices (3.39), the position subsystem (4.29) can be separated into two independent subsystems on which the tuning procedures are applied. These two subsystems are:

$$\begin{aligned} \begin{bmatrix} x_i(k+1) \\ v_{x,i}(k+1) \end{bmatrix} &= \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_i(k) \\ v_{x,i}(k) \end{bmatrix} + \frac{1}{2} g T_s \begin{bmatrix} T_s \\ 2 \end{bmatrix} \theta_i(k) + \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \mathbf{d}_i(k) \\ x_i(k) &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_i(k) \\ v_{x,i}(k) \end{bmatrix} + \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{w}_i(k) \end{aligned} \tag{4.30}$$

and:

$$\begin{aligned} \begin{bmatrix} y_i(k+1) \\ v_{y,i}(k+1) \end{bmatrix} &= \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \begin{bmatrix} y_i(k) \\ v_{y,i}(k) \end{bmatrix} - \frac{1}{2} g T_s \begin{bmatrix} T_s \\ 2 \end{bmatrix} \phi_i(k) + \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{d}_i(k) \\ y_i(k) &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} y_i(k) \\ v_{y,i}(k) \end{bmatrix} + \begin{bmatrix} 0 & 1 \end{bmatrix} \mathbf{w}_i(k) \end{aligned} \tag{4.31}$$

with $g = 9.81\,\mathrm{m \cdot s^{-2}}$ and $T_s = 0.2\,\mathrm{s}$. With the decomposition given in (4.30) and (4.31), the tuning procedures are run two times to obtain the gain matrices $\boldsymbol{L}_x$ and $\boldsymbol{K}_x$ for (4.30) and $\boldsymbol{L}_y$, $\boldsymbol{K}_y$ for (4.31) such that:

$$\boldsymbol{L} = \begin{bmatrix} \boldsymbol{L}_x & \mathbf{0}_{2\times 1} \\ \mathbf{0}_{2\times 1} & \boldsymbol{L}_y \end{bmatrix} \quad \text{and} \quad \boldsymbol{K} = \begin{bmatrix} \mathbf{0}_{1\times 2} & \boldsymbol{K}_y \\ \boldsymbol{K}_x & \mathbf{0}_{1\times 2} \end{bmatrix}.$$

Then, using these gains, it is possible to obtain the sets $\mathcal{S}_{\tilde{\mathbf{x}},x}$ and $\mathcal{S}_{\check{\mathbf{x}},x}$, approximations of the mRPI sets for the estimation error and deviation error of the subsystem (4.30) and the sets $\mathcal{S}_{\tilde{\mathbf{x}},y}$ and $\mathcal{S}_{\check{\mathbf{x}},y}$, approximations of the mRPI sets for the estimation error and deviation error of the subsystem (4.31) such that $\mathcal{S}_{\tilde{\mathbf{x}}} = \mathcal{S}_{\tilde{\mathbf{x}},x} \times \mathcal{S}_{\tilde{\mathbf{x}},y}$ and $\mathcal{S}_{\check{\mathbf{x}}} = \mathcal{S}_{\check{\mathbf{x}},x} \times \mathcal{S}_{\check{\mathbf{x}},y}$.

Solving the BMI constrained problem (4.16) for (4.30) and (4.31) by considering τ as a decision variable yields:

$$\boldsymbol{L}_x = \boldsymbol{L}_y = \begin{bmatrix} 0.9901 \\ 1.0482 \end{bmatrix}.$$

Then, Algorithm 2.1 is run to obtain $\mathcal{S}_{\tilde{\mathbf{x}},x}$ and $\mathcal{S}_{\tilde{\mathbf{x}},y}$ which are such that $\mathcal{S}_{\tilde{\mathbf{x}},x} = \mathcal{S}_{\tilde{\mathbf{x}},y}$. Let $\boldsymbol{H}_{\tilde{\mathbf{x}}}$ and $\boldsymbol{\theta}_{\tilde{\mathbf{x}}}$ be the matrices inducing $\mathcal{S}_{\tilde{\mathbf{x}},x}$ and $\mathcal{S}_{\tilde{\mathbf{x}},y}$ in $\mathbb{R}^2$. Then:

$$\mathcal{S}_{\tilde{\mathbf{x}}} = \left\{ \mathbf{x} \in \mathbb{R}^2 \;\middle|\; (\boldsymbol{I}_2 \otimes \boldsymbol{H}_{\tilde{\mathbf{x}}})\mathbf{x} \leq \mathbf{1}_{2\times 1} \otimes \boldsymbol{\theta}_{\tilde{\mathbf{x}}} \right\}.$$

The values of $\boldsymbol{H}_{\tilde{\mathbf{x}}}$ and $\boldsymbol{\theta}_{\tilde{\mathbf{x}}}$ are not reported here since $\mathcal{S}_{\tilde{\mathbf{x}},x}$ is composed of 43 inequalities. However, it is shown in Figure 4.8.

Solving the BMI constrained problem (4.23) for (4.30) and (4.31) by considering τ as a decision variable gives, with the weights $\alpha = 10$ and $\beta = 0.1$:

$$\boldsymbol{K}_x = -\boldsymbol{K}_y = \begin{bmatrix} -0.3628 & -0.4986 \end{bmatrix}.$$

Then, Algorithm 2.1 is run to obtain $\mathcal{S}_{\check{\mathbf{x}},x}$ and $\mathcal{S}_{\check{\mathbf{x}},y}$ which are such that $\mathcal{S}_{\check{\mathbf{x}},x} = \mathcal{S}_{\check{\mathbf{x}},y}$. From the definition of these two sets, $\mathcal{S}_{\check{\mathbf{x}}}$ is obtained the same way as $\mathcal{S}_{\tilde{\mathbf{x}}}$. Again, since $\mathcal{S}_{\check{\mathbf{x}},x}$ is composed of 29 inequalities, the values of the matrices inducing it are not reported here. However, it is shown in Figure 4.8. Along with the sets $\mathcal{S}_{\tilde{\mathbf{x}},x} = \mathcal{S}_{\tilde{\mathbf{x}},y}$ and $\mathcal{S}_{\check{\mathbf{x}},x} = \mathcal{S}_{\check{\mathbf{x}},y}$, Figure 4.8 also displays the set $\boldsymbol{K}\mathcal{S}_{\check{\mathbf{x}}}$.

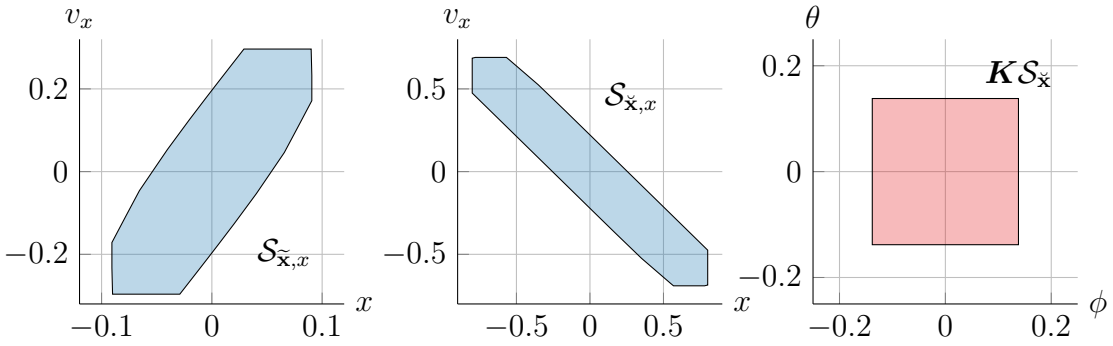


Figure 4.8: Invariant sets for the tube-based MPC controller in the UAV case.

According to Section 3.3.2, a cascaded control structure is considered to control each UAV. For the inner loop, a continuous-time controller based on feedback linearization is used for attitude control. For the outer loop, the discrete-time tube-based MPC of Section 4.1.3 is used for position control. However, for simulation purpose, the controller of the inner loop is run in discrete-time with a sampling period of 1 ms. This sampling period is 200 times shorter than the sampling period of the outer loop. It is not a problem in the nominal case described in Section 3.3.3 but in the perturbed case, this might cause a larger estimation error and even lead to instability. Then the outer loop needs a modified version of the position controller described in Figure 3.7. This structure is based on the ones used in Chevet et al. (2020a) and Rousseau et al. (2018). The MPC part of the tube-based MPC controller runs at the sampling period T_s. However, the control input applied to the agent $i \in \overline{1,N}$ is changed to $\mathbf{u}_i(k_{\text{in}}) = \breve{\mathbf{u}}_i(k) + \boldsymbol{K}(\widehat{\mathbf{x}}_i(k_{\text{in}}) - \breve{\mathbf{x}}_i(k))$ for all $k_{\text{in}} \in \overline{\frac{T_s}{T_s^{\text{in}}}k, \frac{T_s}{T_s^{\text{in}}}(k+1) - 1}$ where $T_s^{\text{in}} = 1\,\text{ms}$ is the sampling period of the inner loop. The gain matrix $\boldsymbol{L}_{\text{in}}$ of the observer is obtained by pole placement. The poles placed to obtain $\boldsymbol{L}_{\text{in}}$ are the eigenvalues of $\boldsymbol{A} - \boldsymbol{LC}$. The matrix $\boldsymbol{A}$ used to get the gain matrix $\boldsymbol{L}_{\text{in}}$ is the matrix $\boldsymbol{A}$ of (4.30) and (4.31) obtained by replacing T_s by T_s^{in}. With such a structure, the estimated state $\widehat{\mathbf{x}}_i(k_{\text{in}})$ used for the tube-based controller is more accurate and guarantees that the estimation and deviation errors remain

inside their respective sets. Indeed, having the observer running at T_s and using $\widehat{\mathbf{x}}_i(k)$ for the control input might provoke a transient state where the estimation and deviation error do not belong to their sets, voiding the stability guarantees.

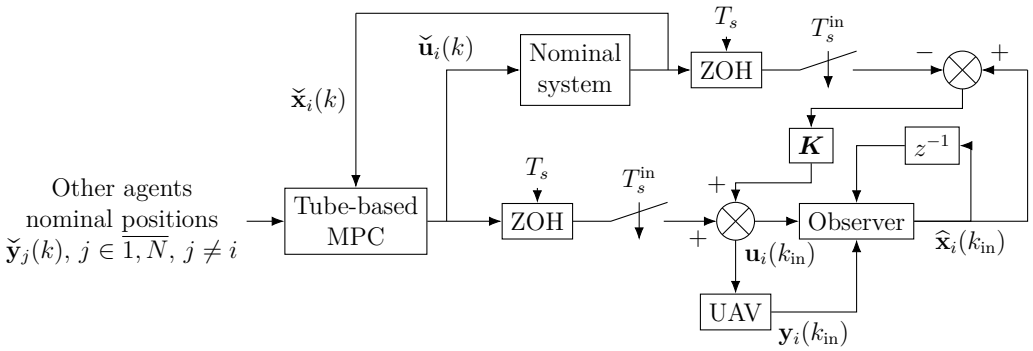


Figure 4.9: Structure of the position controller for a quadrotor UAV subject to perturbations.

As in Section 3.3.3, the contraction factor for the terminal constraint is $\lambda_i = 0.9$, with $i \in \overline{1,N}$. The weighting matrices are chosen such that $\boldsymbol{Q} = \boldsymbol{I}_4$, $\boldsymbol{R} = \boldsymbol{I}_2$ and $\boldsymbol{P}$ is the solution of the algebraic Riccati equation:

$$\boldsymbol{A}^\top \boldsymbol{P}\boldsymbol{A} - \boldsymbol{P} - \boldsymbol{A}^\top \boldsymbol{P}\boldsymbol{B}\big(\boldsymbol{B}^\top \boldsymbol{P}\boldsymbol{B} + \boldsymbol{R}\big)^{-1}\boldsymbol{B}^\top \boldsymbol{P}\boldsymbol{A} + \boldsymbol{Q} = \boldsymbol{0}_4$$

with $\boldsymbol{A}$ and $\boldsymbol{B}$ defined in (3.39). The solver for the optimization problem (4.9) is generated with CVXGEN (Mattingley and Boyd, 2012, 2013). Moreover, the UAVs are simulated with the full nonlinear model described in Section 3.3.

The only components of the state vector of each UAV agent that are initialized are the position components. The speed, angles and angular speeds are null at $k = 0$ and the altitude is such that $z_i(0) = \bar{z} = 5\,\text{m}$ for all $i \in \overline{1,N}$. The nominal position $\breve{\mathbf{y}}_i$ of the N quadrotor UAVs is initialized to a random value such that the sets $\{\breve{\mathbf{y}}_i(0)\} \oplus \mathcal{W}$ do not overlap. The initial value of the estimated state $\widehat{\mathbf{x}}_i$ for all $i \in \overline{1,N}$ is set such that $\boldsymbol{C}\widehat{\mathbf{x}}_i(0) = \breve{\mathbf{y}}_i(0)$ and the other components are zero, and the position $\mathbf{y}_i$ is initialized to a random value in $\{\breve{\mathbf{y}}_i(0)\} \oplus \boldsymbol{C}\mathcal{S}_{\tilde{\mathbf{x}}}$, the components of $\mathbf{x}_i$ not participating in $\boldsymbol{C}\mathbf{x}_i$ being zero. During the whole simulation, the input and output perturbation signals $\mathbf{d}_i(k)$ and $\mathbf{w}_i(k)$ take a random value in $\mathcal{D}$ and $\mathcal{W}$ for all $i \in \overline{1,N}$ every 10 s.

The initial configuration of the MAS Σ in $\mathcal{Y}$ is displayed in Figure 4.10. The nominal position $\breve{\mathbf{y}}_i(0)$, with $i \in \overline{1,N}$, of the agents is represented by circles, the real position of the agents $\mathbf{y}_i(0)$, by diamonds, and the Chebyshev center position $\mathbf{c}_i(0)$, by stars. In addition to the guaranteed Voronoi tessellation of the MAS are also presented the output constraint sets $\breve{\mathcal{V}}_i^g(0)$ for the MPC problem (4.9) with dashed frontiers. The sets $\{\breve{\mathbf{y}}_i(0)\} \oplus \boldsymbol{C}(\mathcal{S}_{\tilde{\mathbf{x}}} \oplus \mathcal{S}_{\breve{\mathbf{x}}})$ are shown to ensure that $\mathbf{y}_i(0)$ is correctly initialized.

The agents of Σ then follow their Chebyshev centers inside $\mathcal{Y}$. An intermediate state, at $t = 1\,\text{s}$, of the deployment is presented in Figure 4.11 and the final configuration is shown in Figure 4.12 with the same elements as in Figure 4.10. It is obvious from Figure 4.12 that the nominal positions $\breve{\mathbf{y}}_i$, with $i \in \overline{1,N}$, have reached the Chebyshev centers $\mathbf{c}_i$ of their guaranteed Voronoi cell $\mathcal{V}_i^g$. Moreover, Figure 4.12 gives the impression that the real position $\mathbf{y}_i$, with $i \in \overline{1,N}$, of the agents

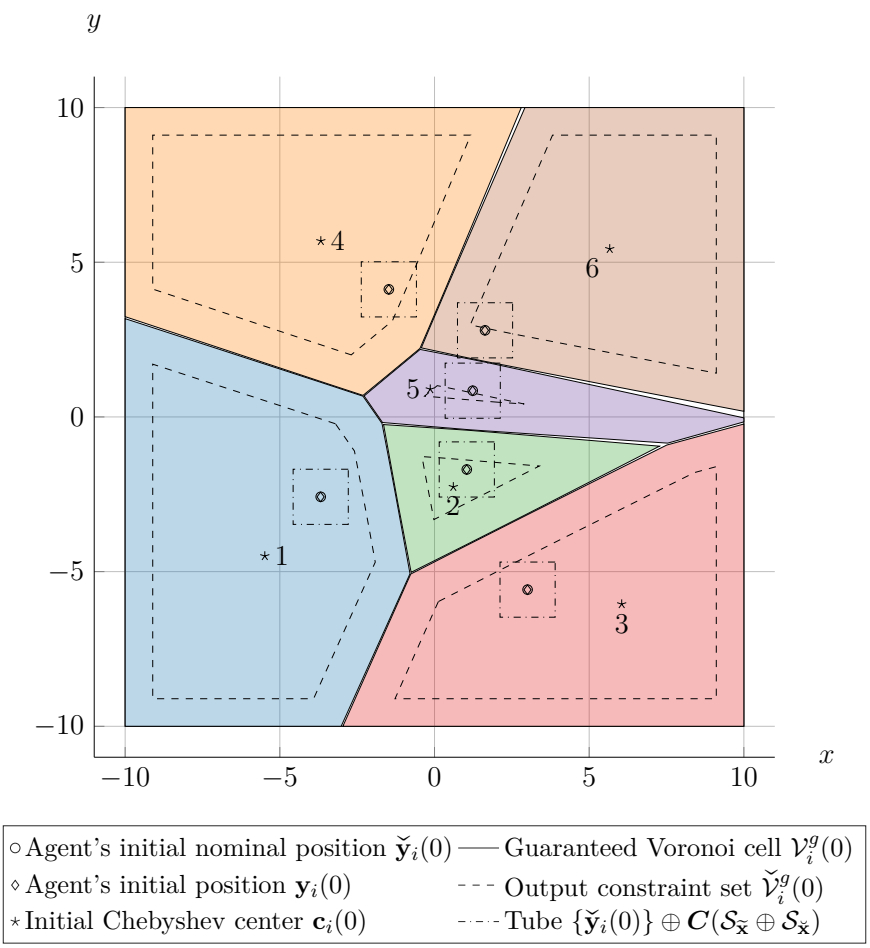


Figure 4.10: Initial position of the MVS Σ in the output space.

has converged to the corresponding nominal position $\breve{\mathbf{y}}_i$. Indeed, due to the small amplitude of the perturbations, the estimation and deviation errors $\tilde{\mathbf{x}}_i$ and $\breve{\mathbf{x}}_i$ are small, leading to $\mathbf{y}_i \approx \breve{\mathbf{y}}_i$.

For Figures 4.10 to 4.12, the area not covered by the guaranteed Voronoi tessellation is reduced with respect to what appears in the single integrator dynamics case of Paragraph 4.1.4.1. Indeed, the perturbation applied to the output is smaller than the perturbation applied in the single integrator case, thus bringing the guaranteed Voronoi tessellation close to the classical Voronoi tessellation.

A measure of the convergence of the system to a static configuration is presented in Figure 4.13. This figure presents the distance over time between the Chebyshev center and the nominal position such that:

$$d_i(k) = \|\breve{\mathbf{y}}_i(k) - \mathbf{c}_i(k)\|_2$$

for all $i \in \overline{1, N}$. As expected from the nominal case, the distances converge to 0. Since the distance stays at zero for the remainder of the simulation, the plot is cut after 25 s.

As for the single integrator case presented in Paragraph 4.1.4.1, the efficiency of the proposed control algorithm can be measured via the norm of the estimation error $\tilde{\mathbf{x}}_i$ as well as the norm of the deviation error $\breve{\mathbf{x}}_i$. Figure 4.14 and Figure 4.15 show the evolution of these two norms over time. Since no offset cancellation strategy (Limón et al., 2010) is present in the proposed control algorithm, the estimation and

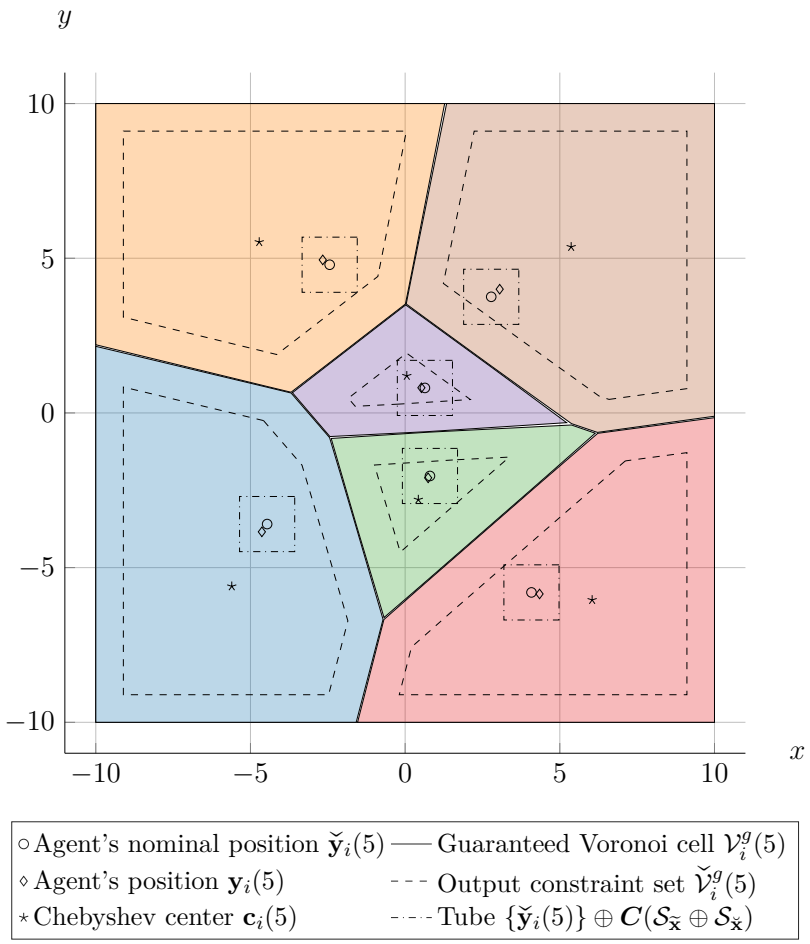


Figure 4.11: Position of the MVS Σ in the output space at $t = 1\,\mathrm{s}$.

deviation errors never go back to zero but their norms converge to a constant value. The maximum norm of the vectors of $\mathcal{S}_{\tilde{\mathbf{x}}}$ and $\mathcal{S}_{\breve{\mathbf{x}}}$ are defined as in Paragraph 4.1.4.1 and are such that $\|\tilde{\mathbf{x}}_{\max}\|_2 = 0.4380\,\mathrm{m}$ and $\|\breve{\mathbf{x}}_{\max}\|_2 = 1.5058\,\mathrm{m}$. These two values are indeed upper bounds of the plots presented in Figure 4.14 and Figure 4.15. Once again, the pikes that appear in both figures come from the change of the perturbation signals $\mathbf{d}_i(k)$ and $\mathbf{w}_i(k)$ every 10 s.

The amplitude of the perturbations the tube-based MPC of Section 4.1.3 can deal with is however limited. Indeed, if the perturbation sets $\mathcal{D}$ and $\mathcal{W}$ are too wide, the invariant sets associated with the error dynamics (4.4) and (4.6) might end up being too wide, equating the output constraint set to the empty set. A discussion on further robustification of the tube-based MPC introduced in Section 4.1.3 is then drawn in Section 4.3.

In Section 4.1, a tube-based MPC algorithm with state estimation and guaranteed Voronoi cells is introduced for the deployment of a multi-vehicle system subject to bounded perturbations. The next section discusses the case of unbounded stochastic perturbations and proposes a chance-constrained MPC algorithm for the Voronoi-based deployment of a MVS.

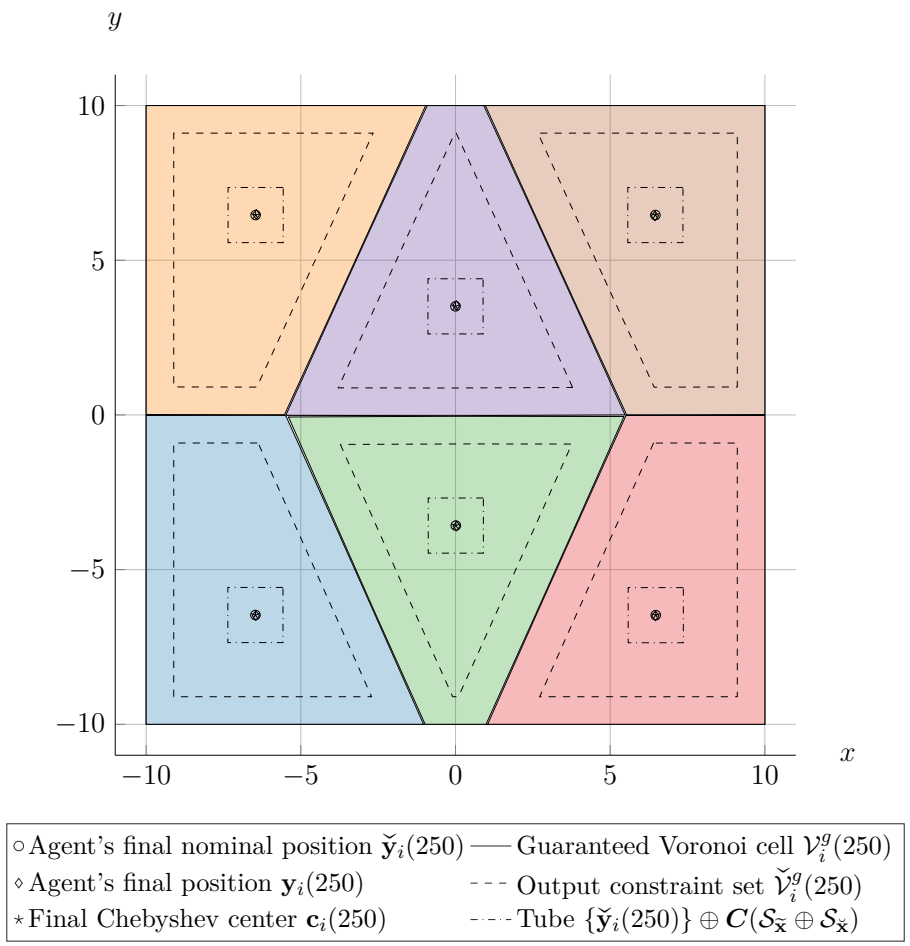


Figure 4.12: Final position of the MVS Σ in the output space at $t = 50\,\mathrm{s}$.

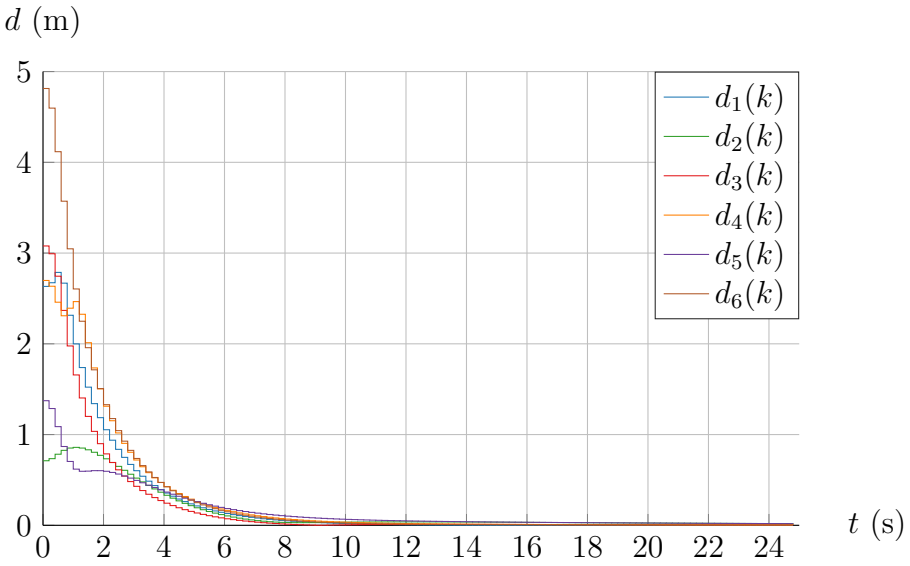


Figure 4.13: Distance of the nominal position of each agent of Σ to its Chebyshev center over time.

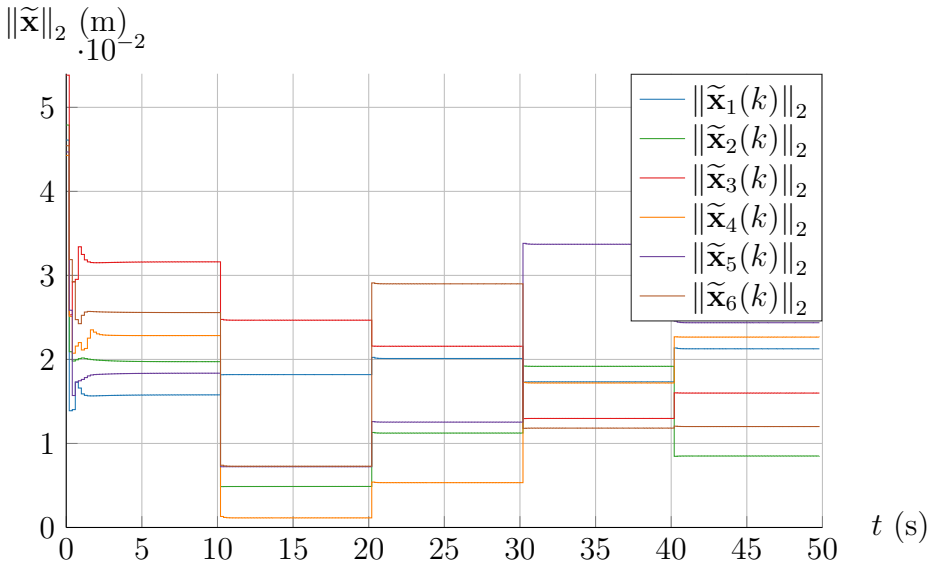


Figure 4.14: Norm of the estimation error of each agent of Σ.

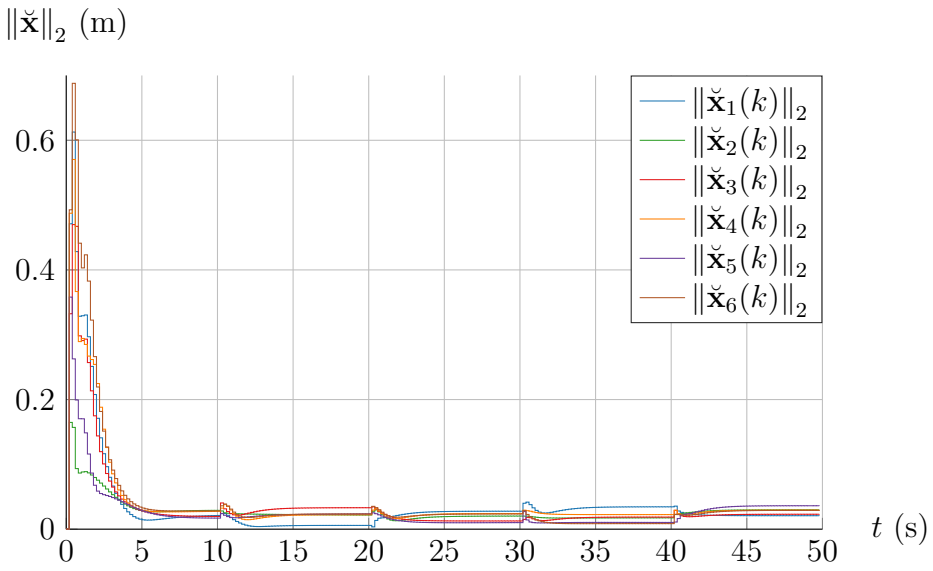


Figure 4.15: Norm of the deviation error of each agent of Σ.

4.2 Unbounded stochastic perturbations

In the following section, unbounded stochastic perturbations will be considered on the system model. To this end, Section 4.2.1 introduces the necessary modifications on the model of an agent part of a larger MAS before presenting an overview of existing methods for control of systems subject to unbounded stochastic perturbations in Section 4.2.2. The decentralized model predictive control algorithm presented in Section 3.2.2 is reformulated to cope with agents which dynamics is subject to process and measurement noises. The proposed chance-constrained MPC algorithm is then exposed in Section 4.2.3 before finally being applied in simulation to a MAS composed of vehicle agents following single integrator or UAV dynamics.

4.2.1 System model

Let Σ be a multi-agent system composed of N agents. Each agent obeys discrete-time linear time-invariant dynamics as in the previous chapter. However, while it was not relevant for the cases presented in Chapter 3 and Section 4.1, it is important to note that such discrete-time dynamics often, if not always, arise from discretized continuous-time dynamics. The continuous-time linear time-invariant dynamics of an agent subject to process and measurement noise is written:

$$\begin{aligned} \dot{\mathbf{x}}_i(t) &= \boldsymbol{A}_C\mathbf{x}_i(t) + \boldsymbol{B}_C\mathbf{u}_i(t) + \boldsymbol{M}\mathbf{d}_i(t) \\ \mathbf{y}_i(t) &= \boldsymbol{C}\mathbf{x}_i(t) + \mathbf{w}_i(t) \end{aligned} \tag{4.32}$$

where $\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^n$, $\mathbf{u}_i \in \mathcal{U} \subset \mathbb{R}^m$, $\mathbf{y}_i \in \mathcal{Y} \subset \mathbb{R}^2$, with $i \in \overline{1, N}$, $\boldsymbol{A}_C \in \mathbb{R}^{n\times n}$, $\boldsymbol{B}_C \in \mathbb{R}^{n\times m}$, $\boldsymbol{C} \in \mathbb{R}^{2\times n}$ and $\boldsymbol{M} \in \mathbb{R}^{n\times q}$, $q \leq n$. Since the agents share the same dynamics and the same output space by Assumptions 2.6 and 3.1, the dependency in $i \in \overline{1, N}$ is dropped for $\boldsymbol{A}_C$, $\boldsymbol{B}_C$ and $\boldsymbol{C}$ and for $\mathcal{Y}$. The dependency in $i \in \overline{1, N}$ for $\mathcal{X}$ and $\mathcal{U}$ is also dropped for the sake of simplicity. The signals $\mathbf{d}_i \in \mathbb{R}^q$ and $\mathbf{w}_i \in \mathbb{R}^2$, with $i \in \overline{1, N}$, are the process and measurement noises.

Assumption 4.2: Process noise matrix

The matrix $\boldsymbol{M} \in \mathbb{R}^{n\times q}$ has rank q, with $q \leq n$.

The process noise signal $\mathbf{d}_i(t)$ present in the dynamics (4.32) is in fact twofold. It is the sum $\mathbf{d}_i(t) = \mathbf{p}_i(t) + \mathbf{w}_i^s(t)$ of a deterministic exogenous perturbation signal denoted by $\mathbf{p}_i(t) \in \mathbb{R}^q$ and a normally distributed white noise signal $\mathbf{w}_i^s \in \mathbb{R}^q$ (as defined in Definition 2.37) representing the modeling errors. As such, the mean and variance matrix of $\mathbf{d}_i(t)$ are such that $\boldsymbol{\mu}_{\mathbf{d}_i}(t) = \mathbf{p}_i(t)$ and $\boldsymbol{\Sigma}_{\mathbf{d}_i}(t) = \boldsymbol{\Sigma}_{\mathbf{w}_i^s}(t)$. For its part, the measurement noise signal $\mathbf{w}_i(t)$ is such that $\boldsymbol{\mu}_{\mathbf{w}_i}(t) = \mathbf{0}_{2\times 1}$ and a variance matrix given in the following assumption.

Assumption 4.3: Normally distributed noises

The signal $\mathbf{w}_i^s(t)$ and the measurement noise $\mathbf{w}_i$, with $i \in \overline{1, N}$, are independent and normally distributed white noises as defined in Definition 2.37. The means of $\mathbf{w}_i^s(t)$ and $\mathbf{w}_i(t)$ are then $\boldsymbol{\mu}_{\mathbf{w}_i^s}(t) = \mathbf{0}_{q\times 1}$ and $\boldsymbol{\mu}_{\mathbf{w}_i}(t) = \mathbf{0}_{2\times 1}$ and their variance matrices are the positive definite matrices $\boldsymbol{\Sigma}_{\mathbf{w}_i^s}(t) \succ 0$ and $\boldsymbol{\Sigma}_{\mathbf{w}_i}(t) \succ 0$.

The discrete-time linear time-invariant dynamics used later for control purposes is obtained by discretizing (4.32):

$$\begin{aligned} \mathbf{x}_i(k+1) &= \boldsymbol{A}\mathbf{x}_i(k) + \boldsymbol{B}\mathbf{u}_i(k) + \boldsymbol{\delta}_i(k) \\ \mathbf{y}_i(k) &= \boldsymbol{C}\mathbf{x}_i(k) + \boldsymbol{\gamma}_i(k) \end{aligned} \tag{4.33}$$

where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{B} \in \mathbb{R}^{n\times m}$ and $\boldsymbol{C} \in \mathbb{R}^{p\times n}$. The signals $\boldsymbol{\delta}_i \in \mathbb{R}^n$ and $\boldsymbol{\gamma}_i \in \mathbb{R}^2$, with $i \in \overline{1, N}$, are the discrete-time process and measurement noises. The matrices in (4.33) are obtained as (Franklin et al., 1997):

$$\boldsymbol{A} = \exp(\boldsymbol{A}_C \cdot T_s) \text{ and } \boldsymbol{B} = \int_0^{T_s} \exp(\boldsymbol{A}_C \cdot \tau)\boldsymbol{B}\mathrm{d}\tau$$

where T_s is the sampling period and exp is the exponential function. The matrix $\boldsymbol{C}$ is not modified by the discretization process.

The discretization of the random signals is not as straightforward as the discretization of the state-space matrices. Indeed, the discrete-time process noise is (Franklin et al., 1997):

$$\boldsymbol{\delta}_i(k) = \int_0^{T_s} \exp(\boldsymbol{A}_C \cdot \tau)\boldsymbol{M}\mathbf{d}_i((k+1)T_s - \tau)\mathrm{d}\tau$$

while the discrete-time measurement noise is $\boldsymbol{\gamma}_i(k) = \mathbf{w}_i(kT_s)$. However, for control purposes, it is important to characterize random signals such as the discrete-time process and measurement noises $\boldsymbol{\delta}_i(k)$ and $\boldsymbol{\gamma}_i(k)$ with their mean and variance. They can be obtained from the value of the mean and variance matrix of their continuous-time counterparts $\mathbf{d}_i(k)$ and $\mathbf{w}_i(k)$. For the following mathematical developments, let $h_k(t) = (k+1)T_s - t$.

By definition of the mathematical expectation, it is immediate that:

$$\boldsymbol{\mu}_{\boldsymbol{\gamma}_i}(k) = \mathbb{E}(\boldsymbol{\gamma}_i(k)) = \mathbb{E}(\mathbf{w}_i(kT_s)) = \boldsymbol{\mu}_{\mathbf{w}_i}(kT_s) \tag{4.34}$$

and, by linearity of the mathematical expectation:

$$\begin{aligned}\boldsymbol{\mu}_{\boldsymbol{\delta}_i}(k) = \mathbb{E}(\boldsymbol{\delta}_i(k)) &= \int_0^{T_s} \exp(\boldsymbol{A}_C \cdot \tau)\boldsymbol{M}\mathbb{E}(\mathbf{d}_i(h_k(\tau)))\mathrm{d}\tau \\ &= \int_0^{T_s} \exp(\boldsymbol{A}_C \cdot \tau)\boldsymbol{M}\boldsymbol{\mu}_{\mathbf{d}_i}(h_k(\tau))\mathrm{d}\tau. \end{aligned} \tag{4.35}$$

Moreover, it is a well known result (Kamen and Su, 1999) that:

$$\boldsymbol{\Sigma}_{\boldsymbol{\gamma}_i}(k) = \frac{1}{T_s}\boldsymbol{\Sigma}_{\mathbf{w}_i}(kT_s) \tag{4.36}$$

for all $k \geq 0$ such that $\boldsymbol{\Sigma}_{\boldsymbol{\gamma}_i}(k) \succ 0$. The variance matrix of $\boldsymbol{\delta}_i(k)$ is:

$$\begin{aligned}\boldsymbol{\Sigma}_{\boldsymbol{\delta}_i}(k) &= \mathbb{E}\Big((\boldsymbol{\delta}_i(k) - \boldsymbol{\mu}_{\boldsymbol{\delta}_i}(k))(\boldsymbol{\delta}_i(k) - \boldsymbol{\mu}_{\boldsymbol{\delta}_i}(k))^\top\Big) \\ &= \mathbb{E}\Bigg(\int_0^{T_s} \exp(\boldsymbol{A}_C \cdot \tau)\boldsymbol{M}(\mathbf{d}_i(h_k(\tau)) - \boldsymbol{\mu}_{\mathbf{d}_i}(h_k(\tau)))\mathrm{d}\tau \\ &\qquad \cdot \int_0^{T_s} (\mathbf{d}_i(h_k(\nu)) - \boldsymbol{\mu}_{\mathbf{d}_i}(h_k(\nu)))^\top \boldsymbol{M}^\top \exp\big(\boldsymbol{A}_C^\top \cdot \nu\big)\mathrm{d}\nu\Bigg)\end{aligned}$$

which, by Fubini's theorem (DiBenedetto, 2016), gives:

$$\begin{aligned}\boldsymbol{\Sigma}_{\boldsymbol{\delta}_i}(k) = \mathbb{E}\Bigg(\int_0^{T_s}\int_0^{T_s} &\exp(\boldsymbol{A}_C \cdot \tau)\boldsymbol{M}(\mathbf{d}_i(h_k(\tau)) - \boldsymbol{\mu}_{\mathbf{d}_i}(h_k(\tau)))\cdot \\ &(\mathbf{d}_i(h_k(\nu)) - \boldsymbol{\mu}_{\mathbf{d}_i}(h_k(\nu)))^\top \boldsymbol{M}^\top \exp\big(\boldsymbol{A}_C^\top \cdot \nu\big)\mathrm{d}\tau\mathrm{d}\nu\Bigg).\end{aligned}$$

By definition:

$$\mathbb{E}\Big((\mathbf{d}_i(h_k(\tau)) - \boldsymbol{\mu}_{\mathbf{d}_i}(h_k(\tau)))(\mathbf{d}_i(h_k(\nu)) - \boldsymbol{\mu}_{\mathbf{d}_i}(h_k(\nu)))^\top\Big) = \boldsymbol{\Sigma}_{\mathbf{d}_i}(\tau)\delta(\tau - \nu)$$

where δ is the Dirac delta function such that $\delta(\tau - \nu) \neq 0$ when $\tau = \nu$. Thus, by linearity of the mathematical expectation:

$$\boldsymbol{\Sigma}_{\boldsymbol{\delta}_i}(k) = \int_0^{T_s} \exp(\boldsymbol{A}_C \cdot \tau)\boldsymbol{M}\boldsymbol{\Sigma}_{\mathbf{d}_i}(\tau)\boldsymbol{M}^\top \exp\big(\boldsymbol{A}_C^\top \cdot \tau\big)\mathrm{d}\tau. \tag{4.37}$$

Property 4.1: Positive definiteness of the process noise variance matrix

The variance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\delta}_i}(k)$ of the discrete-time process noise is positive definite for all $k \geq 0$.

Proof. Given Assumption 4.3, $\boldsymbol{\Sigma}_{\mathbf{d}_i}(t) \succ 0$ for all $t \in \mathbb{R}_+$. By combining Assumption 4.2 and Property 2.2:

$$\mathbf{x}^\top \exp(\boldsymbol{A}_C \cdot t)\boldsymbol{M}\boldsymbol{\Sigma}_{\mathbf{d}_i}(t)\boldsymbol{M}^\top \exp\big(\boldsymbol{A}_C^\top \cdot t\big)\mathbf{x} > 0$$

for all $\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}_{n\times 1}\}$, $t \in \mathbb{R}_+$. By integration over $[0, T_s]$, it is immediate that $\mathbf{x}^\top \boldsymbol{\Sigma}_{\boldsymbol{\delta}_i}(k)\mathbf{x} > 0$, hence the result. ■

Due to the presence of unknown process and measurement noise, the real value of the state is unknown and has to be estimated. To do so, a Luenberger observer (Luenberger, 1964) is introduced. The dynamics of the state observer is then:

$$\begin{aligned} \widehat{\mathbf{x}}_i(k+1) &= \boldsymbol{A}\widehat{\mathbf{x}}_i(k) + \boldsymbol{B}\mathbf{u}_i(k) + \boldsymbol{L}(\mathbf{y}_i(k) - \widehat{\mathbf{y}}_i(k)) \\ \widehat{\mathbf{y}}_i(k) &= \boldsymbol{C}\widehat{\mathbf{x}}_i(k) \end{aligned} \tag{4.38}$$

where $\boldsymbol{L} \in \mathbb{R}^{n\times 2}$ is the observer gain.

From the dynamics of the estimated state, it is possible to derive the estimation error $\widetilde{\mathbf{x}}_i$, i.e. the difference between the real state $\mathbf{x}_i$ and the estimated state $\widehat{\mathbf{x}}_i$, obeying the dynamics:

$$\widetilde{\mathbf{x}}_i(k+1) = (\boldsymbol{A} - \boldsymbol{LC})\widetilde{\mathbf{x}}_i(k) - \boldsymbol{L}\boldsymbol{\gamma}_i(k) + \boldsymbol{\delta}_i(k) \tag{4.39}$$

the calculations leading to an equation similar to (4.4) in Section 4.1.1, where $\boldsymbol{\delta}_i$ and $\boldsymbol{\gamma}_i$ replace $\mathbf{d}_i$ and $\mathbf{w}_i$, respectively.

Due to the presence of the process and measurement noise in the dynamics of the estimation error (4.39), $\widetilde{\mathbf{x}}_i$ can be characterized by its mean and variance matrix. The mean of the estimation error is defined recursively by following the dynamics:

$$\boldsymbol{\mu}_{\widetilde{\mathbf{x}}_i}(k+1) = (\boldsymbol{A} - \boldsymbol{LC})\boldsymbol{\mu}_{\widetilde{\mathbf{x}}_i}(k) - \boldsymbol{L}\boldsymbol{\mu}_{\boldsymbol{\gamma}_i}(k) + \boldsymbol{\mu}_{\boldsymbol{\delta}_i}(k). \tag{4.40}$$

Remark 4.6: Independence
Due to Assumption 4.3, $\boldsymbol{\delta}_i(k)$ and $\boldsymbol{\gamma}_i(k)$ are independent for all $k \in \mathbb{N}$. By the same assumption, $\boldsymbol{\delta}_i(k)$ and $\boldsymbol{\delta}_i(l)$ (respectively $\boldsymbol{\gamma}_i(k)$ and $\boldsymbol{\gamma}_i(l)$) for all $k, l \in \mathbb{N}$ are independent due to the whiteness of $\mathbf{w}_i^s(t)$ (respectively $\mathbf{w}_i(t)$). ◊

The variance matrix of the estimation error $\widetilde{\mathbf{x}}_i$ can also be defined recursively. The estimation error $\widetilde{\mathbf{x}}_i(k)$ only depends on $\widetilde{\mathbf{x}}_i(k-1)$, $\boldsymbol{\delta}_i(k-1)$ and $\boldsymbol{\gamma}_i(k-1)$. Then $\widetilde{\mathbf{x}}_i(k)$, $\boldsymbol{\delta}_i(k)$ and $\boldsymbol{\gamma}_i(k)$ are independent by Remark 4.6. Then, the dynamics of the variance matrix of the estimation error $\widetilde{\mathbf{x}}_i$ is:

$$\boldsymbol{\Sigma}_{\widetilde{\mathbf{x}}_i}(k+1) = (\boldsymbol{A} - \boldsymbol{LC})\boldsymbol{\Sigma}_{\widetilde{\mathbf{x}}_i}(k)(\boldsymbol{A} - \boldsymbol{LC})^\top + \boldsymbol{L}\boldsymbol{\Sigma}_{\boldsymbol{\gamma}_i}(k)\boldsymbol{L}^\top + \boldsymbol{\Sigma}_{\boldsymbol{\delta}_i}(k). \tag{4.41}$$

Assumption 4.4: Positive definite initialization

The initial estimation error variance matrix is positive definite, i.e. $\boldsymbol{\Sigma}_{\widetilde{\mathbf{x}}_i}(0) \succ 0$.

Property 4.2: Positive definiteness of the estimation error variance

The variance matrix $\boldsymbol{\Sigma}_{\widetilde{\mathbf{x}}_i}(k)$ of the estimation error $\widetilde{\mathbf{x}}_i$, with $i \in \overline{1, N}$, is positive definite for all $k > 0$.

Proof. Consider the dynamics (4.41). Since $\boldsymbol{\Sigma}_{\boldsymbol{\delta}_i}(k) \succ 0$ and $\boldsymbol{\Sigma}_{\boldsymbol{\gamma}_i}(k) \succ 0$ for all $k \geq 0$ and $\boldsymbol{\Sigma}_{\widetilde{\mathbf{x}}_i}(0) \succ 0$ by Assumption 4.4, by recursion, it leads to $\boldsymbol{\Sigma}_{\widetilde{\mathbf{x}}_i}(k) \succ 0$ for all $k > 0$. Indeed, $(\boldsymbol{A} - \boldsymbol{LC})\boldsymbol{\Sigma}_{\widetilde{\mathbf{x}}_i}(k)(\boldsymbol{A} - \boldsymbol{LC})^\top \succeq 0$ and $\boldsymbol{L}\boldsymbol{\Sigma}_{\boldsymbol{\gamma}_i}(k)\boldsymbol{L}^\top \succeq 0$ according to Property 2.2 and $\boldsymbol{\Sigma}_{\boldsymbol{\delta}_i}(k) \succ 0$ for all $k \geq 0$ by Property 4.1 thus their sum is positive definite. ■

4.2.2 Overview of chance-constrained MPC for systems with stochastic perturbations

As in Section 4.1, perturbations are added to the system model (4.33). However, these perturbations now have a stochastic nature and the control methods have to be adapted to cope with this new kind of endogenous or exogenous signals. As in the classical bounded perturbations case presented in 4.1, several control strategies have been studied and applied when the perturbations on a system are stochastic. For example, as in the deterministic perturbations case, H_∞ optimization methods can be used (Ugrinovskii, 1998) as well as sliding mode control (Niu et al., 2005).

However, as stated previously, when dealing with dynamical systems subject to operating constraints, one of the most popular technique is model predictive control, and most particularly, stochastic, or chance-constrained, MPC (Farina et al., 2016), when the system is subject to perturbations having a stochastic nature. In this case, the system follows the dynamics presented in (4.33), leading to the definition of constraints of the form $\mathbb{P}(\mathbf{x} \in \mathcal{X}) \geq P$ denoting the fact that the probability of the vector $\mathbf{x}$ belonging to the set $\mathcal{X}$ has to be greater or equal to an arbitrary value P. Such constraints can be defined on any of the decision variables of the problem (e.g. the input vector sequence).

Farina et al. (2016) reviewed several existing methods of chance-constrained model predictive control (CCMPC) and they found three main CCMPC categories for linear systems. To define their categories, they use the superposition principle to separate the state equation of (4.33) into a stochastic part depending only on the stochastic variable (e.g. $\boldsymbol{\delta}_i$ and/or $\boldsymbol{\gamma}_i$ in (4.33)) and a deterministic part which can be assimilated to the nominal system defined in (4.2). They can then make a distinction between what they call "open-loop" control methods and "disturbance feedback" or "state-feedback" control. The "open-loop" control methods can be found for example in Blackmore et al. (2011), Gavilan et al. (2012), Hashimoto (2013) or Chevet et al. (2020a). Such methods are called "open-loop" since the controller only acts on the deterministic part while the stochastic part evolves in an uncontrolled open-loop fashion. Such an evolution is opposed to the feedback methods, where the control signal acts on both the deterministic and stochastic part of the system. For disturbance feedback (Prandini et al., 2012, Zhang et al., 2013, Korda et al., 2014), the control input is an affine function of the disturbance over the prediction horizon and the optimization problem is meant to find the value of the parameters of this function. In the case of state feedback (Primbs and Sung, 2009, Cannon et al., 2010, 2012, Kouvaritakis and Cannon, 2016), the formulation of the problem is similar, if not identical, to the tube-based MPC approach presented in

Section 4.1.2, the main difference being that a stochastic RPI set has to be defined for the control strategy to work. Both feedback methods have the advantage of enlarging the feasibility region of the MPC optimization problem.

Regardless of the category defined in Farina et al. (2016) the controller falls into, the main problem that will appear is the presence of unknown terms in the constraints. This is due to the probabilistic nature of said constraints and the fact that stochastic signals are taken into account when expressing them. To overcome this difficulty, the probabilistic constraints are reformulated into algebraic constraints by using the properties of the stochastic process used to represent the perturbation. This often amounts to tighten the constraints by finding a bound on the perturbation signal ensuring that the probabilistic constraint is respected with the required probability.

CCMPC approaches have been used for the control of different types of vehicles (Blackmore et al., 2011) such as ground vehicles (Carvalho et al., 2014, Wan et al., 2017), spacecrafts (Gavilan et al., 2012, Zhu et al., 2018) or unmanned aerial vehicles (Chevet et al., 2020a). Chance-constrained approaches have also been developed for multi-agent systems (Lyons et al., 2012, Dai et al., 2015, 2016, 2018) and applied on vehicles such as UAVs (Chevet et al., 2020a) or multi-robot frameworks (Zhu and Alonso-Mora, 2019).

This section focuses on the work presented in Chevet et al. (2020a). The CCMPC algorithm presented is derived from the work of Gavilan et al. (2012) which describes an "open-loop" (as per Farina et al. (2016) naming) scheme to control a spacecraft. The controller proposed in Gavilan et al. (2012) is improved to work in the case where full state information is not known and state estimation is necessary. It is then applied in a decentralized fashion to a multi-agent system composed first of agents obeying single integrator dynamics before being applied to a MAS composed of quadrotor UAVs.

4.2.3 Deployment algorithm

As in Chapter 3, the objective of the deployment algorithm that is presented in this paragraph is to drive the MAS defined in Section 4.2.1 into a static configuration in $\mathcal{Y}$. The set $\mathcal{Y}$ is a convex bounded polytope as defined in Assumption 3.1. However, an important difference with the algorithm for the nominal case presented in Chapter 3 appears due to the unbounded stochastic perturbation on the dynamics (4.33). Indeed, such perturbations induce the presence of probabilistic constraints in the MPC problem. As described in Section 4.2.2, such constraints have to be relaxed into algebraic constraints for the optimization problem on which the MPC is based to be solvable. This section then describe how such a relaxation can be done.

4.2.3.1 Deployment objective and chance-constrained optimization problem

In the nominal case of Chapter 3, the MPC algorithm uses as input the current state of the system $\mathbf{x}_i(k)$ for all $i \in \overline{1, N}$. In the case of systems subject to perturbations and/or with an output $\mathbf{y}_i(k)$ such that $\mathbf{y}_i(k) \neq \mathbf{x}_i(k)$, the characterization of the system is done via the estimated state $\widehat{\mathbf{x}}_i(k)$ obtained through a state observer. Then, in this case, the MPC uses this estimated state of the system $\widehat{\mathbf{x}}_i(k)$ to obtain the control input $\mathbf{u}_i(k)$ to be applied to the system. As for the bounded perturbations

case, a first assumption is made, based on Assumption 2.5, to deal with the case of stochastic perturbations.

Assumption 4.5: Knowledge of environment under unbounded stochastic perturbations

Each agent of Σ knows, at all time, the estimated position of the other agents of Σ.

The deployment objective is the same as the one presented in Section 3.2. At each time instant, each agent $i \in \overline{1,N}$ computes its Voronoi cell $\mathcal{V}_i(k)$ with the knowledge of $\widehat{\mathbf{y}}_j(k)$, with $j \in \overline{1,N}$, $j \neq i$, by Assumption 4.5. Then, with the knowledge of its Voronoi cell, each agent is able to compute the Chebyshev center $\mathbf{c}_i(k)$ of $\mathcal{V}_i(k)$ by solving the problem (3.5). The objective of the controller is then to drive each agent towards the Chebyshev center $\mathbf{c}_i(k)$ of $\mathcal{V}_i(k)$ using a chance-constrained MPC algorithm. However, while in the nominal case, each agent had to reach the Chebyshev center of its Voronoi cell, i.e. $\lim_{k\to+\infty} \mathbf{y}_i(k) = \lim_{k\to+\infty} \mathbf{c}_i(k)$, here, the estimated position of each agent has to reach the Chebyshev center of its Voronoi cell, i.e. $\lim_{k\to+\infty} \widehat{\mathbf{y}}_i(k) = \lim_{k\to+\infty} \mathbf{c}_i(k)$.

In order to introduce the optimization problem on which the MPC is based, it is necessary to formulate the evolution of the estimated state vector over the prediction horizon $N_p \in \mathbb{N}$. Since the dynamics (4.38) are defined recursively, it is possible to write:

$$\begin{aligned}\widehat{\mathbf{x}}_i(k+l) &= \boldsymbol{A}\widehat{\mathbf{x}}_i(k+l-1) + \boldsymbol{B}\mathbf{u}_i(k+l-1) + \boldsymbol{L}\boldsymbol{C}\widetilde{\mathbf{x}}_i(k+l-1) + \boldsymbol{L}\boldsymbol{\gamma}_i(k+l-1)\\ &= \boldsymbol{A}^l\widehat{\mathbf{x}}_i(k) + \sum_{j=0}^{l-1} \boldsymbol{A}^{l-j-1}(\boldsymbol{B}\mathbf{u}_i(k+j) + \boldsymbol{L}(\boldsymbol{C}\widetilde{\mathbf{x}}_i(k+j) + \boldsymbol{\gamma}_i(k+j))).\end{aligned}$$

Then, defining the following vectors and matrices:

$$\widehat{\mathbf{X}}_i(k) = \begin{bmatrix}\widehat{\mathbf{x}}_i(k+1)\\ \vdots \\ \widehat{\mathbf{x}}_i(k+N_p)\end{bmatrix} \qquad \boldsymbol{F} = \begin{bmatrix}\boldsymbol{A}\\ \vdots \\ \boldsymbol{A}^{N_p}\end{bmatrix}$$

$$\mathbf{U}_i(k) = \begin{bmatrix}\mathbf{u}_i(k)\\ \vdots \\ \mathbf{u}_i(k+N_p-1)\end{bmatrix} \qquad \boldsymbol{G} = \begin{bmatrix}\boldsymbol{B} & \mathbf{0}_{n\times m} & \cdots & \mathbf{0}_{n\times m}\\ \boldsymbol{A}\boldsymbol{B} & \boldsymbol{B} & \ddots & \vdots \\ \vdots & & \ddots & \mathbf{0}_{n\times m}\\ \boldsymbol{A}^{N_p-1}\boldsymbol{B} & \cdots & \boldsymbol{A}\boldsymbol{B} & \boldsymbol{B}\end{bmatrix}$$

$$\widetilde{\mathbf{X}}_i(k) = \begin{bmatrix}\widetilde{\mathbf{x}}_i(k)\\ \vdots \\ \widetilde{\mathbf{x}}_i(k+N_p-1)\end{bmatrix} \qquad \boldsymbol{G}_{\widetilde{\mathbf{x}}} = \begin{bmatrix}\boldsymbol{L}\boldsymbol{C} & \mathbf{0}_{n\times n} & \cdots & \mathbf{0}_{n\times n}\\ \boldsymbol{A}\boldsymbol{L}\boldsymbol{C} & \boldsymbol{L}\boldsymbol{C} & \ddots & \vdots \\ \vdots & & \ddots & \mathbf{0}_{n\times n}\\ \boldsymbol{A}^{N_p-1}\boldsymbol{L}\boldsymbol{C} & \cdots & \boldsymbol{A}\boldsymbol{L}\boldsymbol{C} & \boldsymbol{L}\boldsymbol{C}\end{bmatrix}$$

$$\boldsymbol{\Gamma}_i(k) = \begin{bmatrix}\boldsymbol{\gamma}_i(k)\\ \vdots \\ \boldsymbol{\gamma}_i(k+N_p-1)\end{bmatrix} \qquad \boldsymbol{G}_{\boldsymbol{\gamma}} = \begin{bmatrix}\boldsymbol{L} & \mathbf{0}_{n\times 2} & \cdots & \mathbf{0}_{n\times 2}\\ \boldsymbol{A}\boldsymbol{L} & \boldsymbol{L} & \ddots & \vdots \\ \vdots & & \ddots & \mathbf{0}_{n\times 2}\\ \boldsymbol{A}^{N_p-1}\boldsymbol{L} & \cdots & \boldsymbol{A}\boldsymbol{L} & \boldsymbol{L}\end{bmatrix}$$

the estimated state over the prediction horizon is:

$$\widehat{\mathbf{X}}_i(k) = \boldsymbol{F}\widehat{\mathbf{x}}_i(k) + \boldsymbol{G}\mathbf{U}_i(k) + \boldsymbol{G}_{\widetilde{\mathbf{x}}}\widetilde{\mathbf{X}}_i(k) + \boldsymbol{G}_{\boldsymbol{\gamma}}\boldsymbol{\Gamma}_i(k). \tag{4.42}$$

However, the estimation error $\widetilde{\mathbf{x}}_i$ is also defined recursively and can be expressed over the prediction horizon. Defining:

$$\boldsymbol{\Delta}_i(k) = \begin{bmatrix} \boldsymbol{\delta}_i(k) \\ \vdots \\ \boldsymbol{\delta}_i(k+N_p-1) \end{bmatrix} \quad \widetilde{\boldsymbol{G}}_{\boldsymbol{\delta}} = \begin{bmatrix} \boldsymbol{I}_n & \mathbf{0}_{n\times n} & \cdots & \mathbf{0}_{n\times n} \\ \boldsymbol{A}-\boldsymbol{LC} & \boldsymbol{I}_n & \ddots & \vdots \\ \vdots & & \ddots & \mathbf{0}_{n\times n} \\ (\boldsymbol{A}-\boldsymbol{LC})^{N_p-1} & \cdots & \boldsymbol{A}-\boldsymbol{LC} & \boldsymbol{I}_n \end{bmatrix}$$

$$\widetilde{\boldsymbol{F}} = \begin{bmatrix} \boldsymbol{A}-\boldsymbol{LC} \\ \vdots \\ (\boldsymbol{A}-\boldsymbol{LC})^{N_p} \end{bmatrix} \quad \widetilde{\boldsymbol{G}}_{\boldsymbol{\gamma}} = \begin{bmatrix} \boldsymbol{L} & \mathbf{0}_{n\times 2} & \cdots & \mathbf{0}_{n\times 2} \\ (\boldsymbol{A}-\boldsymbol{LC})\boldsymbol{L} & \boldsymbol{L} & \ddots & \vdots \\ \vdots & & \ddots & \mathbf{0}_{n\times 2} \\ (\boldsymbol{A}-\boldsymbol{LC})^{N_p-1}\boldsymbol{L} & \cdots & (\boldsymbol{A}-\boldsymbol{LC})\boldsymbol{L} & \boldsymbol{L} \end{bmatrix}$$

the estimation error over the prediction horizon is:

$$\widetilde{\mathbf{X}}_i(k+1) = \widetilde{\boldsymbol{F}}\widetilde{\mathbf{x}}_i(k) - \widetilde{\boldsymbol{G}}_{\boldsymbol{\gamma}}\boldsymbol{\Gamma}_i(k) + \widetilde{\boldsymbol{G}}_{\boldsymbol{\delta}}\boldsymbol{\Delta}_i(k). \tag{4.43}$$

The system (4.38) satisfies Assumption 3.2, i.e. there exists a couple $(\widehat{\mathbf{x}}_{\mathbf{c}_i}(k), \mathbf{u}_{\mathbf{c}_i}(k))$ such that $(\widehat{\mathbf{x}}_{\mathbf{c}_i}(k), \mathbf{u}_{\mathbf{c}_i}(k), \mathbf{c}_i(k))$ is an equilibrium point of the nominal dynamics (3.1) satisfying:

$$\begin{aligned} \widehat{\mathbf{x}}_{\mathbf{c}_i}(k) &= \boldsymbol{A}\widehat{\mathbf{x}}_{\mathbf{c}_i}(k) + \boldsymbol{B}\mathbf{u}_{\mathbf{c}_i}(k) \\ \mathbf{c}_i(k) &= \boldsymbol{C}\widehat{\mathbf{x}}_{\mathbf{c}_i}(k). \end{aligned} \tag{4.44}$$

Since the equilibrium point is the constant objective point over the entire prediction horizon, it is possible to define $\widehat{\mathbf{X}}_{\mathbf{c}_i}(k) = \mathbf{1}_{N_p\times 1} \otimes \widehat{\mathbf{x}}_{\mathbf{c}_i}(k)$ and $\mathbf{U}_{\mathbf{c}_i}(k) = \mathbf{1}_{N_p\times 1} \otimes \mathbf{u}_{\mathbf{c}_i}(k)$.

With all these elements, based on Gavilan et al. (2012) and Chevet et al. (2020a), a decentralized chance-constrained model predictive controller for an agent $i \in \overline{1, N}$ obeying the dynamics (4.33) is formulated as:

$$\underset{\mathbf{U}_i(k)}{\text{minimize}} \quad J\Big(\widehat{\mathbf{X}}_i(k), \mathbf{U}_i(k), \widehat{\mathbf{X}}_{\mathbf{c}_i}(k), \mathbf{U}_{\mathbf{c}_i}(k)\Big) \tag{4.45a}$$

subject to

$$\mathbb{P}\Big(\widehat{\mathbf{X}}_i(k) \in \overline{\mathcal{X}}\Big) \geq P_x, \tag{4.45b}$$

$$\mathbf{U}_i(k) \in \overline{\mathcal{U}}, \tag{4.45c}$$

$$\mathbb{P}\Big((\boldsymbol{I}_{N_p} \otimes \boldsymbol{C})\widehat{\mathbf{X}}_i(k) \in \overline{\mathcal{V}}_i(k)\Big) \geq P_y \tag{4.45d}$$

where the cost function (4.45a) is:

$$\begin{aligned} &J\Big(\widehat{\mathbf{X}}_i(k), \mathbf{U}_i(k), \widehat{\mathbf{X}}_{\mathbf{c}_i}(k), \mathbf{U}_{\mathbf{c}_i}(k)\Big) \\ &\qquad = \mathbb{E}\bigg(\Big\|\widehat{\mathbf{X}}_i(k) - \widehat{\mathbf{X}}_{\mathbf{c}_i}(k)\Big\|^2_{\overline{\boldsymbol{Q}}_i} + \|\mathbf{U}_i(k) - \mathbf{U}_{\mathbf{c}_i}(k)\|^2_{\overline{\boldsymbol{R}}_i}\bigg). \end{aligned} \tag{4.46}$$

The weighting matrices $\overline{\boldsymbol{Q}}_i \in \mathbb{R}^{N_p n\times N_p n}$ and $\overline{\boldsymbol{R}}_i \in \mathbb{R}^{N_p m\times N_p m}$ are block diagonal matrices composed of N_p individual positive definite matrices $\boldsymbol{Q}_i, \boldsymbol{P}_i \in \mathbb{R}^{n\times n}$ and $\boldsymbol{R}_i \in \mathbb{R}^{m\times m}$ such that $\boldsymbol{Q}_i^\top = \boldsymbol{Q}_i \succ 0$, $\boldsymbol{P}_i^\top = \boldsymbol{P}_i \succ 0$, $\boldsymbol{R}_i^\top = \boldsymbol{R}_i \succ 0$, $\overline{\boldsymbol{Q}}_i = \operatorname{diag}\big(\boldsymbol{I}_{N_p-1} \otimes \boldsymbol{Q}_i, \boldsymbol{P}_i\big)$ and $\overline{\boldsymbol{R}}_i = \boldsymbol{I}_{N_p} \otimes \boldsymbol{R}_i$.

Remark 4.7: Terminal cost and terminal constraint
By construction of the matrix $\overline{\boldsymbol{Q}}_i$, the terminal cost is embedded in the term $\left\|\widehat{\mathbf{X}}_i(k) - \widehat{\mathbf{X}}_{\mathbf{c}_i}(k)\right\|^2_{\overline{\boldsymbol{Q}}_i}$ of the cost function (4.46). The same way, by construction of the set $\overline{\mathcal{V}}_i(k)$ described below, the terminal constraint is embedded in the constraint (4.45d). ◊

Let $\boldsymbol{H}_{\mathcal{X}} \in \mathbb{R}^{r_x \times n}$ and $\boldsymbol{\theta}_{\mathcal{X}} \in \mathbb{R}^{r_x}$ (respectively $\boldsymbol{H}_{\mathcal{U}} \in \mathbb{R}^{r_u \times m}$ and $\boldsymbol{\theta}_{\mathcal{U}} \in \mathbb{R}^{r_u}$) be the matrices inducing $\mathcal{X}$ (respectively $\mathcal{U}$). Then, the set $\overline{\mathcal{X}}$ (respectively $\overline{\mathcal{U}}$) used in the constraint (4.45b) (respectively (4.45c)) is induced by the matrices $\boldsymbol{H}_{\overline{\mathcal{X}}} = \boldsymbol{I}_{N_p} \otimes \boldsymbol{H}_{\mathcal{X}} \in \mathbb{R}^{r_x N_p \times n N_p}$ and $\boldsymbol{\theta}_{\overline{\mathcal{X}}} = \mathbf{1}_{N_p \times 1} \otimes \boldsymbol{\theta}_{\mathcal{X}} \in \mathbb{R}^{r_x N_p \times 1}$ (respectively $\boldsymbol{H}_{\overline{\mathcal{U}}} = \boldsymbol{I}_{N_p} \otimes \boldsymbol{H}_{\mathcal{U}} \in \mathbb{R}^{r_u N_p \times m N_p}$ and $\boldsymbol{\theta}_{\overline{\mathcal{U}}} = \mathbf{1}_{N_p \times 1} \otimes \boldsymbol{\theta}_{\mathcal{U}} \in \mathbb{R}^{r_u N_p \times 1}$). The set $\overline{\mathcal{V}}_i(k)$ is constructed based on the Voronoi cell $\mathcal{V}_i(k)$ and the invariant set $\Omega_i(k)$ is defined as in Assumption 3.5. If $\boldsymbol{H}_i(k) \in \mathbb{R}^{r_y(k) \times 2}$ and $\boldsymbol{\theta}_i(k) \in \mathbb{R}^{r_y(k)}$ are the matrices inducing $\mathcal{V}_i(k)$ in $\mathbb{R}^2$, $\overline{\boldsymbol{H}}_i(k) = \boldsymbol{I}_{N_p} \otimes \boldsymbol{H}_i(k) \in \mathbb{R}^{r_y(k) N_p \times 2 N_p}$ and:

$$\overline{\boldsymbol{\theta}}_i(k) = \begin{bmatrix} \mathbf{1}_{(N_p-1)\times 1} \otimes \boldsymbol{\theta}_i(k) \\ \boldsymbol{\theta}_i(k) + (1 - \lambda_i(k))\boldsymbol{H}_i(k)\mathbf{c}_i(k) \end{bmatrix} \in \mathbb{R}^{r_y(k) N_p \times 1},$$

with $\lambda_i(k) \in [0, 1)$, are the matrices inducing $\overline{\mathcal{V}}_i(k)$. The last line of $\overline{\boldsymbol{\theta}}_i(k)$ is obtained from the definition of $\Omega_i(k) = \{\mathbf{c}_i(k)\} \oplus \lambda_i(k)(\mathcal{V}_i(k) \oplus \{-\mathbf{c}_i(k)\})$ with Property 2.7 and Property 2.10.

With these parameters, the constraint (4.45b) is meant to ensure that the estimated state over the prediction horizon $\widehat{\mathbf{X}}_i(k)$ remains inside $\overline{\mathcal{X}}$ with a probability greater than or equal to $P_x \in [0, 1]$. The constraint (4.45c) is meant to ensure that the control input over the prediction horizon $\mathbf{U}_i(k)$ remains inside the input set $\overline{\mathcal{U}}$. Finally, the constraint (4.45d) ensures that the estimated position over the prediction horizon $\widehat{\mathbf{Y}}_i(k) = \left(\boldsymbol{I}_{N_p} \otimes \boldsymbol{C}\right)\widehat{\mathbf{X}}_i(k)$ of the agent $i \in \overline{1, N}$ remains inside the set $\overline{\mathcal{V}}_i(k)$ induced by $\overline{\boldsymbol{H}}_i(k)$ and $\overline{\boldsymbol{\theta}}_i(k)$ given above with a probability greater than or equal to $P_y \in [0, 1]$. It can be noticed that if $P_x = 1$, the constraint (4.45b) is an equality constraint. The same way, if $P_y = 1$, the constraint (4.45d) is an equality constraint.

Before investigating further the constraints appearing in problem (4.45), the cost function (4.46) is expressed as a function of the control sequence $\mathbf{U}_i(k)$ and the available information at time k, i.e. $\widehat{\mathbf{X}}_i(k)$, $\boldsymbol{\mu}_{\widehat{\mathbf{x}}_i}(k)$, $\boldsymbol{\mu}_{\boldsymbol{\gamma}_i}(k)$, $\widehat{\mathbf{X}}_{\mathbf{c}_i}(k)$ and $\mathbf{U}_{\mathbf{c}_i}(k)$. Indeed, the norms can be expanded in order to calculate the mathematical expectation. Using (4.42), the expansion of the norms is:

$$\begin{aligned}
\left\|\widehat{\mathbf{X}}_i(k) - \widehat{\mathbf{X}}_{\mathbf{c}_i}(k)\right\|^2_{\overline{\boldsymbol{Q}}_i} &= \left(\widehat{\mathbf{X}}_i(k) - \widehat{\mathbf{X}}_{\mathbf{c}_i}(k)\right)^\top \overline{\boldsymbol{Q}}_i \left(\widehat{\mathbf{X}}_i(k) - \widehat{\mathbf{X}}_{\mathbf{c}_i}(k)\right) \\
&= \mathbf{U}_i(k)^\top \boldsymbol{G}^\top \overline{\boldsymbol{Q}}_i \boldsymbol{G} \mathbf{U}_i(k) \\
&\quad + 2\left(\boldsymbol{F}\widehat{\mathbf{x}}_i(k) + \boldsymbol{G}_{\widetilde{\mathbf{x}}}\widetilde{\mathbf{X}}_i(k) + \boldsymbol{G}_{\boldsymbol{\gamma}}\boldsymbol{\Gamma}_i(k) - \widehat{\mathbf{X}}_{\mathbf{c}_i}(k)\right)^\top \overline{\boldsymbol{Q}}_i \boldsymbol{G} \mathbf{U}_i(k) \\
&\quad + f_x\left(\widehat{\mathbf{x}}_i(k), \widetilde{\mathbf{X}}_i(k), \boldsymbol{\Gamma}_i(k), \widehat{\mathbf{X}}_{\mathbf{c}_i}(k)\right) \\
\|\mathbf{U}_i(k) - \mathbf{U}_{\mathbf{c}_i}(k)\|^2_{\overline{\boldsymbol{R}}_i} &= \left(\mathbf{U}_i(k) - \mathbf{U}_{\mathbf{c}_i}(k)\right)^\top \overline{\boldsymbol{R}}_i \left(\mathbf{U}_i(k) - \mathbf{U}_{\mathbf{c}_i}(k)\right) \\
&= \mathbf{U}_i^\top(k) \overline{\boldsymbol{R}}_i \mathbf{U}_i(k) - 2\mathbf{U}_{\mathbf{c}_i}(k)^\top \overline{\boldsymbol{R}}_i \mathbf{U}_i(k) + f_u(\mathbf{U}_{\mathbf{c}_i}(k)))
\end{aligned}$$

where f_x and f_u are functions of terms that will not be relevant for the optimization process since they are terms that do not depend on the decision variable $\mathbf{U}_i(k)$.

Assumption 4.6: Invariance of the stochastic properties over the prediction horizon
The mean and variance matrix of the noise signals $\boldsymbol{\delta}_i$ and $\boldsymbol{\gamma}_i$ are constant over the prediction horizon.

Taking the expectation of the previous norms considering Assumption 4.6, the cost function (4.46) is then reduced to:

$$\begin{aligned} &J\Big(\widehat{\mathbf{X}}_i(k), \mathbf{U}_i(k), \widehat{\mathbf{X}}_{\mathbf{c}_i}(k), \mathbf{U}_{\mathbf{c}_i}(k)\Big) \\ &\quad = \mathbf{U}_i(k)^\top \big(\boldsymbol{G}^\top \overline{\boldsymbol{Q}}_i \boldsymbol{G} + \overline{\boldsymbol{R}}_i\big)\mathbf{U}_i(k) - 2\mathbf{U}_{\mathbf{c}_i}(k)^\top \overline{\boldsymbol{R}}_i \mathbf{U}_i(k) \\ &\qquad + 2\Big(\boldsymbol{F}\widehat{\mathbf{x}}_i(k) + \boldsymbol{G}_{\tilde{\mathbf{x}}}\boldsymbol{\mu}_{\tilde{\mathbf{X}}_i}(k) + \boldsymbol{G}_{\boldsymbol{\gamma}}\boldsymbol{\mu}_{\boldsymbol{\Gamma}_i}(k) - \widehat{\mathbf{X}}_{\mathbf{c}_i}(k)\Big)^\top \overline{\boldsymbol{Q}}_i \boldsymbol{G}\mathbf{U}_i(k) \\ &\qquad + f_x\Big(\widehat{\mathbf{x}}_i(k), \boldsymbol{\mu}_{\tilde{\mathbf{X}}_i}(k), \boldsymbol{\mu}_{\boldsymbol{\Gamma}_i}(k), \widehat{\mathbf{X}}_{\mathbf{c}_i}(k)\Big) + f_u(\mathbf{U}_{\mathbf{c}_i}(k)) \end{aligned} \tag{4.47}$$

where $\boldsymbol{\mu}_{\boldsymbol{\Gamma}_i}(k) = \mathbf{1}_{N_p \times 1} \otimes \boldsymbol{\mu}_{\boldsymbol{\gamma}_i}(k)$ is the mean of the discrete-time measurement noise over the prediction horizon and:

$$\boldsymbol{\mu}_{\tilde{\mathbf{X}}_i}(k) = \begin{bmatrix} \boldsymbol{\mu}_{\tilde{\mathbf{x}}_i}(k) \\ \vdots \\ \boldsymbol{\mu}_{\tilde{\mathbf{x}}_i}(k + N_p - 1) \end{bmatrix}$$

is the mean of the state estimation error over the prediction horizon. In the vector $\boldsymbol{\mu}_{\tilde{\mathbf{X}}_i}(k)$, each individual mean $\boldsymbol{\mu}_{\tilde{\mathbf{x}}_i}(k+l+1)$ for all $l \in \overline{0, N_p - 1}$ is calculated recursively with (4.40) from $\boldsymbol{\mu}_{\tilde{\mathbf{x}}_i}(k)$. Indeed, $\boldsymbol{\mu}_{\tilde{\mathbf{x}}_i}(k)$ is known and, with Assumption 4.6, $\boldsymbol{\mu}_{\boldsymbol{\delta}_i}(k+l) = \boldsymbol{\mu}_{\boldsymbol{\delta}_i}(k)$ and $\boldsymbol{\mu}_{\boldsymbol{\gamma}_i}(k+l) = \boldsymbol{\mu}_{\boldsymbol{\gamma}_i}(k)$ for all $l \in \overline{0, N_p - 1}$, $\boldsymbol{\mu}_{\boldsymbol{\delta}_i}(k)$ and $\boldsymbol{\mu}_{\boldsymbol{\gamma}_i}(k)$ being known.

Finally, since the functions f_x and f_u in (4.47) are fixed by the system properties, they are constant and are not minimized. They can then be removed from the expression of the cost function. Thus, the cost function to be minimized is:

$$\begin{aligned} &J\Big(\widehat{\mathbf{X}}_i(k), \mathbf{U}_i(k), \widehat{\mathbf{X}}_{\mathbf{c}_i}(k), \mathbf{U}_{\mathbf{c}_i}(k)\Big) \\ &\quad = \mathbf{U}_i(k)^\top \big(\boldsymbol{G}^\top \overline{\boldsymbol{Q}}_i \boldsymbol{G} + \overline{\boldsymbol{R}}_i\big)\mathbf{U}_i(k) - 2\mathbf{U}_{\mathbf{c}_i}(k)^\top \overline{\boldsymbol{R}}_i \mathbf{U}_i(k) \\ &\qquad + 2\Big(\boldsymbol{F}\widehat{\mathbf{x}}_i(k) + \boldsymbol{G}_{\tilde{\mathbf{x}}}\boldsymbol{\mu}_{\tilde{\mathbf{X}}_i}(k) + \boldsymbol{G}_{\boldsymbol{\gamma}}\boldsymbol{\mu}_{\boldsymbol{\Gamma}_i}(k) - \widehat{\mathbf{X}}_{\mathbf{c}_i}(k)\Big)^\top \overline{\boldsymbol{Q}}_i \boldsymbol{G}\mathbf{U}_i(k). \end{aligned} \tag{4.48}$$

With (4.48), the cost function that is minimized by the optimization problem (4.45) do not contain any unknown stochastic term. However, the probabilistic constraints (4.45b) and (4.45d) prevent the problem to be easily solvable. As explained before, such constraints have to be relaxed into algebraic constraints to end up with a classical optimization problem that can be solved with a quadratic programming solver. Such a relaxation is the purpose of Paragraph 4.2.3.2.

4.2.3.2 Relaxation of the probabilistic constraints into algebraic constraints

In order to be able to formulate the constraints (4.45b) and (4.45d) as algebraic constraints, it is first necessary to express them as a function of the different signals acting on the system (4.38).

With the notations introduced in Paragraph 4.2.3.1, it is possible to rewrite the constraint (4.45b) as a function of the different signals acting on the system. The

event which probability is studied is the event $\widehat{\mathbf{X}}_i(k) \in \overline{\mathcal{X}}$. Such an event can be rewritten, with (4.42), as the inequality:

$$\begin{aligned} &\boldsymbol{H}_{\overline{\mathcal{X}}}\widehat{\mathbf{X}}_i(k) \leq \boldsymbol{\theta}_{\overline{\mathcal{X}}} \\ &\quad \Leftrightarrow \boldsymbol{H}_{\overline{\mathcal{X}}}\boldsymbol{G}\mathbf{U}_i(k) \leq \boldsymbol{\theta}_{\overline{\mathcal{X}}} - \boldsymbol{H}_{\overline{\mathcal{X}}}\Big(\boldsymbol{F}\widehat{\mathbf{x}}_i(k) + \boldsymbol{G}_{\widetilde{\mathbf{x}}}\widetilde{\mathbf{X}}_i(k) + \boldsymbol{G}_{\boldsymbol{\gamma}}\boldsymbol{\Gamma}_i(k)\Big). \end{aligned} \tag{4.49}$$

The same way, the event which probability is studied in (4.45d) is the event $\big(\boldsymbol{I}_{N_p} \otimes \boldsymbol{C}\big)\widehat{\mathbf{X}}_i(k) \in \overline{\mathcal{V}}_i(k)$ which can be rewritten as the inequality:

$$\begin{aligned} &\overline{\boldsymbol{H}}_i(k)\overline{\boldsymbol{C}}\widehat{\mathbf{X}}_i(k) \leq \overline{\boldsymbol{\theta}}_i(k) \\ &\quad \Leftrightarrow \overline{\boldsymbol{H}}_i(k)\overline{\boldsymbol{C}}\boldsymbol{G}\mathbf{U}_i(k) \leq \overline{\boldsymbol{\theta}}_i(k) - \overline{\boldsymbol{H}}_i(k)\overline{\boldsymbol{C}}\Big(\boldsymbol{F}\widehat{\mathbf{x}}_i(k) + \boldsymbol{G}_{\widetilde{\mathbf{x}}}\widetilde{\mathbf{X}}_i(k) + \boldsymbol{G}_{\boldsymbol{\gamma}}\boldsymbol{\Gamma}_i(k)\Big) \end{aligned} \tag{4.50}$$

where $\overline{\boldsymbol{C}} = \boldsymbol{I}_{N_p} \otimes \boldsymbol{C}$. In both inequalities (4.49) and (4.50), a stochastic term $\boldsymbol{\Xi}_i(k) = \begin{bmatrix}\widetilde{\mathbf{X}}_i(k)^\top & \boldsymbol{\Gamma}_i(k)^\top\end{bmatrix}^\top \in \mathbb{R}^{(2+n)N_p}$ appears.

Theorem 4.1: Satisfaction of the constraints (Cannon et al., 2012)

At time k, the constraints (4.45b) and (4.45d) are satisfied if and only if $\mathbf{U}_i(k)$ satisfies:

$$\boldsymbol{H}_{\overline{\mathcal{X}}}\boldsymbol{G}\mathbf{U}_i(k) \leq \boldsymbol{\theta}_{\overline{\mathcal{X}}} - \boldsymbol{H}_{\overline{\mathcal{X}}}\boldsymbol{F}\widehat{\mathbf{x}}_i(k) - \mathbf{b}_x(k) \tag{4.51}$$

$$\overline{\boldsymbol{H}}_i(k)\overline{\boldsymbol{C}}\boldsymbol{G}\mathbf{U}_i(k) \leq \overline{\boldsymbol{\theta}}_i(k) - \overline{\boldsymbol{H}}_i(k)\overline{\boldsymbol{C}}\boldsymbol{F}\widehat{\mathbf{x}}_i(k) - \mathbf{b}_y(k) \tag{4.52}$$

where $\mathbf{b}_x(k) \in \mathbb{R}^{r_x N_p}$ and $\mathbf{b}_y(k) \in \mathbb{R}^{r_y(k) N_p}$ are defined as the minimum values such that, respectively:

$$\mathbb{P}\big(\boldsymbol{H}_{\overline{\mathcal{X}}}\begin{bmatrix}\boldsymbol{G}_{\widetilde{\mathbf{x}}} & \boldsymbol{G}_{\boldsymbol{\gamma}}\end{bmatrix}\boldsymbol{\Xi}_i(k) \leq \mathbf{b}_x(k)\big) = P_x \tag{4.53}$$

$$\mathbb{P}\big(\overline{\boldsymbol{H}}_i(k)\overline{\boldsymbol{C}}\begin{bmatrix}\boldsymbol{G}_{\widetilde{\mathbf{x}}} & \boldsymbol{G}_{\boldsymbol{\gamma}}\end{bmatrix}\boldsymbol{\Xi}_i(k) \leq \mathbf{b}_y(k)\big) = P_y. \tag{4.54}$$

In the general case, the bounds $\mathbf{b}_x$ and $\mathbf{b}_y$ satisfying (4.53) and (4.54) are not easy to compute. However, here, given Assumption 4.3, the stochastic term $\boldsymbol{\Xi}_i(k)$ is defined from normally distributed signals. Thus, $\boldsymbol{\Xi}_i(k)$ is normally distributed. With such a property, it would be possible to find the bound $\mathbf{b}_x = \begin{bmatrix}b_{x,1} & \cdots & b_{x,r_x N_p}\end{bmatrix}^\top$ by solving:

$$\underset{\mathbf{b}_x(k)}{\text{minimize}} \quad f(\mathbf{b}_x(k)) \tag{4.55a}$$

subject to

$$\mathbb{P}\big(\boldsymbol{H}_{\overline{\mathcal{X}}}\begin{bmatrix}\boldsymbol{G}_{\widetilde{\mathbf{x}}} & \boldsymbol{G}_{\boldsymbol{\gamma}}\end{bmatrix}\boldsymbol{\Xi}_i(k) \leq \mathbf{b}_x(k)\big) = P_x \tag{4.55b}$$

where:

$$\begin{aligned} &\mathbb{P}\big(\boldsymbol{H}_{\overline{\mathcal{X}}}\begin{bmatrix}\boldsymbol{G}_{\widetilde{\mathbf{x}}} & \boldsymbol{G}_{\boldsymbol{\gamma}}\end{bmatrix}\boldsymbol{\Xi}_i(k) \leq \mathbf{b}_x(k)\big) \\ &\qquad = \int_{-\infty}^{b_{x,r_x N_p}(k)} \cdots \int_{-\infty}^{b_{x,1}(k)} f_{\boldsymbol{\Xi},x}(x_1, \ldots, x_{r_x N_p}) \mathrm{d}x_1 \cdots \mathrm{d}x_{r_x N_p} \end{aligned}$$

where $f_{\boldsymbol{\Xi},x}$ is the probability density function of the random variable $\boldsymbol{H}_{\overline{\mathcal{X}}}\boldsymbol{G}_{\boldsymbol{\Xi}}\boldsymbol{\Xi}_i(k)$, with $\boldsymbol{G}_{\boldsymbol{\Xi}} = \begin{bmatrix}\boldsymbol{G}_{\tilde{\mathbf{x}}} & \boldsymbol{G}_{\boldsymbol{\gamma}}\end{bmatrix}$. The bound $\mathbf{b}_y(k)$ can be found by solving a problem similar to (4.55), replacing $\mathbf{b}_x(k)$ by $\mathbf{b}_y(k)$, $\boldsymbol{H}_{\overline{\mathcal{X}}}\begin{bmatrix}\boldsymbol{G}_{\tilde{\mathbf{x}}} & \boldsymbol{G}_{\boldsymbol{\gamma}}\end{bmatrix}$ by $\overline{\boldsymbol{H}}_i\overline{\boldsymbol{C}}\begin{bmatrix}\boldsymbol{G}_{\tilde{\mathbf{x}}} & \boldsymbol{G}_{\boldsymbol{\gamma}}\end{bmatrix}$ and P_x by P_y.

However, a problem such as (4.55) is difficult to solve. Usually (Farina et al., 2016), the probabilistic constraints are separated such that (4.53) (the same can be said for (4.54)) would rather be:

$$\mathbb{P}(\mathbf{e}_j\boldsymbol{H}_{\overline{\mathcal{X}}}\boldsymbol{G}_{\boldsymbol{\Xi}}\boldsymbol{\Xi}_i(k) \leq \mathbf{e}_j\mathbf{b}_x(k)) = P_x$$

where $\mathbf{e}_j \in \mathbb{R}^{1\times r_x N_p}$, with $j \in \overline{1, r_x N_p}$, is a vector of the canonical basis of $\mathbb{R}^{r_x N_p}$ such that:

$$\mathbf{e}_1 = \begin{bmatrix}1 & 0 & \cdots & 0\end{bmatrix} \qquad \cdots \qquad \mathbf{e}_{r_x N_p} = \begin{bmatrix}0 & \cdots & 0 & 1\end{bmatrix}.$$

Remark 4.8: Index
In the following it is implicitly assumed that when $\mathbf{e}_j$ is used, j is in the range $\overline{1, r_x N_p}$ unless stated otherwise. ♢

A way to avoid the complexity of solving the previous optimization problem (4.55), albeit conservative, is found in Gavilan et al. (2012) or Chevet et al. (2020a). It consists in setting $P_x = P_y = 1$ such that the constraint is satisfied for all perturbations. The constraints (4.45b) and (4.45d) are then equality constraints since a probability cannot be higher than 1. Then, the constraint (4.45b) can be relaxed by taking:

$$-\mathbf{e}_j\mathbf{b}_x(k) = \min_{\boldsymbol{\Xi}_i(k)\in\mathcal{D}_{\boldsymbol{\Xi}}} -\mathbf{e}_j\boldsymbol{H}_{\overline{\mathcal{X}}}\boldsymbol{G}_{\boldsymbol{\Xi}}\boldsymbol{\Xi}_i(k) \tag{4.56}$$

with $j \in \overline{1, r_x N_p}$, where $\mathcal{D}_{\boldsymbol{\Xi}}$ is the support of the distribution of the random variable $\boldsymbol{\Xi}_i$. The same can be done to relax (4.45d). For the minimum (4.56) to exist, the support $\mathcal{D}_{\boldsymbol{\Xi}}$ has to be bounded. Here, since $\boldsymbol{\Xi}_i$ follows a multivariate normal distribution, the support is unbounded. Thus, to obtain the bounds $\mathbf{b}_x(k)$ and $\mathbf{b}_y(k)$, the problem (4.56) has to be modified.

Since $\boldsymbol{\Xi}_i(k)$ is normally distributed with mean $\boldsymbol{\mu}_{\boldsymbol{\Xi}_i}(k)$ and variance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\Xi}_i}(k)$, the random variable:

$$\xi_i(k) = (\boldsymbol{\Xi}_i(k) - \boldsymbol{\mu}_{\boldsymbol{\Xi}_i}(k))^\top \boldsymbol{\Sigma}_{\boldsymbol{\Xi}_i}^{-1}(k)(\boldsymbol{\Xi}_i(k) - \boldsymbol{\mu}_{\boldsymbol{\Xi}_i}(k)) \tag{4.57}$$

follows a chi-squared law with $(n+2)N_p$ degrees of freedom according to Property 2.13, denoted by $\chi^2_{(n+2)N_p}$, since $\boldsymbol{\Xi}_i(k) \in \mathbb{R}^{(n+2)N_p}$.

Since the chi-squared distribution is often used in statistical testing, tables exist (Elderton, 1902) to find the value of the parameter α such that:

$$\mathbb{P}(\xi_i(k) \leq \alpha) = P_\chi \tag{4.58}$$

with $P_\chi \in [0, 1]$. With this parameter α, it is then guaranteed with probability P_χ that all the random vectors $\boldsymbol{\Xi}_i(k)$ are inside the ellipsoid $\mathscr{E}(\boldsymbol{\Sigma}_{\boldsymbol{\Xi}_i}(k)^{-1}, \boldsymbol{\mu}_{\boldsymbol{\Xi}_i}(k), \alpha)$. To ensure that most of the perturbations are covered by the bounds $\mathbf{b}_x(k)$ and $\mathbf{b}_y(k)$, it is preferable to chose P_χ as close to 1 as possible. In the following, only the procedure to find $\mathbf{b}_x(k)$ is treated, this procedure being immediately transposable to $\mathbf{b}_y(k)$.

With the knowledge of the bounding ellipsoid $\mathscr{E}(\boldsymbol{\Sigma}_{\boldsymbol{\Xi}_i}(k)^{-1}, \boldsymbol{\mu}_{\boldsymbol{\Xi}_i}(k), \alpha)$, the problem (4.56) is changed into:

$$-\mathbf{e}_j \mathbf{b}_x(k) = \underset{\boldsymbol{\Xi}_i(k)}{\text{minimize}} \quad -\mathbf{e}_j \boldsymbol{H}_{\overline{\mathcal{X}}} \boldsymbol{G}_{\boldsymbol{\Xi}} \boldsymbol{\Xi}_i(k) \tag{4.59a}$$

$$\text{subject to}$$

$$(\boldsymbol{\Xi}_i(k) - \boldsymbol{\mu}_{\boldsymbol{\Xi}_i}(k))^\top \boldsymbol{\Sigma}_{\boldsymbol{\Xi}_i}^{-1}(k)(\boldsymbol{\Xi}_i(k) - \boldsymbol{\mu}_{\boldsymbol{\Xi}_i}(k)) \leq \alpha. \tag{4.59b}$$

Despite the problem (4.59) being formulated, the positive definiteness of $\boldsymbol{\Sigma}_{\boldsymbol{\Xi}_i}(k)$ has been implicitly assumed since the definition of $\xi_i(k)$. A formal proof of such a property has to be drawn.

Property 4.3: Positive definiteness of the variance matrix of $\boldsymbol{\Xi}_i$

The variance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\Xi}_i}(k)$ is positive definite for all $k > 0$.

Proof. By definition:

$$\boldsymbol{\Sigma}_{\boldsymbol{\Xi}_i}(k) = \mathbb{E}\big(\boldsymbol{\Xi}_i(k)\boldsymbol{\Xi}_i(k)^\top\big) - \mathbb{E}(\boldsymbol{\Xi}_i(k))\mathbb{E}(\boldsymbol{\Xi}_i(k))^\top.$$

Considering $\boldsymbol{\Xi}_i(k) = \begin{bmatrix}\widetilde{\mathbf{X}}_i(k)^\top & \boldsymbol{\Gamma}_i(k)^\top\end{bmatrix}^\top$, with $\widetilde{\mathbf{X}}_i(k)$ and $\boldsymbol{\Gamma}_i(k)$ the estimation error and the measurement noise, respectively, which are random variables, over the prediction horizon:

$$\boldsymbol{\Sigma}_{\boldsymbol{\Xi}_i}(k) = \begin{bmatrix} \boldsymbol{\Sigma}_{\widetilde{\mathbf{X}}_i}(k) & \operatorname{cov}\Big(\widetilde{\mathbf{X}}_i(k), \boldsymbol{\Gamma}_i(k)\Big) \\ \operatorname{cov}\Big(\widetilde{\mathbf{X}}_i(k), \boldsymbol{\Gamma}_i(k)\Big)^\top & \boldsymbol{\Sigma}_{\boldsymbol{\Gamma}_i}(k) \end{bmatrix}$$

where $\boldsymbol{\Sigma}_{\widetilde{\mathbf{X}}_i}(k) \in \mathbb{R}^{(n+2)N_p \times (n+2)N_p}$ is described below in (4.60), $\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}_i}(k) = \boldsymbol{I}_{N_p} \otimes \boldsymbol{\Sigma}_{\boldsymbol{\gamma}_i}(k) \in \mathbb{R}^{2N_p \times 2N_p}$ and:

$$\operatorname{cov}\Big(\widetilde{\mathbf{X}}_i(k), \boldsymbol{\Gamma}_i(k)\Big) = \mathbb{E}\Big(\widetilde{\mathbf{X}}_i(k)\boldsymbol{\Gamma}_i(k)^\top\Big) - \mathbb{E}\Big(\widetilde{\mathbf{X}}_i(k)\Big)\mathbb{E}(\boldsymbol{\Gamma}_i(k))^\top.$$

The expression of the covariance matrix of $\widetilde{\mathbf{X}}_i(k)$ can be deduced from (4.43). Remark 4.6 states that $\widetilde{\mathbf{x}}_i(k-1)$, $\boldsymbol{\Gamma}_i(k-1)$ and $\boldsymbol{\Delta}_i(k-1)$ are independent, yielding null covariances. Thus the variance matrix $\boldsymbol{\Sigma}_{\widetilde{\mathbf{X}}_i}(k)$ is such that:

$$\boldsymbol{\Sigma}_{\widetilde{\mathbf{X}}_i}(k) = \widetilde{\boldsymbol{F}}\boldsymbol{\Sigma}_{\widetilde{\mathbf{x}}_i}(k-1)\widetilde{\boldsymbol{F}}^\top + \widetilde{\boldsymbol{G}}_{\boldsymbol{\gamma}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}_i}(k-1)\widetilde{\boldsymbol{G}}_{\boldsymbol{\gamma}}^\top + \widetilde{\boldsymbol{G}}_{\boldsymbol{\delta}}\boldsymbol{\Sigma}_{\boldsymbol{\Delta}_i}(k-1)\widetilde{\boldsymbol{G}}_{\boldsymbol{\delta}}^\top, \tag{4.60}$$

where:

$$\boldsymbol{\Sigma}_{\boldsymbol{\Delta}_i}(k-1) = \operatorname{diag}\big(\boldsymbol{\Sigma}_{\boldsymbol{\delta}_i}(k-1), \boldsymbol{I}_{N_p-1} \otimes \boldsymbol{\Sigma}_{\boldsymbol{\delta}_i}(k)\big) \in \mathbb{R}^{nN_p \times nN_p} \tag{4.61}$$

and:

$$\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}_i}(k-1) = \operatorname{diag}\big(\boldsymbol{\Sigma}_{\boldsymbol{\gamma}_i}(k-1), \boldsymbol{I}_{N_p-1} \otimes \boldsymbol{\Sigma}_{\boldsymbol{\gamma}_i}(k)\big) \in \mathbb{R}^{2N_p \times 2N_p}. \tag{4.62}$$

Indeed, $\boldsymbol{\Delta}_i(k-1) = \begin{bmatrix}\boldsymbol{\delta}_i(k-1)^\top & \boldsymbol{\delta}_i(k)^\top & \cdots & \boldsymbol{\delta}_i(k+N_p-2)^\top\end{bmatrix}^\top$. Then, from Assumption 4.3, the $\boldsymbol{\delta}_i(k-1+l)$ are independent for all $l \in \overline{0, N_p-1}$. The variance matrix of $\boldsymbol{\Delta}_i(k-1)$ is then:

$$\boldsymbol{\Sigma}_{\boldsymbol{\Delta}_i}(k-1) = \operatorname{diag}(\boldsymbol{\Sigma}_{\boldsymbol{\delta}_i}(k-1), \boldsymbol{\Sigma}_{\boldsymbol{\delta}_i}(k), \ldots, \boldsymbol{\Sigma}_{\boldsymbol{\delta}_i}(k+N_p-2)).$$

However, with Assumption 4.6, the variance matrix of $\boldsymbol{\delta}_i(k+l)$ is considered constant for all $l \in \overline{0, N_p - 1}$. Then, $\boldsymbol{\Sigma}_{\boldsymbol{\delta}_i}(k) = \boldsymbol{\Sigma}_{\boldsymbol{\delta}_i}(k+1) = \cdots = \boldsymbol{\Sigma}_{\boldsymbol{\delta}_i}(k+N_p-1)$, hence the expression given in (4.61). By replacing $\boldsymbol{\delta}_i$ by $\boldsymbol{\gamma}_i$, the same reasoning leads to the expression of $\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}_i}(k-1)$ given in (4.62).

The only term that remains to be calculated in the variance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\Xi}_i}(k)$ is $\operatorname{cov}\Big(\widetilde{\mathbf{X}}_i(k), \boldsymbol{\Gamma}_i(k)\Big)$. In the expression of $\widetilde{\mathbf{X}}_i(k)$ given in (4.43), all the terms are independent of $\boldsymbol{\Gamma}_i(k)$, leading to null covariances, except the one term depending on $\boldsymbol{\Gamma}_i(k-1)$. Then, the covariance appearing in $\boldsymbol{\Sigma}_{\boldsymbol{\Xi}_i}(k)$ is such that:

$$\begin{aligned}
\operatorname{cov}\Big(\widetilde{\mathbf{X}}_i(k), \boldsymbol{\Gamma}_i(k)\Big) &= \operatorname{cov}\Big(-\widetilde{\boldsymbol{G}}_{\boldsymbol{\gamma}}\boldsymbol{\Gamma}_i(k-1), \boldsymbol{\Gamma}_i(k)\Big) \\
&= -\widetilde{\boldsymbol{G}}_{\boldsymbol{\gamma}}\operatorname{cov}(\boldsymbol{\Gamma}_i(k-1), \boldsymbol{\Gamma}_i(k)) \\
&= -\widetilde{\boldsymbol{G}}_{\boldsymbol{\gamma}}\mathbb{E}\left(\begin{bmatrix} \boldsymbol{\gamma}_i(k-1) \\ \vdots \\ \boldsymbol{\gamma}_i(k+N_p-2) \end{bmatrix} \begin{bmatrix} \boldsymbol{\gamma}_i(k)^\top & \cdots & \boldsymbol{\gamma}_i(k+N_p-1)^\top \end{bmatrix}\right).
\end{aligned}$$

From Assumption 4.3, $\mathbb{E}\big(\boldsymbol{\gamma}_i(k+a)\boldsymbol{\gamma}_i(k+b)^\top\big) = \boldsymbol{\Sigma}_{\boldsymbol{\gamma}_i}(k)\delta(a-b)$, where δ is the Dirac delta function. Then, it leads to:

$$\operatorname{cov}\Big(\widetilde{\mathbf{X}}_i(k), \boldsymbol{\Gamma}_i(k)\Big) = -\widetilde{\boldsymbol{G}}_{\boldsymbol{\gamma}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}_i,L}(k)$$

where:

$$\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}_i,L}(k) = \begin{bmatrix} \mathbf{0}_2 & \cdots & \cdots & \cdots & \mathbf{0}_2 \\ \boldsymbol{\Sigma}_{\boldsymbol{\gamma}_i}(k) & \ddots & & & \vdots \\ \mathbf{0}_2 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \mathbf{0}_2 & \cdots & \mathbf{0}_2 & \boldsymbol{\Sigma}_{\boldsymbol{\gamma}_i}(k) & \mathbf{0}_2 \end{bmatrix} \in \mathbb{R}^{2N_p \times 2N_p}.$$

The variance matrix of $\boldsymbol{\Xi}_i(k)$ is then finally:

$$\boldsymbol{\Sigma}_{\boldsymbol{\Xi}_i}(k) = \begin{bmatrix} \boldsymbol{\Sigma}_{\widetilde{\mathbf{X}}_i}(k) & -\widetilde{\boldsymbol{G}}_{\boldsymbol{\gamma}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}_i,L}(k) \\ -\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}_i,L}(k)^\top\widetilde{\boldsymbol{G}}_{\boldsymbol{\gamma}}^\top & \boldsymbol{\Sigma}_{\boldsymbol{\Gamma}_i}(k) \end{bmatrix}. \tag{4.63}$$

Consider the matrix:

$$\boldsymbol{S}(k) = \boldsymbol{\Sigma}_{\widetilde{\mathbf{X}}_i}(k) - \widetilde{\boldsymbol{G}}_{\boldsymbol{\gamma}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}_i,L}(k)\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}_i}^{-1}(k)\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}_i,L}(k)^\top\widetilde{\boldsymbol{G}}_{\boldsymbol{\gamma}}^\top \tag{4.64}$$

obtained by using the Schur complement (2.7) on the matrix $\boldsymbol{\Sigma}_{\boldsymbol{\Xi}_i}(k)$ in (4.63). The variance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\Xi}_i}(k)$ is positive definite if and only if $\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}_i}(k) \succ 0$ and $\boldsymbol{S}(k) \succ 0$ by Property 2.3. It then remains to prove that $\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}_i}(k) \succ 0$ and $\boldsymbol{S}(k) \succ 0$.

The variance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}_i}(k)$ given by (4.62) is positive definite (thus invertible) since $\boldsymbol{\Sigma}_{\boldsymbol{\gamma}_i}(k) \succ 0$ for all $k > 0$. Moreover, since $\boldsymbol{\Sigma}_{\boldsymbol{\gamma}_i}(k)$ is diagonal, its inverse is the diagonal matrix $\boldsymbol{\Sigma}_{\boldsymbol{\gamma}_i}^{-1}(k)$, and $\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}_i}^{-1}(k) = \boldsymbol{I}_{N_p} \otimes \boldsymbol{\Sigma}_{\boldsymbol{\gamma}_i}^{-1}(k)$. Then:

$$\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}_i,L}(k)\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}_i}^{-1}(k)\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}_i,L}(k)^\top = \begin{bmatrix} \mathbf{0}_2 & \mathbf{0}_2 & \cdots & \cdots & \mathbf{0}_2 \\ \mathbf{0}_2 & \boldsymbol{\Sigma}_{\boldsymbol{\gamma}_i}(k) & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \mathbf{0}_2 \\ \mathbf{0}_2 & \cdots & \cdots & \mathbf{0}_2 & \boldsymbol{\Sigma}_{\boldsymbol{\gamma}_i}(k) \end{bmatrix}$$

since $\boldsymbol{\Sigma}_{\boldsymbol{\gamma}_i}(k)^\top = \boldsymbol{\Sigma}_{\boldsymbol{\gamma}_i}(k)$, and, by injecting this result and (4.60) into (4.64):

$$\boldsymbol{S}(k) = \widetilde{\boldsymbol{F}}\boldsymbol{\Sigma}_{\widetilde{\mathbf{x}}_i}(k-1)\widetilde{\boldsymbol{F}}^\top + \widetilde{\boldsymbol{G}}_{\boldsymbol{\gamma}_i}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}_i,1}(k-1)\widetilde{\boldsymbol{G}}_{\boldsymbol{\gamma}}^\top + \widetilde{\boldsymbol{G}}_{\boldsymbol{\delta}}\boldsymbol{\Sigma}_{\boldsymbol{\Delta}_i}(k-1)\widetilde{\boldsymbol{G}}_{\boldsymbol{\delta}}^\top$$

where:

$$\begin{aligned}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}_i,1}(k-1) &= \boldsymbol{\Sigma}_{\boldsymbol{\Gamma}_i}(k-1) - \boldsymbol{\Sigma}_{\boldsymbol{\Gamma}_i,L}(k)\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}_i}^{-1}(k)\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}_i,L}(k)^\top \\ &= \operatorname{diag}\big(\boldsymbol{\Sigma}_{\boldsymbol{\gamma}_i}(k-1), \mathbf{0}_{2(N_p-1)\times 2(N_p-1)}\big).\end{aligned}$$

Property 4.2 gives $\boldsymbol{\Sigma}_{\widetilde{\mathbf{x}}_i}(k-1) \succ 0$, thus $\widetilde{\boldsymbol{F}}\boldsymbol{\Sigma}_{\widetilde{\mathbf{x}}_i}(k-1)\widetilde{\boldsymbol{F}}^\top \succeq 0$ according to Property 2.2. By definition, $\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}_i,1}(k-1) \succeq 0$ thus, with Property 2.2, it leads to $\widetilde{\boldsymbol{G}}_{\boldsymbol{\gamma}}\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}_i,1}(k-1)\widetilde{\boldsymbol{G}}_{\boldsymbol{\gamma}}^\top \succeq 0$. To prove that $\boldsymbol{S}(k) \succ 0$, it remains to prove that $\widetilde{\boldsymbol{G}}_{\boldsymbol{\delta}}\boldsymbol{\Sigma}_{\boldsymbol{\Delta}_i}(k-1)\widetilde{\boldsymbol{G}}_{\boldsymbol{\delta}}^\top \succ 0$.

By definition, $\widetilde{\boldsymbol{G}}_{\boldsymbol{\delta}} = \boldsymbol{I}_{nN_p} + \boldsymbol{T}$, where $\boldsymbol{T}$ is the strictly lower block triangular matrix:

$$\boldsymbol{T} = \begin{bmatrix} \mathbf{0}_n & \cdots & \cdots & \mathbf{0}_n \\ \boldsymbol{A}-\boldsymbol{LC} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ (\boldsymbol{A}-\boldsymbol{LC})^{N_p-1} & \cdots & \boldsymbol{A}-\boldsymbol{LC} & \mathbf{0}_n \end{bmatrix}$$

and:

$$\begin{aligned}\widetilde{\boldsymbol{G}}_{\boldsymbol{\delta}_i}\boldsymbol{\Sigma}_{\boldsymbol{\Delta}_i}(k-1)\widetilde{\boldsymbol{G}}_{\boldsymbol{\delta}_i}^\top &= \big(\boldsymbol{I}_{nN_p}+\boldsymbol{T}\big)\boldsymbol{\Sigma}_{\boldsymbol{\Delta}_i}(k-1)\big(\boldsymbol{I}_{nN_p}+\boldsymbol{T}\big)^\top \\ &= \boldsymbol{\Sigma}_{\boldsymbol{\Delta}_i}(k-1) + \boldsymbol{T}\boldsymbol{\Sigma}_{\boldsymbol{\Delta}_i}(k-1) \\ &\qquad + \boldsymbol{\Sigma}_{\boldsymbol{\Delta}_i}(k-1)\boldsymbol{T}^\top + \boldsymbol{T}\boldsymbol{\Sigma}_{\boldsymbol{\Delta}_i}(k-1)\boldsymbol{T}^\top.\end{aligned}$$

Since $\boldsymbol{\Sigma}_{\boldsymbol{\delta}_i}(k-1) \succ 0$ and $\boldsymbol{\Sigma}_{\boldsymbol{\delta}_i}(k) \succ 0$, the variance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\Delta}_i}(k-1)$ is positive definite. The products $\boldsymbol{T}\boldsymbol{\Sigma}_{\boldsymbol{\Delta}_i}(k-1)$ and $\boldsymbol{\Sigma}_{\boldsymbol{\Delta}_i}(k-1)\boldsymbol{T}^\top$ are strictly lower and upper block triangular matrices, respectively, their eigenvalues are then their diagonal elements. Then, the matrices $\boldsymbol{T}\boldsymbol{\Sigma}_{\boldsymbol{\Delta}_i}(k-1)$ and $\boldsymbol{\Sigma}_{\boldsymbol{\Delta}_i}(k-1)\boldsymbol{T}^\top$ are positive semidefinite. By Property 2.2, $\boldsymbol{T}\boldsymbol{\Sigma}_{\boldsymbol{\Delta}_i}(k-1)\boldsymbol{T}^\top \succeq 0$.

Finally, it is proven that $\boldsymbol{S}(k)$ is positive definite as the sum of positive semidefinite matrices and at least one positive definite matrix. Moreover, $\boldsymbol{\Sigma}_{\boldsymbol{\Gamma}_i}(k)$ is positive definite by definition since $\boldsymbol{\Sigma}_{\boldsymbol{\gamma}_i}(k)$ is positive definite. Then, it is proven from Property 2.3 that the variance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\Xi}_i}(k)$ of $\boldsymbol{\Xi}_i(k)$ is positive definite for all $k > 0$. ■

Under the assumptions of this section, the problem (4.59) is feasible. Based on the method from Gavilan et al. (2012), an explicit solution to (4.59) is proposed.

Let $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_{\boldsymbol{\Xi}_i}(k)^{-1}/\alpha$ and $\mathbf{z}_i(k) = \boldsymbol{\Sigma}^{1/2}(\boldsymbol{\Xi}_i(k) - \boldsymbol{\mu}_{\boldsymbol{\Xi}_i}(k))$. Since $\boldsymbol{\Sigma}$ is symmetric by definition, $\boldsymbol{\Sigma}^{1/2}$ is symmetric and $\boldsymbol{\Sigma}^{-1/2}$ is also symmetric. Then, the optimization problem (4.59) can be rewritten:

$$\begin{aligned}-\mathbf{e}_j\mathbf{b}_x(k) = \underset{\mathbf{z}_i(k)}{\text{minimize}} \quad & -\mathbf{e}_j\boldsymbol{H}_{\overline{\mathcal{X}}}\boldsymbol{G}_{\boldsymbol{\Xi}}\big(\boldsymbol{\Sigma}^{-1/2}\mathbf{z}_i(k) + \boldsymbol{\mu}_{\boldsymbol{\Xi}_i}(k)\big) \\ \text{subject to} \quad & \\ & \mathbf{z}_i(k)^\top\mathbf{z}_i(k) \leq 1.\end{aligned} \tag{4.65}$$

Using the notations $\mathbf{a}_j = -\mathbf{e}_j \boldsymbol{H}_{\overline{\mathcal{X}}} \boldsymbol{G}_{\boldsymbol{\Xi}}$, $f_z(\mathbf{z}_i(k)) = \mathbf{a}_j\big(\boldsymbol{\Sigma}^{-1/2}\mathbf{z}_i(k) + \boldsymbol{\mu}_{\boldsymbol{\Xi}_i}(k)\big)$ and $g_z(\mathbf{z}_i(k)) = \mathbf{z}_i(k)^\top \mathbf{z}_i(k) - 1$, the following expressions hold:

$$\nabla f_z(\mathbf{z}_i(k)) = \big(\boldsymbol{\Sigma}^{-1/2}\big)^\top \mathbf{a}_j^\top \tag{4.66a}$$

$$\nabla g_z(\mathbf{z}_i(k)) = 2\mathbf{z}_i(k) \tag{4.66b}$$

where ∇ is the gradient operator. The optimal value of $\mathbf{z}_i(k)$, denoted by $\mathbf{z}_i^*(k)$, can then be found by using the Karush-Kuhn-Tucker conditions (Boyd and Vandenberghe, 2009).

The Karush-Kuhn-Tucker conditions gather a stationarity condition:

$$\nabla f_z(\mathbf{z}_i^*(k)) + \zeta \nabla g_z(\mathbf{z}_i^*(k)) = \mathbf{0}_{(n+2)N_p \times 1}, \tag{4.67}$$

a primal feasibility condition:

$$g_z(\mathbf{z}_i^*(k)) \leq 0, \tag{4.68}$$

a dual feasibility condition:

$$\zeta \geq 0, \tag{4.69}$$

and a complementary slackness condition:

$$\zeta g_z(\mathbf{z}_i^*(k)) = 0. \tag{4.70}$$

Using (4.66), the stationarity condition (4.67) can be rewritten:

$$\mathbf{z}_i^*(k) = -\frac{1}{2\zeta}\boldsymbol{\Sigma}^{-1/2}\mathbf{a}_j^\top. \tag{4.71}$$

The value of $\mathbf{z}_i^*(k)$ can be injected into the complementary slackness condition (4.70), knowing that $g_z(\mathbf{z}_i^*(k)) = \mathbf{z}_i^*(k)^\top \mathbf{z}_i^*(k) - 1$:

$$\zeta^2 = \frac{1}{4}\mathbf{a}_j \boldsymbol{\Sigma}^{-1} \mathbf{a}_j^\top. \tag{4.72}$$

The multiplier ζ satisfies the dual feasibility condition (4.69) when taking the positive square root of (4.72). Finally, combining (4.71) and (4.72), the expression of $\mathbf{z}_i^*(k)$ is:

$$\mathbf{z}_i^*(k) = -\frac{\boldsymbol{\Sigma}^{-1/2}\mathbf{a}_j^\top}{\sqrt{\mathbf{a}_j \boldsymbol{\Sigma}^{-1} \mathbf{a}_j^\top}} \tag{4.73}$$

The vector $\mathbf{z}_i^*(k)$ satisfies the primal feasibility condition (4.68) since $g_z(\mathbf{z}_i^*(k)) = 0$. Injecting (4.73) into (4.65), the value of the bound is:

$$\begin{aligned}
\mathbf{e}_j \mathbf{b}_x(k) &= \mathbf{e}_j \boldsymbol{H}_{\overline{\mathcal{X}}} \boldsymbol{G}_{\boldsymbol{\Xi}} \big(\boldsymbol{\Sigma}^{-1/2}\mathbf{z}_i^*(k) + \boldsymbol{\mu}_{\boldsymbol{\Xi}_i}(k)\big) \\
&= \mathbf{e}_j \boldsymbol{H}_{\overline{\mathcal{X}}} \boldsymbol{G}_{\boldsymbol{\Xi}} \left(\boldsymbol{\Sigma}^{-1/2} \frac{\boldsymbol{\Sigma}^{-1/2} \boldsymbol{G}_{\boldsymbol{\Xi}}^\top \boldsymbol{H}_{\overline{\mathcal{X}}}^\top \mathbf{e}_j^\top}{\sqrt{\mathbf{e}_j \boldsymbol{H}_{\overline{\mathcal{X}}} \boldsymbol{G}_{\boldsymbol{\Xi}} \boldsymbol{\Sigma}^{-1} \boldsymbol{G}_{\boldsymbol{\Xi}}^\top \boldsymbol{H}_{\overline{\mathcal{X}}}^\top \mathbf{e}_j^\top}} + \boldsymbol{\mu}_{\boldsymbol{\Xi}_i}(k) \right)
\end{aligned}$$

such that, remembering that $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_{\boldsymbol{\Xi}_i}(k)^{-1}/\alpha$, the bound for the robust satisfaction of the constraint (4.45b) is:

$$\mathbf{e}_j \mathbf{b}_x(k) = \sqrt{\alpha \mathbf{e}_j \boldsymbol{H}_{\overline{\mathcal{X}}} \boldsymbol{G}_{\boldsymbol{\Xi}} \boldsymbol{\Sigma}_{\boldsymbol{\Xi}_i}(k) \boldsymbol{G}_{\boldsymbol{\Xi}}^\top \boldsymbol{H}_{\overline{\mathcal{X}}}^\top \mathbf{e}_j^\top} + \mathbf{e}_j \boldsymbol{H}_{\overline{\mathcal{X}}} \boldsymbol{G}_{\boldsymbol{\Xi}} \boldsymbol{\mu}_{\boldsymbol{\Xi}_i}(k) \tag{4.74}$$

for all $j \in \overline{1, r_x N_p}$. Equation (4.74) is used to find $\mathbf{b}_y(k)$, the bound for the robust satisfaction of the constraint (4.45d), by replacing $\boldsymbol{H}_{\overline{\mathcal{X}}}$ by $\overline{\boldsymbol{H}}_i(k)\overline{\boldsymbol{C}}$.

With the bounds $\mathbf{b}_x(k)$ and $\mathbf{b}_y(k)$, the chance-constrained MPC optimization problem (4.45) is reformulated into the classical MPC optimization problem:

$$\begin{aligned}
&\underset{\mathbf{U}_i(k)}{\text{minimize}} \quad J\Big(\widehat{\mathbf{X}}_i(k), \mathbf{U}_i(k), \widehat{\mathbf{X}}_{\mathbf{c}_i}(k), \mathbf{U}_{\mathbf{c}_i}(k)\Big) && \text{(4.75a)}\\
&\text{subject to}\\
&\quad \boldsymbol{H}_{\overline{\mathcal{X}}}\boldsymbol{G}\mathbf{U}_i(k) \leq \boldsymbol{\theta}_{\overline{\mathcal{X}}} - \boldsymbol{H}_{\overline{\mathcal{X}}}\boldsymbol{F}\widehat{\mathbf{x}}_i(k) - \mathbf{b}_x(k), && \text{(4.75b)}\\
&\quad \mathbf{U}_i(k) \in \overline{\mathcal{U}}, && \text{(4.75c)}\\
&\quad \overline{\boldsymbol{H}}_i(k)\overline{\boldsymbol{C}}\boldsymbol{G}\mathbf{U}_i(k) \leq \overline{\boldsymbol{\theta}}_i(k) - \overline{\boldsymbol{H}}_i\overline{\boldsymbol{C}}\boldsymbol{F}\widehat{\mathbf{x}}_i(k) - \mathbf{b}_y(k) && \text{(4.75d)}
\end{aligned}$$

where $J\Big(\widehat{\mathbf{X}}_i(k), \mathbf{U}_i(k), \widehat{\mathbf{X}}_{\mathbf{c}_i}(k), \mathbf{U}_{\mathbf{c}_i}(k)\Big)$ is given by (4.48), the matrices in (4.75b) and (4.75d) have been extensively described in Paragraph 4.2.3.1 and the bounds $\mathbf{b}_x(k)$ and $\mathbf{b}_y(k)$ are found with (4.74).

However, despite the will for the chance constraints (4.45b) and (4.45d) to be satisfied for all perturbations, expressed by choosing $P_x = P_y = 1$, the ellipsoid $\mathscr{E}(\boldsymbol{\Sigma}_{\boldsymbol{\Xi}_i}(k)^{-1}, \boldsymbol{\mu}_{\boldsymbol{\Xi}_i}(k), \alpha)$ contains almost all the perturbations and there is probability $1 - P_\chi$ that the constraints are not satisfied. Indeed, the fraction P_χ can be as close to 1 as possible but not exactly equal to 1 due to the properties of the chi-squared distribution. The control strategy presented in this paragraph can then allow the constraints to not be satisfied with a probability $1 - P_\chi$. This concludes on the algebraic relaxation of the proposed probabilistic constraints.

4.2.4 Proposed deployment results for MAS with stochastic perturbations

In this paragraph the chance-constrained MPC strategy proposed above is applied to two multi-vehicle systems. These two MVSs are composed of vehicles obeying single integrator dynamics for the first system and quadrotor UAVs dynamics as described in Section 3.3 for the second system. Then, Paragraph 4.2.4.1 presents the deployment of a MAS composed of vehicles obeying single integrator dynamics subject to unbounded stochastic perturbations, while Paragraph 4.2.4.1 shows the deployment of a MAS composed of quadrotor UAVs subject to the same kind of perturbations.

4.2.4.1 Single integrator dynamics

Let Σ be a multi-agent system composed of $N = 6$ vehicles obeying the continuous-time dynamics:

$$\begin{aligned}
\dot{\mathbf{x}}_i(t) &= \mathbf{u}_i(k) + \mathbf{d}_i(t)\\
\mathbf{y}_i(t) &= \mathbf{x}_i(t) + \mathbf{w}_i(t)
\end{aligned} \tag{4.76}$$

with $\mathbf{x}_i, \mathbf{u}_i, \mathbf{y}_i, \mathbf{d}_i, \mathbf{w}_i \in \mathbb{R}^2$, which is discretized as:

$$\begin{aligned}
\mathbf{x}_i(k+1) &= \mathbf{x}_i(k) + T_s\mathbf{u}_i(k) + \boldsymbol{\delta}_i(k)\\
\mathbf{y}_i(k) &= \mathbf{x}_i(k) + \boldsymbol{\gamma}_i(k)
\end{aligned} \tag{4.77}$$

where $T_s = 0.2\,\mathrm{s}$ is the sampling period. The variance matrices of $\mathbf{d}_i(t)$ and $\mathbf{w}_i(t)$ are constant over time and are, respectively, $\boldsymbol{\Sigma}_{\mathbf{d}_i}(t) = 0.01 \cdot \boldsymbol{I}_2$ and $\boldsymbol{\Sigma}_{\mathbf{w}_i}(t) = 10^{-4} \cdot \boldsymbol{I}_2$ for all $i \in \overline{1, N}$. Then, from (4.36) and (4.37), $\boldsymbol{\Sigma}_{\boldsymbol{\gamma}_i}(k) = 5 \cdot 10^{-3} \cdot \boldsymbol{I}_2$ and $\boldsymbol{\Sigma}_{\boldsymbol{\delta}_i}(k) = 2 \cdot 10^{-3} \cdot \boldsymbol{I}_2$ for all $i \in \overline{1, N}$ and for all $k \geq 0$. For all agents, the variance matrix of the estimation error is initialized as $\boldsymbol{\Sigma}_{\widetilde{\mathbf{x}}_i}(0) = 0.1 \cdot \boldsymbol{I}_2$, while its initial mean is $\boldsymbol{\mu}_{\widetilde{\mathbf{x}}_i}(0) = \mathbf{0}_{2\times 1}$. The mean of the measurement noise $\mathbf{w}_i$ is always zero for all $i \in \overline{1, N}$, i.e. $\boldsymbol{\mu}_{\mathbf{w}_i}(t) = \mathbf{0}_{2\times 1}$. A discussion on the value of the mean of the process noise $\mathbf{d}_i$ is held later in this paragraph. For now, it is considered to be also zero, i.e. $\boldsymbol{\mu}_{\mathbf{d}_i}(t) = \mathbf{0}_{2\times 1}$ for all $i \in \overline{1, N}$. The values of the mean vectors $\boldsymbol{\mu}_{\boldsymbol{\delta}_i}(k)$ and $\boldsymbol{\mu}_{\boldsymbol{\gamma}_i}(k)$ are then immediately $\mathbf{0}_{2\times 1}$ from (4.34) and (4.35).

The MVS is deployed inside the output space:

$$\mathcal{Y} = \mathcal{X} = \left\{ \mathbf{x} \in \mathbb{R}^2 \;\middle|\; \begin{bmatrix} -1 & 4 \\ -3 & -2 \\ 3 & -7 \\ 5 & 1 \end{bmatrix} \mathbf{x} \leq \begin{bmatrix} 24 \\ 20 \\ 44 \\ 48 \end{bmatrix} \right\}$$

with an input space $\mathcal{U} = \mathbb{B}^2(2 \cdot \mathbf{1}_{2\times 1})$. The goal for the MVS is then to deploy in $\mathcal{Y}$ into a static Chebyshev configuration, where the estimated state of each agent lies on the Chebyshev center of its Voronoi cell.

As for the tube-based MPC case of Paragraph 4.1.4.1, the weighting matrices are chosen such that $\boldsymbol{Q} = \boldsymbol{R} = \boldsymbol{I}_2$ and $\boldsymbol{P}$ is the solution of the algebraic Riccati equation:

$$\boldsymbol{A}^\top \boldsymbol{P} \boldsymbol{A} - \boldsymbol{P} - \boldsymbol{A}^\top \boldsymbol{P} \boldsymbol{B} \left(\boldsymbol{B}^\top \boldsymbol{P} \boldsymbol{B} + \boldsymbol{R} \right)^{-1} \boldsymbol{B}^\top \boldsymbol{P} \boldsymbol{A} + \boldsymbol{Q} = \mathbf{0}_2$$

with $\boldsymbol{A} = \boldsymbol{I}_2$ and $\boldsymbol{B} = T_s \boldsymbol{I}_2$. The prediction horizon is chosen to be $N_p = 10$. The scaling factor used in the terminal constraint is chosen to be $\lambda_i = 0.9$, with $i \in \overline{1, N}$. The parameter P_χ is chosen to be $P_\chi = 0.99$ such that, according to chi-squared tables (Elderton, 1902), the parameter α appearing in (4.58) is $\alpha = 63.691$ since the random variable $\xi_i(k)$ of (4.57) follows a chi-squared law with 40 degrees of freedom. Finally, the observer gain matrix $\boldsymbol{L}$ is chosen such that it is the solution of a LQR design problem with the weighting matrices $\boldsymbol{Q}_{\boldsymbol{L}} = 10 \cdot \boldsymbol{I}_2$ and $\boldsymbol{R}_{\boldsymbol{L}} = \boldsymbol{I}_2$.

The estimated state $\widehat{\mathbf{x}}_i$, with $i \in \overline{1, N}$, is initialized to a random position in $\mathcal{Y}$ such that the optimization problem (4.45) is feasible. The real state $\mathbf{x}_i$, with $i \in \overline{1, N}$, is initialized with $\mathbf{x}_i(0) = \widehat{\mathbf{x}}_i(0)$.

The initial and final configuration of the multi-agent system Σ in $\mathcal{Y}$ are displayed in Figures 4.16 and 4.17. The estimated states $\widehat{\mathbf{x}}_i(0)$ and $\widehat{\mathbf{x}}_i(250)$, with $i \in \overline{1, N}$, are represented by circles while the Chebyshev centers $\mathbf{c}_i(0)$ and $\mathbf{c}_i(250)$ are represented by stars. It is obvious from Figure 4.17 that the estimated position $\widehat{\mathbf{x}}_i$, with $i \in \overline{1, N}$, has reached the Chebyshev center $\mathbf{c}_i$ of its Voronoi cell $\mathcal{V}_i$.

As for the tube-based case, a measure of the convergence of Σ to a static configuration is given by the distance between the estimated states and the Chebyshev centers:

$$d_i(k) = \left\| \widehat{\mathbf{x}}_i(k) - \mathbf{c}_i(k) \right\|_2 \tag{4.78}$$

with $i \in \overline{1, N}$. As shown by Figure 4.18, all these distances converge to 0. Moreover, Figure 4.19 shows that the norm of the estimation error $\widetilde{\mathbf{x}}_i$, with $i \in \overline{1, N}$, denoted by $\left\| \widetilde{\mathbf{x}}_i(k) \right\|_2$, remains bounded with $\left\| \widetilde{\mathbf{x}}_i(k) \right\|_2 \leq 0.008\,\mathrm{m}$.

To show that the estimation error $\widetilde{\mathbf{x}}_i(k)$ and the distance $d_i(k)$ remain bounded when the mean of the process noise $\boldsymbol{\mu}_{\mathbf{d}_i}$ is not null, simulations are run to compare

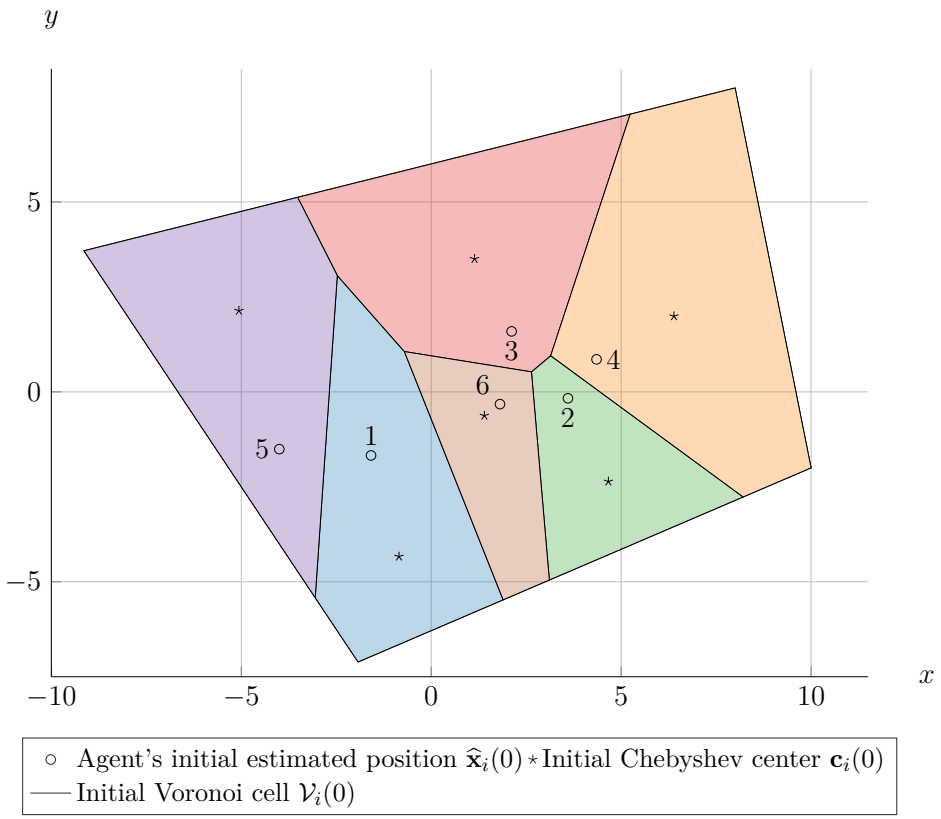


Figure 4.16: Initial position of the MVS Σ in the output space.

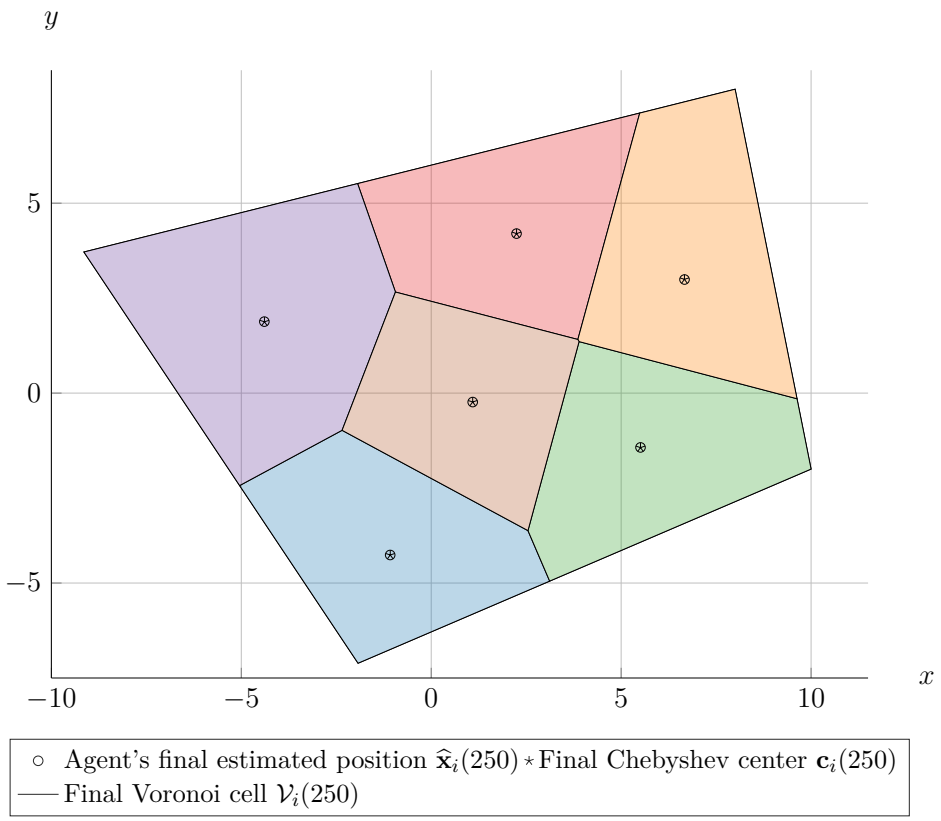


Figure 4.17: Final position of the MVS Σ in the output space.

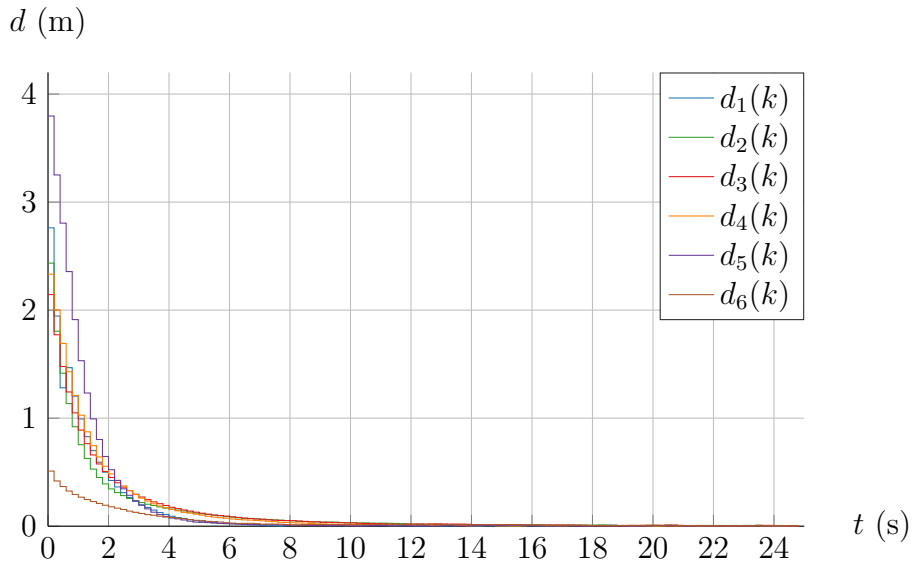


Figure 4.18: Distance of the estimated position of each agent of Σ to its Chebyshev center over time.

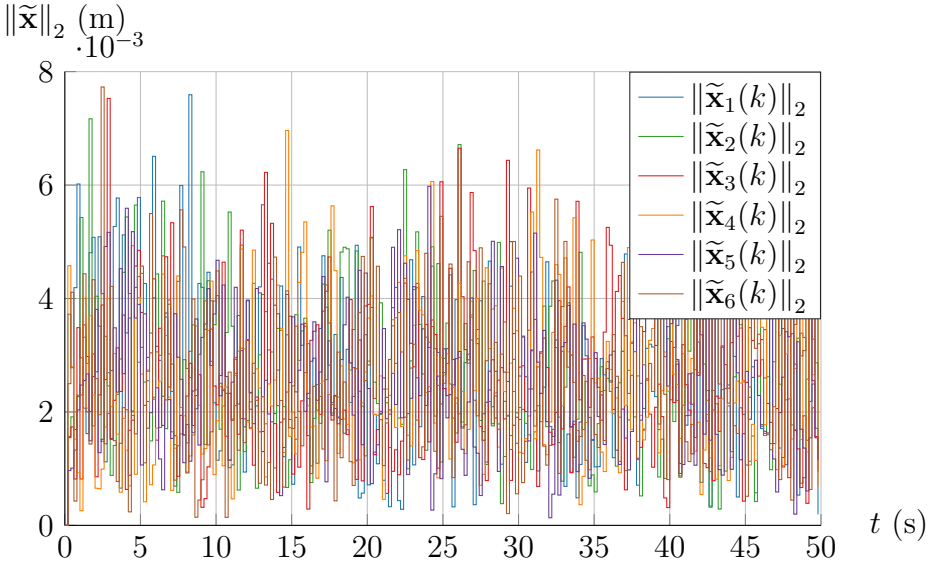


Figure 4.19: Norm of the estimation error for each agent of Σ.

the efficiency of the chance-constrained MPC (4.75) with that of the nominal MPC (3.12) similarly to Gavilan et al. (2012). Three batches of 2000 simulations are run for a total of 6000 simulations. Half of these simulations, i.e. 3000 simulations, are run with the CCMPC of Section 4.2.3, while the other half are run with the nominal MPC of Section 3.2.2. In all the simulations, the mean of the process noise $\mathbf{d}_i(t)$ is modified such that:

$$\boldsymbol{\mu}_{\mathbf{d}_i}(t) = \begin{cases} \mathbf{0}_{2\times 1} & \text{if} \quad t < 2\,\text{s} \\ \boldsymbol{\mu}_{\mathbf{d}_i} & \text{else} \end{cases}$$

where $\boldsymbol{\mu}_{\mathbf{d}_i} \in \mathbb{R}^2$ is a vector which components are chosen randomly in one of three given intervals $\mathcal{I}$. Thus, for each interval $\mathcal{I}$, there are 1000 simulations of Σ where each agent is subject to a given perturbation $\mathbf{d}_i(k)$ and driven with the chance-

constrained MPC (4.75). Then, 3000 more simulations (1000 simulations for a given interval $\mathcal{I}$) are run with the exact same scenario, where each agent is driven with the decentralized nominal MPC (3.12). In this last case, the agents are simulated using the dynamics (4.77) and the control input is obtained by replacing the state $\mathbf{x}_i$ by the estimated state $\widehat{\mathbf{x}}_i$ in the optimization problem (3.12).

For each of the three batches, the interval $\mathcal{I}$ is such that $\mathcal{I} = [-1, 1]$, $\mathcal{I} = [-1.5, 1.5]$ and $\mathcal{I} = [-2, 2]$, respectively. When running the simulation, whenever an agent does not satisfy a constraint, the whole simulation is stopped and its results are not counted in the following. The results on the deployment success rate are gathered in Table 4.1. For each batch of simulations, 100 % of the simulations with the chance-constrained MPC result in the deployment of the MAS in a static configuration. With the nominal MPC for the first batch, i.e. for $\mathcal{I} = [-1, 1]$, 24.2 % of the simulations result in at least one agent not satisfying a constraint. This percentage goes up to 84.8 % for the second batch, i.e. for $\mathcal{I} = [-1.5, 1.5]$, and 98.5 % for the third one, i.e. for $\mathcal{I} = [-2, 2]$. Since the number of successful simulations for the second and third batch is not representative, a comparative analysis is performed between the performances of the CCMPC and of the nominal MPC for the first batch only.

The mean value of $d_i(k)$, with $i \in \overline{1, N}$, as defined in (4.78) for $k \in \overline{30, 150}$, where $k = 150$ is the end of the simulation, is computed. The value of $k = 30$ for the beginning of the interval is chosen such that the MAS has converged into a static Chebyshev configuration based on the results presented in Figure 4.18. The use of the mean value of $d_i(k)$ allows to remove the potential sudden variations that can appear on $d_i(k)$ due to a sudden change in the position of the Chebyshev center $\mathbf{c}_i(k)$ as was observed in the nominal case in Section 3.2.4. Let m_{d_i} denote the aforementioned mean, i.e.:

$$m_{d_i} = \frac{1}{121} \sum_{k=30}^{150} d_i(k).$$

The mean m_{d_i} is studied for each agent and the results are gathered in Table 4.2. For each simulation, six agents are deployed in the workspace $\mathcal{Y}$. This means that 6000 values for m_{d_i} are retrieved in the chance-constrained case and 4548 values are retrieved in the nominal case since only 758 simulations ran successfully, the other 242 ending up in at least one agent not satisfying the constraints. With the use of the CCMPC, the maximum value of m_{d_i} is $\max m_{d_i} = 25.38\,\text{cm}$ and 100 % of the 6000 simulated agents are such that $m_{d_i} \leq \max m_{d_i}$. With the use of the nominal MPC, the maximum value of m_{d_i} is $\max m_{d_i} = 154.41\,\text{cm}$. To compare with the CCMPC case, only 3.03 % of the 6000 simulated agents are able to maintain $m_{d_i} \leq 25.38\,\text{cm}$ while 72.77 % of these 6000 agents are such that $25.38\,\text{cm} < m_{d_i} \leq 154.41\,\text{cm}$. For the other 24.2 % of the simulated agents no value is obtained for m_{d_i} since the simulations were interrupted.

Since for all simulations, $\boldsymbol{\mu}_{\boldsymbol{\gamma}_i}(t) = \mathbf{0}_{2\times 1}$, there is no significant difference in the values of the norm of the estimation error $\|\mathbf{x}_i(k) - \widehat{\mathbf{x}}_i(k)\|_2$. However, the Monte-Carlo simulations could be repeated to measure the impact of a non null mean for the measurement noise, caused potentially by a bias on the sensors. These results highlight the necessity of the robust chance-constrained model predictive controller proposed in this section for the deployment of a MAS subject to unbounded stochastic perturbations.

Table 4.1: Comparison of the deployment success rate with the CCMPC scheme and the nominal MPC scheme.

	$\mathcal{I} = [-1, 1]$		$\mathcal{I} = [-1.5, 1.5]$		$\mathcal{I} = [-2, 2]$	
	Nominal MPC	CCMPC	Nominal MPC	CCMPC	Nominal MPC	CCMPC
Number of simulations	1000	1000	1000	1000	1000	1000
Simulation success rate	75.8 %	100 %	15.2 %	100 %	1.5 %	100 %

Table 4.2: Comparison of the mean value of the distance m_{d_i} for the agents in the case $\mathcal{I} = [-1, 1]$.

		$m_{d_i} \leq 25.38\,\text{cm}$		$25.38\,\text{cm} < m_{d_i} \leq 154.41\,\text{cm}$	
	Number of simulated agents	Number of agents	Percentage of agents	Number of agents	Percentage of agents
Nominal MPC	6000	182	3.03 %	4366	72.77 %
CCMPC	6000	6000	100 %	0	0 %

4.2.4.2 UAV dynamics

Let Σ be a multi-agent system composed of $N = 6$ vehicles. Each agent $i \in \overline{1, N}$ is a quadrotor UAV obeying the dynamics and the control structure presented in Section 3.3. The predictive controller used for the outer loop in Paragraph 3.3.2.3 is replaced by the chance-constrained MPC described in Paragraph 4.2.3.2.

The linearized continuous-time dynamics of the outer loop are:

$$\begin{aligned} \dot{\mathbf{x}}_i(t) &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{x}_i(t) + \begin{bmatrix} 0 & 0 \\ 0 & g \\ 0 & 0 \\ -g & 0 \end{bmatrix} \mathbf{u}_i(t) + \underbrace{\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}}_{\boldsymbol{M}} \mathbf{d}_i(t) \\ \mathbf{y}_i(t) &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{x}_i(t) + \mathbf{w}_i(t) \end{aligned} \tag{4.79}$$

where $\mathbf{x}_i \in \mathbb{R}^4$, $\mathbf{u}_i, \mathbf{y}_i, \mathbf{d}_i, \mathbf{w}_i \in \mathbb{R}^2$. Assumption 4.2 is satisfied since $\boldsymbol{M}$ has full rank. In the dynamics of (4.79), the perturbation $\mathbf{d}_i(t)$ only acts on the horizontal speed of the UAVs. Indeed, the perturbations considered are potential wind gusts. Moreover, the main source of modeling error when linearizing the position subsystem is on the horizontal speed dynamics.

Then, the dynamics (4.79) is discretized as:

$$\begin{aligned} \mathbf{x}_i(k+1) &= \boldsymbol{A}\mathbf{x}_i(k) + \boldsymbol{B}\mathbf{u}_i(k) + \boldsymbol{\delta}_i(k) \\ \mathbf{y}_i(k) &= \boldsymbol{C}\mathbf{x}_i(k) + \boldsymbol{\gamma}_i(k) \end{aligned} \tag{4.80}$$

where $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ are the matrices defined in (3.39), with the sampling period $T_s = 0.2\,\text{s}$. The variance matrices of $\mathbf{d}_i$ and $\mathbf{w}_i$ are $\boldsymbol{\Sigma}_{\mathbf{d}_i}(t) = 10^{-3} \cdot \boldsymbol{I}_2$ and $\boldsymbol{\Sigma}_{\mathbf{w}_i}(t) =$

$10^{-5} \cdot \boldsymbol{I}_2$ for all $i \in \overline{1, N}$. With (4.37) and (4.36), $\boldsymbol{\Sigma}_{\boldsymbol{\gamma}_i}(k) = 5 \cdot 10^{-5} \cdot \boldsymbol{I}_2$ and:

$$\boldsymbol{\Sigma}_{\boldsymbol{\delta}_i}(k) = 10^{-4} \cdot \begin{bmatrix} 0.027 & 0.2 & 0 & 0 \\ 0.2 & 2 & 0 & 0 \\ 0 & 0 & 0.027 & 0.2 \\ 0 & 0 & 0.2 & 2 \end{bmatrix}.$$

The variance matrix of the estimation error is initialized as $\boldsymbol{\Sigma}_{\tilde{\mathbf{x}}}(0) = 10^{-3} \cdot \boldsymbol{I}_4$ while its mean is initialized as $\boldsymbol{\mu}_{\tilde{\mathbf{x}}_i}(0) = \mathbf{0}_{4\times 1}$. The mean of the measurement and process noises $\mathbf{w}_i$ and $\mathbf{d}_i$ are always zero for all $i \in \overline{1, N}$, i.e. $\boldsymbol{\mu}_{\mathbf{w}_i}(t) = \boldsymbol{\mu}_{\mathbf{d}_i}(t) = \mathbf{0}_{2\times 1}$. The values of the mean vectors $\boldsymbol{\mu}_{\boldsymbol{\delta}_i}(k)$ and $\boldsymbol{\mu}_{\boldsymbol{\gamma}_i}(k)$ are then immediately $\mathbf{0}_{4\times 1}$ and $\mathbf{0}_{2\times 1}$, respectively, from (4.35) and (4.34).

The MVS is deployed inside the output space:

$$\mathcal{Y} = \left\{ \mathbf{x} \in \mathbb{R}^2 \;\middle|\; \begin{bmatrix} -1 & 4 \\ -3 & -2 \\ 3 & -7 \\ 5 & 1 \end{bmatrix} \mathbf{x} \leq \begin{bmatrix} 24 \\ 20 \\ 44 \\ 48 \end{bmatrix} \right\}$$

and the state space is such that $\boldsymbol{C}\mathcal{X} = \mathcal{Y}$ and the horizontal speeds $v_{x,i}(k)$ and $v_{y_i}(k)$, with $i \in \overline{1, N}$, belong to $\mathbb{B}^2(2 \cdot \mathbf{1}_{2\times 1})$ with an input space $\mathcal{U} = \mathbb{B}^2\left(\frac{\pi}{6} \cdot \boldsymbol{I}_2\right)$.

As discussed in Paragraph 4.1.4.2, the outer loop controller has to be modified with respect to Paragraph 3.3.2.3. Indeed, according to Section 3.3.2, a cascaded control structure is considered to control each UAV. For the inner loop, a continuous-time controller based on feedback linearization is used for attitude control. For the outer loop, the discrete-time CCMPC of Section 4.2.3 is used for position control. However, for simulation purpose, the controller of the inner loop is run in discrete-time with a sampling period of 1 ms. This sampling period is 200 times shorter than the sampling period of the outer loop. Indeed, the control input applied to the agent $i \in \overline{1, N}$ is changed to $\mathbf{u}_i(k_{\text{in}}) = \mathbf{u}_i(k)$ for all $k_{\text{in}} \in \overline{\frac{T_s}{T_s^{\text{in}}}k, \frac{T_s}{T_s^{\text{in}}}(k+1) - 1}$, where $T_s^{\text{in}} = 1$ ms is the sampling period of the inner loop. The gain matrix $\boldsymbol{L}_{\text{in}}$ of the observer shown in Figure 4.20 is obtained by solving a LQR design problem with the weighting matrices $\boldsymbol{Q}_{\boldsymbol{L}_{\text{in}}} = 10 \cdot \boldsymbol{I}_4$ and $\boldsymbol{R}_{\boldsymbol{L}_{\text{in}}} = \boldsymbol{I}_2$ for the dynamics (4.80), where the sampling period is replaced by $T_s^{\text{in}} = 1$ ms. The resulting control structure is presented in Figure 4.20.

As in Section 3.3.3, the contraction factor for the terminal constraints is $\lambda_i = 0.9$, with $i \in \overline{1, N}$. The weighting matrices are chosen such that $\boldsymbol{Q} = \boldsymbol{I}_4$, $\boldsymbol{R} = \boldsymbol{I}_2$ and $\boldsymbol{P}$ is the solution of the algebraic Riccati equation:

$$\boldsymbol{A}^\top \boldsymbol{P} \boldsymbol{A} - \boldsymbol{P} - \boldsymbol{A}^\top \boldsymbol{P} \boldsymbol{B} \left(\boldsymbol{B}^\top \boldsymbol{P} \boldsymbol{B} + \boldsymbol{R}\right)^{-1} \boldsymbol{B}^\top \boldsymbol{P} \boldsymbol{A} + \boldsymbol{Q} = \mathbf{0}_4$$

with $\boldsymbol{A}$ and $\boldsymbol{B}$ defined in (3.39). The observer gain matrix $\boldsymbol{L}$ of the outer loop used in the CCMPC block is obtained as the solution of a LQR design problem with the weighting matrices $\boldsymbol{Q}_{\boldsymbol{L}} = 10 \cdot \boldsymbol{I}_4$ and $\boldsymbol{R}_{\boldsymbol{L}} = \boldsymbol{I}_2$ for the dynamics (4.80). The prediction horizon is chosen such that $N_p = 10$. The solver for optimization problem (4.45) is generated with CVXGEN (Mattingley and Boyd, 2012, 2013). The evolution model for the quadrotor UAV agents in the simulation is then the nonlinear model of (3.28).

The position $\mathbf{y}_i$ of each UAV agent is initialized to a random value of $\mathcal{Y}$ such that the optimization problem (4.59) is feasible. The altitude z_i of each agent is

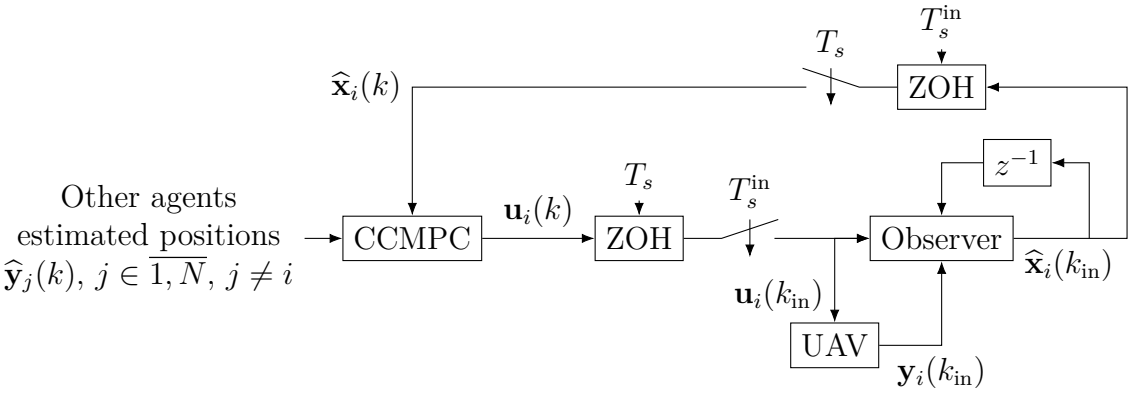


Figure 4.20: Structure of the position controller for a quadrotor UAV subject to perturbations.

initialized to $z_i(0) = \bar{z} = 5\,\text{m}$. Moreover, the speeds, angles and angular speeds are null at $k = 0$. The estimated state $\widehat{\mathbf{x}}_i$ of each agent is such that $\widehat{\mathbf{x}}_i(0) = \mathbf{x}_i(0)$.

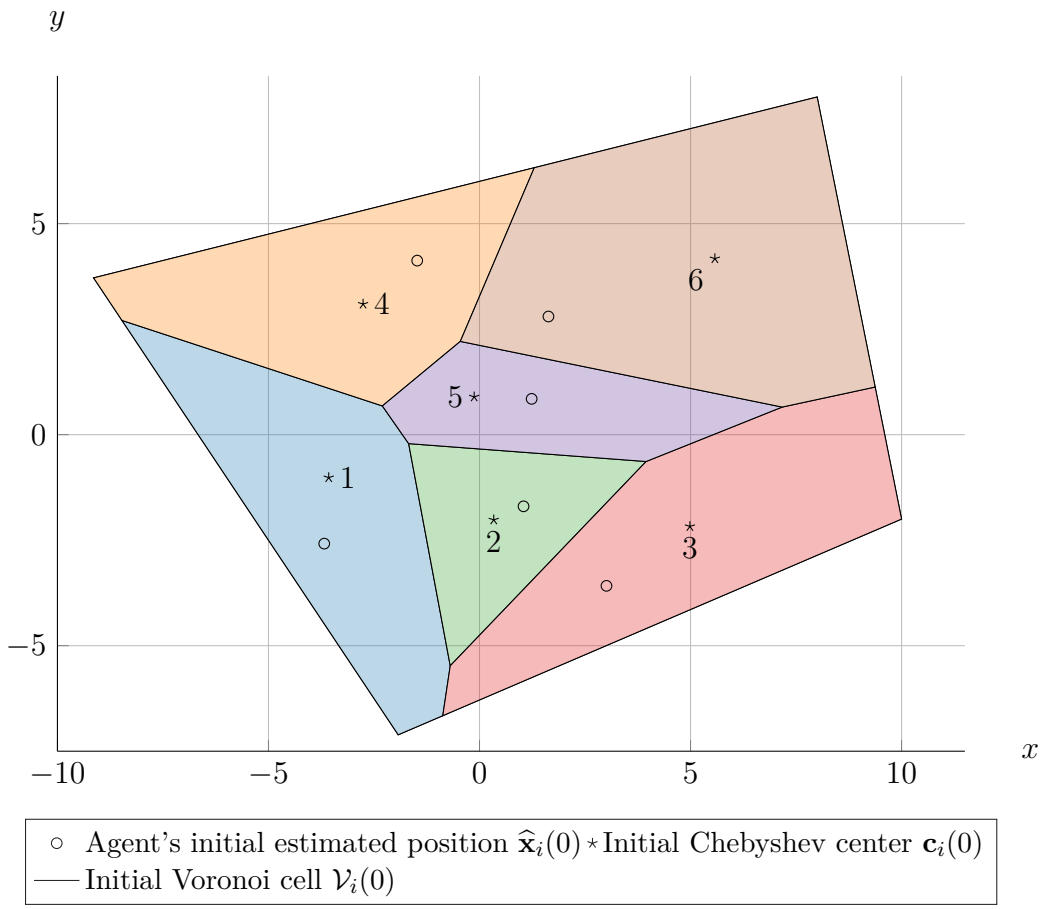


Figure 4.21: Initial position of the MVS Σ in the output space.

The initial and final configuration of the MAS Σ in $\mathcal{Y}$ are displayed in Figure 4.21 and Figure 4.22 respectively. The estimated positions $\widehat{\mathbf{y}}_i(0)$ and $\widehat{\mathbf{y}}_i(250)$, with $i \in \overline{1,N}$, are represented by circles, while the Chebyshev centers $\mathbf{c}_i(0)$ and $\mathbf{c}_i(250)$ are represented by stars. As for the single integrator case of Paragraph 4.2.4.1, it is

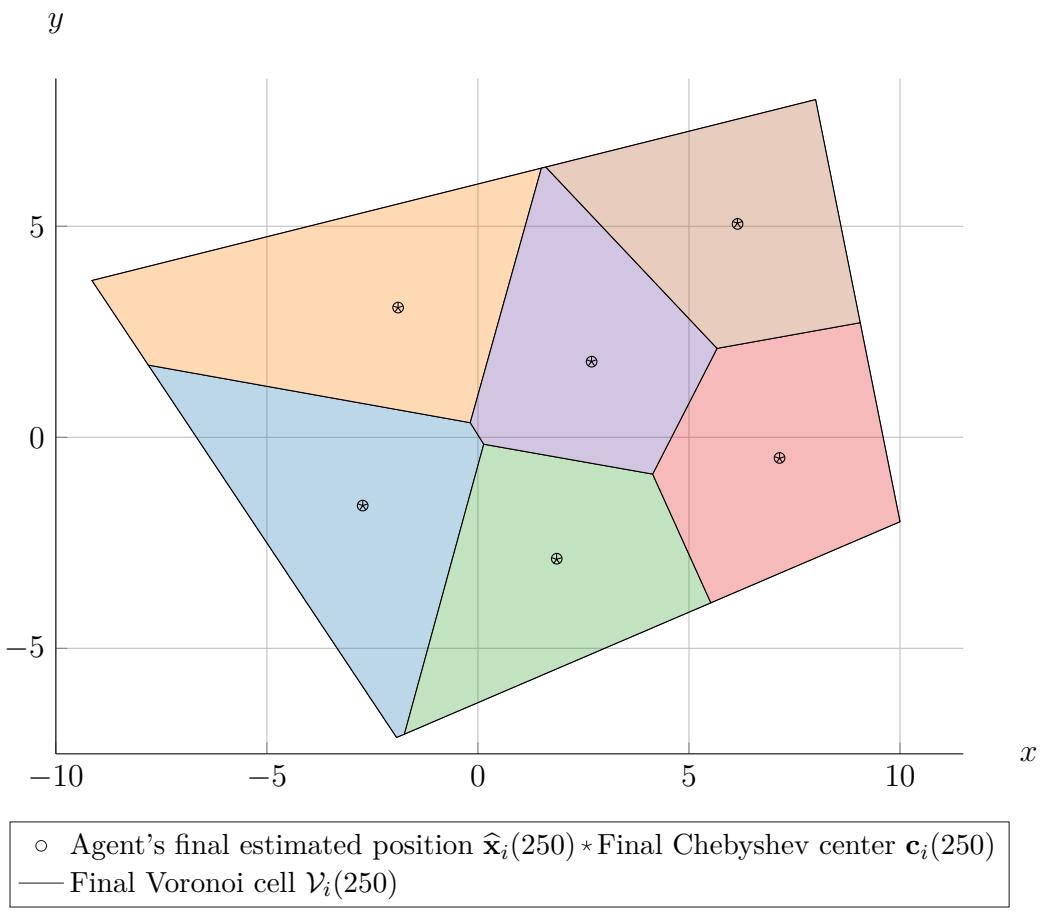


Figure 4.22: Final position of the MVS Σ in the output space.

obvious from Figure 4.22 that the estimated position $\widehat{\mathbf{y}}_i$, with $i \in \overline{1, N}$, has reached the Chebyshev center $\mathbf{c}_i$ of its Voronoi cell $\mathcal{V}_i$. The real positions of the agents are not shown since, given the noise variances, they are close to the estimated position. This assertion is supported by the norm of the estimation errors shown in Figure 4.24.

As for the tube-based case, a measure of the convergence of Σ to a static configuration is given by the distance between the estimated positions and the Chebyshev centers:

$$d_i(k) = \|\widehat{\mathbf{y}}_i(k) - \mathbf{c}_i(k)\|_2$$

with $i \in \overline{1, N}$. As shown by Figure 4.23, all these distances converge to 0. However, the convergence is slower than in the bounded perturbations case of Paragraph 4.1.4.2 due to the high level of noise. Indeed, a static configuration is reached in around 15 s for the unbounded stochastic perturbations case, while it is reached in around 8 s for the bounded deterministic perturbations case. Moreover, Figure 4.24 shows, as in the single integrator case, that the norm of the estimation error remains bounded with $\|\mathbf{x}_i(k) - \widehat{\mathbf{x}}_i(k)\|_2 \leq 42\,\text{cm}$. Due to the modified control strategy and the more complex dynamics, the norm of the estimation error is larger than the one obtained in the single integrator case of Paragraph 4.2.4.1, where $\|\mathbf{x}_i(k) - \widehat{\mathbf{x}}_i(k)\|_2 \leq 0.8\,\text{cm}$. A similar approach than the one developed in Paragraph 4.2.4.1 could be developed here to show that the error remains bounded when the mean of the process noise, i.e. the external disturbance, is not null.

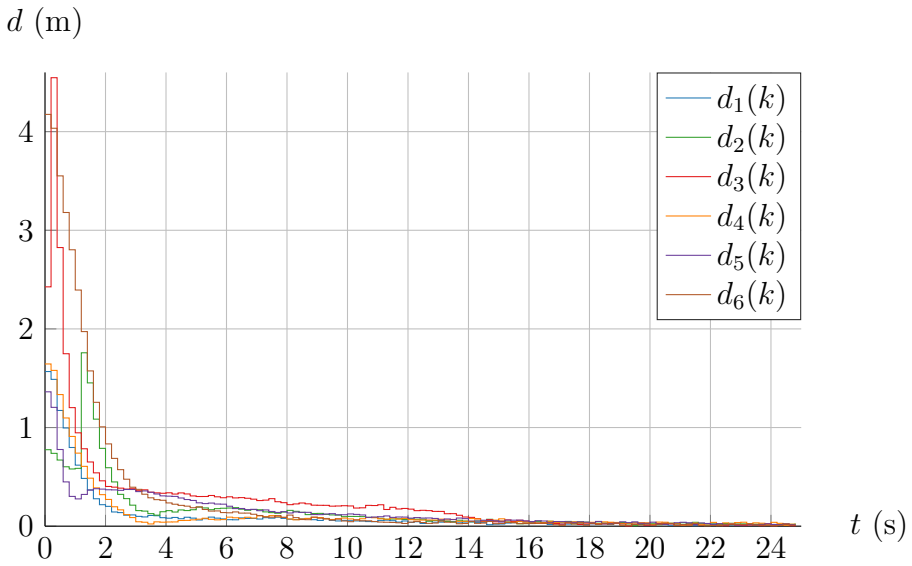


Figure 4.23: Distance of the estimated position of each agent of Σ to its Chebyshev center over time.

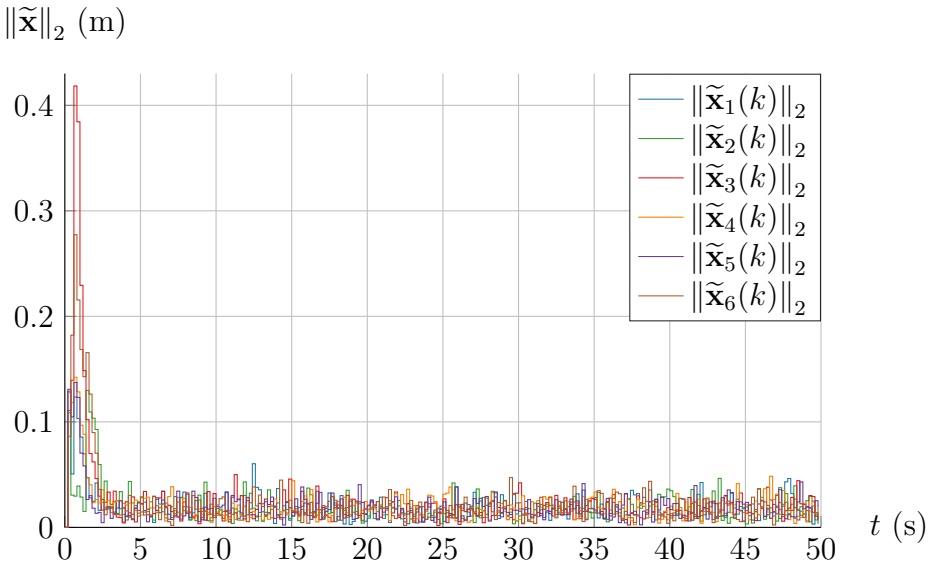


Figure 4.24: Norm of the estimation error for each agent of Σ.

4.3 Conclusion

This chapter presents two control strategies to perform the robust Voronoi-based deployment of a multi-agent system (MAS) where each agent is tracking the Chebyshev center of the Voronoi cell or guaranteed Voronoi cell it belongs to. The vehicles composing the MAS are subject to perturbations either *bounded deterministic* or *unbounded stochastic*. In both cases, a solution based on model predictive control (MPC) has been proposed to adapt the deployment algorithm presented in Chapter 3 to the perturbed case.

As a first step, the input and output perturbations are considered deterministic and bounded. Thus, a decentralized model predictive controller is derived from existing robust output feedback tube-based MPC (Mayne et al., 2006) and adapted for

the deployment control of a MAS. Such a control strategy is based on the knowledge of invariant sets for error dynamics. In order for the optimization problem to remain feasible, these invariant sets have to be of minimal size. Since their size depends on the value of the observer and state feedback controller gain matrices appearing in the output feedback tube-based MPC strategy, tuning procedures derived from the one given by Alvarado (2007) are introduced. These procedures are used to obtain the observer and state feedback gain matrices guaranteeing that the robust positively invariant sets for the associated stable dynamics are of minimal size. Simulations are run on both single integrator and nonlinear UAV dynamics to show that the MAS deploys in a static Chebyshev configuration with guarantees that the state of the system remains bounded inside a set such that, despite the perturbations, collisions between the agents are avoided.

However, despite being efficient, such a strategy is limited in the case of real systems, which opens the way to interesting future research directions. Indeed, in the double integrator case, i.e. for the position control of a quadrotor unmanned aerial vehicles fleet, the strategy depends on the availability of measurements at an important rate which is not realistic. This is due to the properties of the Luenberger observer which might not be adapted to such an application. Then, a more robust observer should be used such as Kalman filter (Chui and Chen, 2017, Chevet et al., 2017) or even set-membership state estimation techniques (Le et al., 2013) on which tube-based MPC techniques have already been studied (Le, 2012) in the case of a single system. This last possibility could be coupled with the tuning procedure proposed in the present thesis since both are based on bilinear matrix inequalities. Indeed, one of the main point arising in the deployment problem is the appearance of time-varying constraints, discussed in Chapter 3, in the optimization problem. An online computation of the invariant sets based on set-membership state estimation would then allow to take such time-varying constraints into account, which is not the case for the method proposed in the present thesis. However, an analysis of the computation time is necessary.

Then, in a second stage, stochastic and unbounded perturbations are considered. Such perturbations are then characterized by a mean and a variance matrix. The mean represents an external deterministic disturbance for the input perturbation or a sensor bias for the output perturbation while the variance matrix is related to the process or measurement noise to which the system is subject. To perform the robust deployment of the MAS, a novel output feedback decentralized model predictive controller is derived from the state feedback chance-constrained MPC strategy introduced in Gavilan et al. (2012). Indeed, Gavilan et al. (2012) proposes a state-feedback chance-constrained model predictive controller applied to a single system. In the present thesis, the control strategy is improved to obtain a decentralized output-feedback chance-constrained model predictive controller applied to a MAS. This strategy consists in using the stochastic properties of the perturbations to compute a probabilistic bound to the perturbation signal, relaxing the chance constraints of the optimization problem into algebraic constraints. Finally, simulations on a MAS composed of agents obeying single integrator dynamics or nonlinear quadrotor UAVs dynamics show that the MAS deploys in a static Chebyshev configuration. Monte-Carlo simulations in the single integrator dynamics case show that the behavior of such a control strategy is analogous to the robust tube-based MPC case in the sense that, with the proposed chance-constrained MPC, the errors remain bounded.

As for the tube-based MPC case, such a strategy can be limited in the case of real

systems. The same problem linked to the observer arises. Moreover, this strategy, with the formulation given in the present thesis, works for stochastic processes characterized by their mean and variance matrix, i.e. the Gaussian processes. A logical next step of the work presented in this thesis would be to adapt it to different probability distributions. Another possibility would be to study tube-based chance-constrained techniques as developed in Cannon et al. (2012) adapted to the case of the deployment of a MAS. Indeed, the work of Cannon et al. (2012) and the subsequent improvements of their algorithm as presented in Dai et al. (2015) or Dai et al. (2018), for example, are based on the assumption that the constraints are constant over time while in the deployment case, the constraints are time-varying. A tube-based strategy could then be coupled with a set-membership state estimation technique or a tuning procedure similar to the one used for the tube-based MPC case.

Chapter 5

Extension to the deployment of a time-varying multi-vehicle system

Table of Contents

Chapter 3 introduces the deployment problem of a multi-agent system over a convex bounded area in the nominal case. For its part, Chapter 4 discusses the same problem when the agents are subject to input and output perturbations. In both these chapters, the multi-agent system is constant over time, i.e. the number of agents in the system does not evolve during the deployment. The present chapter deals with the case where the number of agents is time varying. A first Voronoi-based reconfiguration algorithm using a decentralized MPC approach is proposed for a time-varying multi-agent system (MAS). The considered dynamics for the agents and the reconfiguration strategy when either agents join or one agent leaves the MAS are detailed. Finally results on the case of incoming agents and one outgoing agent during the deployment are presented.

After analyzing the possible limitations of the previous reconfiguration algorithm for the case of several agents leaving the MAS, a robustified algorithm is proposed

to deal with the case of several simultaneous outgoing agents. A comparison is then made between the two algorithms for the case of one outgoing agent before presenting reconfiguration results in simulation for the case of multiple outgoing agents with the robustified algorithm.

The algorithm for the case of agents joining the multi-agent system has been introduced in Chevet et al. (2018). For the case of agents leaving the system, Chevet et al. (2018) presents a novel decentralized MPC-based reconfiguration algorithm. For its part, Chevet et al. (2020b) robustifies the algorithm of Chevet et al. (2018) by designing a novel safe objective along with a safety region in which the remaining agents of the MAS evolve to avoid potential collisions with the outgoing agents.

5.1 A first approach to reconfiguration

Section 5.1.1 inscribes the problem of the present chapter in the context of fault tolerant formation control. Then, Section 5.1.2 starts by introducing the general agent dynamics used throughout this chapter, before presenting two reconfiguration algorithms, one for the case of agents joining the multi-agent system, and one for the case of agents leaving this system. Finally, Section 5.1.3 presents reconfiguration results for both single integrator and nonlinear quadrotor unmanned aerial vehicle dynamics.

5.1.1 Motivation

This thesis presents several control algorithms for the deployment of a multi-agent system (MAS). Despite being subject to perturbations, bias or noise, each agent is always considered healthy, that is, it is not subject to any fault. Varga (2017) defines a fault as an unexpected variation of one or several physical parameters of a plant leading it to evolve outside its normal operation mode. A fault can occur on either an actuator, a sensor or an internal component (Blanke et al., 2000, Varga, 2017). A fault can be characterized by a modification of the input or output signal (e.g. in this sense, the output perturbation considered in Section 4.1.1 can result from a sensor fault) or a modification of the dynamics (e.g. a modification of the $\boldsymbol{A}$, $\boldsymbol{B}$ and/or $\boldsymbol{C}$ matrices from the dynamics (2.40)). When a system can be subject to a fault, a fault detection and isolation scheme has to be used to know when the system is faulty and on which part the fault acts exactly. When a system is faulty, a fault-tolerant control scheme (Zhang and Jiang, 2008, Amin and Hasan, 2019) can be used to mitigate the fault and allow the system to continue working despite the fault or reach a state in which it is possible to stop or repair it safely. Robust control or adaptive control fall into the category of fault-tolerant control (Blanke et al., 2000) and, in this sense, the tube-based model predictive control scheme presented in Section 4.1.3 can be deemed as fault-tolerant since, as mentioned earlier, the output perturbation can result from a sensor fault.

In the present chapter, the focus is not on a fault appearing on individual agents but rather on the multi-agent system itself. Such a fault consists in the addition or the removal of one or several agents from the MAS in the case of MAS deployment. The removal of agents can result from faults occurring on these agents, from the change in the mission objective resulting in the need of less agents in the MAS or even from the need for these agents to recharge. The addition of agents results

mainly from the need of more agents in the MAS to carry out its mission. For example, in a search and rescue mission after an avalanche, more agents could be needed to better cover the area around a victim after it was found. Then, the control algorithms presented in the following are part of a specific branch of fault-tolerant control (FTC), i.e. *fault-tolerant formation control* (FTFC).

FTFC strategies often rely on a predefined formation (Kamel et al., 2020). When one or several agents taking part in this formation are faulty, based on the severity of the fault, a modification on the control algorithm is made. Then, Kamel et al. (2020) define three categories of FTFC (also called *fault-tolerant cooperative control*) strategies in the case of unmanned ground vehicles that are immediately transposable to the case of any vehicle.

The first two are strategies that can be used to mitigate mild to moderate faults, i.e. faults that do not incapacitate the agent, namely individual FTFC and motion re-coordination. Individual FTFC consists in using a FTC strategy (Blanke et al., 2016) directly on the faulty agent to mitigate the fault effect while the other agents continue their mission as if nothing happened. In this case, used for example in Zhou et al. (2014), Xu et al. (2014), Hua et al. (2017) or Khalili et al. (2019), the formation behavior is not taken into account for the mitigation strategy and each faulty agent adapts its control law to be able to continue the mission of the MAS. For its part, motion re-coordination consists in adapting the control algorithm of both healthy and faulty agents. It takes into account the capabilities of the faulty agents such that the healthy ones limit their movement to maintain the formation's integrity or carry out a larger part of the mission to lighten the faulty agents' task. For example, when the mission involves path following, a re-planning of the trajectory is performed so that, even with limited physical capabilities, each agent is able to follow the new trajectory. This kind of method is find for example in Chamseddine et al. (2012), Wang et al. (2014) or Yu et al. (2016). In the case of consensus-based algorithms, such as in Saboori and Khorasani (2015), the motion re-coordination strategy involves the decrease of the importance of the faulty agents to carry out the mission.

Finally, the third category is the one to which the algorithms of this chapter belong to: the category of task assignment strategies. However, while Kamel et al. (2020) limit it to the case of severe fault occurring on one or several agents, here, only moderate faults (or even no faults) are considered. Nevertheless, such strategies come from the need for a faulty (or not) agent to leave the formation or, in the present case, for a healthy agent to join the formation. Then, the healthy agents remaining in the formation have to reconfigure themselves to adopt a new formation either without the outgoing agents (faulty or not) or with the incoming healthy agents. In the case of path following for a formation of vehicles, a natural strategy, used for example in Kamel et al. (2015) or Hafez and Kamel (2016), is to "abandon" the faulty agent along the way. Then, the healthy agents adopt a new formation shape while avoiding collision with the agent left behind. Strategies of the same kind can be found in Ghamry and Zhang (2016) or Huang et al. (2019).

In the present work, no path following or any global movement of the formation is involved. Thus, task assignment strategies as presented in the cited publications are not easily applicable. Indeed, these methods, proposed for MAS composed of various kinds of vehicles, involve leaving faulty agents where they are, while the others continue their movement. For the deployment problem, it is not possible for the faulty agent to be abandoned where it is since it would become an obstacle to

the deployment of the MAS. In this chapter, task assignment methods are tinged with motion re-coordination. Indeed, the idea is to have the healthy agents making room for the faulty ones to leave the workspace without colliding with other agents. The collision avoidance concern is then left to the healthy agents with respect to the limited physical capabilities of the faulty agents, hence the motion re-coordination idea. However, since the main objective is to remove (or integrate) agents from the formation, the algorithms of this chapter indeed belong to the task assignment family.

The proposed methods are meant to be applied on any type of multi-vehicle system. Then, since they can be applied to unmanned ground or surface vehicles, they are developed for a deployment and reconfiguration mission in a two-dimensional planar area. It could be argued that for vehicles able to move in the three spatial directions, such as unmanned aerial vehicles, the agents leaving the workspace in which the MAS is deployed could change their altitude and then move away from the workspace. However, if such a vehicle is faulty, it could be impossible for it to increase its altitude, and in missions such as forest fire monitoring, decreasing the altitude might be dangerous for the vehicle, leading potentially to its total loss. For these reasons, the methods presented in the following are developed for a planar movement, despite the potential three-dimensional movement capabilities of some vehicles.

In the case of outgoing agents, the control algorithms presented in this chapter are based on a novel transient objective for the healthy agents ensuring collision avoidance with the faulty agents by construction. These algorithms are based on the work of Chevet et al. (2018) and Chevet et al. (2020b). The control algorithm for the case of incoming agents incorporates the new vehicles in a natural extension of the deployment algorithm presented in Chapter 3. It is based on the work of Chevet et al. (2018).

5.1.2 Problem formulation

5.1.2.1 Agent dynamics

Let Σ be a multi-agent system composed of N agents. Each agent obeys discrete-time linear time-invariant dynamics as in Chapter 3. The general dynamics are the same as the one used in Chapter 3, i.e.:

$$\mathbf{x}_i(k+1) = \boldsymbol{A}\mathbf{x}_i(k) + \boldsymbol{B}\mathbf{u}_i(k) \tag{5.1a}$$

$$\mathbf{y}_i(k) = \boldsymbol{C}\mathbf{x}_i(k) \tag{5.1b}$$

where $\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^n$, $\mathbf{u}_i \in \mathcal{U} \subset \mathbb{R}^m$, $\mathbf{y}_i \in \mathcal{Y} \subset \mathbb{R}^2$, $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{B} \in \mathbb{R}^{n\times m}$ and $\boldsymbol{C} \in \mathbb{R}^{2\times n}$, with $i \in \overline{1, N}$. As in the previous chapters, the multi-agent system is homogeneous by Assumption 2.6, the dependency in the identifier i is dropped for the matrices $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ as well as for the state, input and output spaces $\mathcal{X}$, $\mathcal{U}$ and $\mathcal{Y}$. All the agents share the same two-dimensional output space (or workspace) by Assumption 2.3 and the system (5.1) admits equilibrium points as per Assumption 3.2.

When the multi-agent system operates normally, each agent follows the deployment algorithm presented in Section 3.2.2. The properties of the model predictive controller are the same as the one described in Section 3.2.2. In this case, the algorithm followed by the MAS presents no difference with Algorithm 3.2.

However, this chapter aims to improve this algorithm to make it resilient to the addition or removal of one or several agents to the MAS. Thus, in the nominal case, i.e. when the system operates without modification in the number of agents, it is clear that the MAS is simply deployed in the workspace $\mathcal{Y}$ and the algorithm needs not be modified. Nevertheless, Paragraph 5.1.2.2 introduces the modification of the algorithm to make it able to integrate agents in the MAS. Moreover, Paragraph 5.1.2.3 presents a new algorithm to make the MAS able to be deployed in $\mathcal{Y}$ while agents leave $\mathcal{Y}$ without causing collisions with the remaining agents.

5.1.2.2 Incoming agents

Let E be a set of agents present in $\mathbb{R}^2 \setminus \mathcal{Y}$. The objective for the incoming agents of E (which can also be called *entering agents*) is to join Σ, while Σ is deployed inside $\mathcal{Y}$. Thus, the goal for E is to enter $\mathcal{Y}$. After entering $\mathcal{Y}$, the set E is part of Σ and participates in the deployment of the multi-agent system Σ. With this in mind, the following paragraph is a natural extension of the decentralized deployment algorithm presented in Section 3.2.2 to allow the agents outside the workspace to participate in the deployment of Σ.

The objective of the present control algorithm is then to drive the agents of E inside the workspace $\mathcal{Y}$ with a decentralized model predictive controller. To do so, an objective point $\mathbf{y}_i^{\mathrm{E}}(k)$ for each agent $i \in \overline{1, |\mathrm{E}|}$ of E is defined as the point in the interior of $\mathcal{Y}$, i.e. the point of $\mathcal{Y} \setminus \partial\mathcal{Y}$, where $\partial\mathcal{Y}$ is the border of $\mathcal{Y}$, closer to the position $\mathbf{y}_i(k)$ of agent i than any other point of $\mathcal{Y} \setminus \partial\mathcal{Y}$. This property is translated mathematically as:

$$\mathbf{y}_i^{\mathrm{E}}(k) = \underset{\mathbf{y} \in \mathcal{Y} \setminus \partial\mathcal{Y}}{\arg\min} \|\mathbf{y}_i(k) - \mathbf{y}\|_2, \ \forall i \in \overline{1, |\mathrm{E}|}. \tag{5.2}$$

This point $\mathbf{y}_i^{\mathrm{E}}(k)$ is chosen since it is computationally easy to obtain and, given the deployment strategy followed by the agents of Σ, the incoming agents of E enter the workspace $\mathcal{Y}$ at a point relatively far from the position of the deploying agents of Σ.

Then, since $\mathbf{y}_i^{\mathrm{E}}(k) \in \mathcal{Y}$, it is possible, according to Assumption 3.2, to find a couple $\left(\mathbf{x}_i^{\mathrm{E}}(k), \mathbf{u}_i^{\mathrm{E}}(k)\right)$ such that $\left(\mathbf{x}_i^{\mathrm{E}}(k), \mathbf{u}_i^{\mathrm{E}}(k), \mathbf{y}_i^{\mathrm{E}}(k)\right)$ is an equilibrium point of the dynamics (5.1):

$$\begin{aligned} \mathbf{x}_i^{\mathrm{E}}(k) &= \boldsymbol{A}\mathbf{x}_i^{\mathrm{E}}(k) + \boldsymbol{B}\mathbf{u}_i^{\mathrm{E}}(k) \\ \mathbf{y}_i^{\mathrm{E}}(k) &= \boldsymbol{C}\mathbf{x}_i^{\mathrm{E}}(k). \end{aligned} \tag{5.3}$$

The couple $\left(\mathbf{x}_i^{\mathrm{E}}(k), \mathbf{u}_i^{\mathrm{E}}(k)\right)$ is then used as a reference point for the decentralized MPC algorithm driving agent i of E. When agent i enters $\mathcal{Y}$, it switches from E to Σ and starts participating in the deployment of Σ by following Algorithm 3.2.

While the agents of Σ follow Algorithm 3.2, the agents of E compute their input

$\mathbf{u}_i(k)$, with $i \in \overline{1, |\mathrm{E}|}$, by finding the solution to the optimization problem:

$$\underset{\substack{\mathbf{u}_i(k+l),\\ \forall l \in \overline{0, N_p-1}}}{\text{minimize}} \quad \sum_{l=0}^{N_p-1} \ell\big(\mathbf{x}_i(k+l), \mathbf{u}_i(k+l), \mathbf{x}_i^{\mathrm{E}}(k), \mathbf{u}_i^{\mathrm{E}}(k)\big) + V\big(\mathbf{x}_i(k+N_p), \mathbf{x}_i^{\mathrm{E}}(k)\big) \tag{5.4a}$$

subject to

$$\mathbf{x}_i(k+l+1) = \boldsymbol{A}\mathbf{x}_i(k+l) + \boldsymbol{B}\mathbf{u}_i(k+l), \qquad \forall l \in \overline{0, N_p-1}, \tag{5.4b}$$

$$\mathbf{x}_i(k+l) \in \mathcal{X}_{\mathrm{E}}, \qquad \forall l \in \overline{0, N_p-1}, \tag{5.4c}$$

$$\mathbf{u}_i(k+l) \in \mathcal{U}, \qquad \forall l \in \overline{0, N_p-1} \tag{5.4d}$$

where $\ell\big(\mathbf{x}_i(k+l), \mathbf{u}_i(k+l), \mathbf{x}_i^{\mathrm{E}}(k), \mathbf{u}_i^{\mathrm{E}}(k)\big)$, with $l \in \overline{0, N_p-1}$, is the stage cost:

$$\ell\big(\mathbf{x}_i(k+l), \mathbf{u}_i(k+l), \mathbf{x}_i^{\mathrm{E}}(k), \mathbf{u}_i^{\mathrm{E}}(k)\big) = \big\|\mathbf{x}_i(k+l) - \mathbf{x}_i^{\mathrm{E}}(k)\big\|^2_{\boldsymbol{Q}_i^{\mathrm{E}}} + \big\|\mathbf{u}_i(k+l) - \mathbf{u}_i^{\mathrm{E}}(k)\big\|^2_{\boldsymbol{R}_i^{\mathrm{E}}} \tag{5.5}$$

and $V\big(\mathbf{x}_i(k+N_p), \mathbf{x}_i^{\mathrm{E}}(k)\big)$ is the terminal cost:

$$V\big(\mathbf{x}_i(k+N_p), \mathbf{x}_i^{\mathrm{E}}(k)\big) = \big\|\mathbf{x}_i(k+N_p) - \mathbf{x}_i^{\mathrm{E}}(k)\big\|^2_{\boldsymbol{P}_i^{\mathrm{E}}}. \tag{5.6}$$

The weighting matrices $\boldsymbol{Q}_i^{\mathrm{E}}, \boldsymbol{P}_i^{\mathrm{E}} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{R}_i^{\mathrm{E}} \in \mathbb{R}^{m\times m}$ in (5.5) and (5.6) are chosen such that $\boldsymbol{Q}_i^{\mathrm{E}} = \boldsymbol{Q}_i^{\mathrm{E}\top} \succ 0$, $\boldsymbol{P}_i^{\mathrm{E}} = \boldsymbol{P}_i^{\mathrm{E}\top} \succ 0$ and $\boldsymbol{R}_i^{\mathrm{E}} = \boldsymbol{R}_i^{\mathrm{E}\top} \succ 0$. The prediction horizon N_p is a positive integer.

The constraint (5.4b) is used to predict the future state $\mathbf{x}_i(k+l+1)$ for all $l \in \overline{0, N_p-1}$ of the agent $i \in \overline{1, |\mathrm{E}|}$ given the value of the input sequence $\mathbf{u}_i(k+l)$ for all $l \in \overline{0, N_p-1}$. The other two constraints are meant to restrict the movement of agent i of E. Constraint (5.4d) is meant to ensure that, over the prediction horizon N_p, the input signal of the agent remains inside a convex polytope $\mathcal{U}$, the input space of the agent, obtained from limitations on the actuators (e.g. physical limitations) or from the considered application. Moreover, constraint (5.4c) ensures that, over the prediction horizon, the state remains inside a convex polyhedron $\mathcal{X}_{\mathrm{E}}$ obtained from the potential physical limitations of the system.

Remark 5.1: Output constraints
Contrary to the MPC optimization problem (3.12), $\mathcal{X}_{\mathrm{E}}$ does not impose constraints on the position of the agent since it is not restricted to evolve in $\mathcal{Y}$ but can move inside $\mathbb{R}^2$. ◊

For the reason invoked in Remark 5.1, contrary to the deployment algorithm of Section 3.2.2, no constraint on the output appears in problem (5.4). Indeed, for the agents of E, the objective is to rally Σ. Then, it is unnecessary to enforce the fact that the outputs of the agents belong to a given area.

Agent $i \in \overline{1, |\mathrm{E}|}$ uses (5.4) to compute its input signal until it reaches $\mathcal{Y}$. When this agent is such that $\mathbf{y}_i(k) \in \mathcal{Y}$, it leaves E and joins Σ. As soon as it joins Σ, it starts participating in the deployment of Σ in $\mathcal{Y}$ by following Algorithm 3.2. This procedure is summarized by Algorithm 5.1.

Such a procedure does not impact the agents already present in $\mathcal{Y}$. Indeed, the decentralized algorithm used for the deployment of the MAS, i.e. Algorithm 3.2, makes the agents dependent only on the knowledge of the position of the other

Algorithm 5.1: Decentralized algorithm for the inclusion of an agent to the MAS.

Input: The initial position $\mathbf{y}_i(0)$ of agent $i \in \overline{1, |\mathrm{E}|}$

$k \leftarrow 0$;
while $\mathbf{y}_i(k) \notin \mathcal{Y}$ **do**
  Compute the objective position $\mathbf{y}_i^{\mathrm{E}}(k) = \underset{\mathbf{y} \in \mathcal{Y} \setminus \partial\mathcal{Y}}{\arg\min} \|\mathbf{y}_i(k) - \mathbf{y}\|_2$;
  Compute $\left(\mathbf{x}_i^{\mathrm{E}}(k), \mathbf{u}_i^{\mathrm{E}}(k)\right)$ with (5.3) such that $\left(\mathbf{x}_i^{\mathrm{E}}(k), \mathbf{u}_i^{\mathrm{E}}(k), \mathbf{y}_i^{\mathrm{E}}(k)\right)$ is an equilibrium point of (5.1);
  Apply $\mathbf{u}_i(k)$, first element of the solution of the optimization problem (5.4) to agent i;
  $k \leftarrow k + 1$;
end
$k_{\mathrm{in}} \leftarrow k$;
for $k \geq k_{\mathrm{in}}$ **do**
  Follow Algorithm 3.2;
end

agents. Thus, it is naturally resilient to the inclusion of any number of agents to the MAS (as long as the agents can fit inside $\mathcal{Y}$ without colliding with each other) and it does not need to be modified.

5.1.2.3 Outgoing agents

Let Σ be the MAS described in Paragraph 5.1.2.1. All the agents of Σ follow Algorithm 3.2 to deploy into the workspace $\mathcal{Y}$. At some point during the deployment, one agent $o \in \overline{1, N}$, called *outgoing agent*, part of Σ, has to leave the MAS. Then, agent o is removed from Σ. Contrary to the case of incoming agents, the behavior of the agents remaining in $\mathcal{Y}$, while o leaves $\mathcal{Y}$, has to be modified.

Agent o is deemed non cooperative either because it is subject to a fault or because it is not needed anymore for the mission of Σ. It then has to leave the workspace $\mathcal{Y}$ to avoid impairing the other agents of Σ during their normal mission proceeding. To leave the workspace $\mathcal{Y}$, agent o follows an objective point $\mathbf{y}_o^{\mathrm{O}} \in \mathbb{R}^2 \setminus \mathcal{Y}$, this point being imposed by external processes out of the scope of the present thesis or by an external user, by using a decentralized model predictive control policy. The point $\mathbf{y}_o^{\mathrm{O}}$ is constant over time[1]. Given the dynamics of the vehicles considered in the present thesis, it is possible to expand Assumption 3.2 to a larger part of $\mathbb{R}^2$ than $\mathcal{Y}$ so that it is possible to find a couple $\left(\mathbf{x}_o^{\mathrm{O}}, \mathbf{u}_o^{\mathrm{O}}\right)$ such that $\left(\mathbf{x}_o^{\mathrm{O}}, \mathbf{u}_o^{\mathrm{O}}, \mathbf{y}_o^{\mathrm{O}}\right)$ is an equilibrium point of (5.1).

Let $\left(\mathbf{x}_o^{\mathrm{O}}, \mathbf{u}_o^{\mathrm{O}}\right)$ be such a couple, satisfying:

$$\begin{aligned} \mathbf{x}_o^{\mathrm{O}} &= \boldsymbol{A}\mathbf{x}_o^{\mathrm{O}} + \boldsymbol{B}\mathbf{u}_o^{\mathrm{O}} \\ \mathbf{y}_o^{\mathrm{O}} &= \boldsymbol{C}\mathbf{x}_o^{\mathrm{O}}. \end{aligned} \tag{5.7}$$

Even if the outgoing agent o is subject to a fault, the pair $\left(\mathbf{x}_o^{\mathrm{O}}, \mathbf{u}_o^{\mathrm{O}}\right)$ is computed with the nominal dynamics (5.1). Indeed, it is an objective point that the outgoing agent tries to reach while it is leaving $\mathcal{Y}$. What happens to agent o when it is outside $\mathcal{Y}$,

[1] However, the following results could be easily adapted to the case where $\mathbf{y}_o^{\mathrm{O}}$ is time-varying.

i.e. when the other agents of Σ can safely continue their mission, is out of the scope of the present work. Then, even if the dynamics of the outgoing agent is modified, due to a fault, it follows the equilibrium point $\left(\mathbf{x}_o^{\mathrm{O}}, \mathbf{u}_o^{\mathrm{O}}, \mathbf{y}_o^{\mathrm{O}}\right)$ of the nominal system to leave $\mathcal{Y}$ and then follows any suitable strategy to get to safety.

The outgoing agent $o \in \overline{1, N}$ computes its input $\mathbf{u}_o(k)$ by finding the solution to the optimization problem:

$$\underset{\substack{\mathbf{u}_o(k+l),\\ \forall l \in \overline{0, N_p-1}}}{\text{minimize}} \quad \sum_{l=0}^{N_p-1} \ell\left(\mathbf{x}_o(k+l), \mathbf{u}_o(k+l), \mathbf{x}_o^{\mathrm{O}}, \mathbf{u}_o^{\mathrm{O}}\right) + V\left(\mathbf{x}_o(k+N_p), \mathbf{x}_o^{\mathrm{O}}\right) \tag{5.8a}$$

subject to

$$\mathbf{x}_o(k+l+1) = \boldsymbol{A}\mathbf{x}_o(k+l) + \boldsymbol{B}\mathbf{u}_o(k+l), \qquad \forall l \in \overline{0, N_p-1}, \tag{5.8b}$$

$$\mathbf{x}_o(k+l) \in \mathcal{X}_{\mathrm{O}}, \qquad \forall l \in \overline{0, N_p-1}, \tag{5.8c}$$

$$\mathbf{u}_o(k+l) \in \mathcal{U}_{\mathrm{O}}, \qquad \forall l \in \overline{0, N_p-1} \tag{5.8d}$$

where $\ell\left(\mathbf{x}_o(k+l), \mathbf{u}_o(k+l), \mathbf{x}_o^{\mathrm{O}}, \mathbf{u}_o^{\mathrm{O}}\right)$, with $l \in \overline{0, N_p-1}$, is the stage cost:

$$\ell\left(\mathbf{x}_o(k+l), \mathbf{u}_o(k+l), \mathbf{x}_o^{\mathrm{O}}, \mathbf{u}_o^{\mathrm{O}}\right) = \left\|\mathbf{x}_o(k+l) - \mathbf{x}_o^{\mathrm{O}}\right\|_{\boldsymbol{Q}_o}^2 + \left\|\mathbf{u}_o(k+l) - \mathbf{u}_o^{\mathrm{O}}\right\|_{\boldsymbol{R}_o}^2 \tag{5.9}$$

and $V\left(\mathbf{x}_o(k+N_p), \mathbf{x}_o^{\mathrm{O}}\right)$ is the terminal cost:

$$V\left(\mathbf{x}_o(k+N_p), \mathbf{x}_o^{\mathrm{O}}\right) = \left\|\mathbf{x}_o(k+N_p) - \mathbf{x}_o^{\mathrm{O}}\right\|_{\boldsymbol{P}_o}^2. \tag{5.10}$$

The weighting matrices $\boldsymbol{Q}_o, \boldsymbol{P}_o \in \mathbb{R}^{n\times n}$ and $\boldsymbol{R}_o \in \mathbb{R}^{m\times m}$ in (5.9) and (5.10) are chosen such that $\boldsymbol{Q}_o = \boldsymbol{Q}_o^\top \succ 0$, $\boldsymbol{P}_o = \boldsymbol{P}_o^\top \succ 0$ and $\boldsymbol{R}_o = \boldsymbol{R}_o^\top \succ 0$. The prediction horizon N_p is a positive integer.

The constraint (5.8b) is used to predict the future state $\mathbf{x}_o(k+l+1)$ for all $l \in \overline{0, N_p-1}$ of the agent o given the value of the input sequence $\mathbf{u}_o(k+l)$ for all $l \in \overline{0, N_p-1}$. The other two constraints are meant to restrict the movement of agent o. Constraint (5.8d) is meant to ensure that, over the prediction horizon N_p, the input signal of the agent remains inside a convex polytope $\mathcal{U}_{\mathrm{O}}$, the input space of the agent. Moreover, constraint (5.8c) ensures that, over the prediction horizon, the state vector remains inside a convex polyhedron $\mathcal{X}_{\mathrm{O}}$. The sets $\mathcal{U}_{\mathrm{O}}$ and $\mathcal{X}_{\mathrm{O}}$ are obtained, as for the case of incoming agents, from the physical limitations of the actuators and of the system or from possible specific restrictions imposed by the considered application. Moreover, $\mathcal{X}_{\mathrm{O}}$ does not constrain the position of agent o since its objective is to leave $\mathcal{Y}$ to reach $\mathbb{R}^2 \setminus \mathcal{Y}$.

The problem (5.8) is written for a healthy agent having to leave the workspace because it is not needed anymore for the deployment mission. In the case of a faulty agent, it is modified to take into account potential modifications due to a fault. This modification impacts the three constraints of (5.8) since the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ of the dynamics appearing in (5.8b) are modified and the sets $\mathcal{U}_{\mathrm{O}}$ and $\mathcal{X}_{\mathrm{O}}$ are changed to account for the modified physical capabilities of agent o.

As for the incoming agent case of Paragraph 5.1.2.2, no constraint on the output appears in eq. (5.8). Indeed, since the goal of agent o is to leave the workspace $\mathcal{Y}$, it would be counterproductive to impose that its output remains inside a given area. The procedure followed by the outgoing agent is summarized by Algorithm 5.2.

Algorithm 5.2: Decentralized algorithm for the removal of the agent o from the MAS.

for $k \geq 0$ **do**
  if *agent o leaves* Σ **then**
    $k_{\text{out}} \leftarrow k$;
    Go to 9;
  else
    Follow Algorithm 3.2;
  end
end
Acquire the objective point $\mathbf{y}_o^{\text{O}}$;
Compute the couple $\left(\mathbf{x}_o^{\text{O}}, \mathbf{u}_o^{\text{O}}\right)$ with (5.7) such that $\left(\mathbf{x}_o^{\text{O}}, \mathbf{u}_o^{\text{O}}, \mathbf{y}_o^{\text{O}}\right)$ is an equilibrium point of (5.1);
for $k \geq k_{\text{out}}$ **do**
  Solve the optimization problem (5.8) to obtain the input signal $\mathbf{u}_o(k)$;
  Apply $\mathbf{u}_o(k)$ to the agent;
end

However, contrary to the case of Paragraph 5.1.2.2, the other agents of Σ which do not leave the workspace $\mathcal{Y}$ do not follow Algorithm 3.2. As mentioned earlier, their behavior is modified to avoid collisions with the outgoing agent o. This first approach has been introduced in Chevet et al. (2018).

Assumption 5.1: Knowledge of the outgoing agent

The agents of Σ know the position $\mathbf{y}_o(k)$ and the objective $\mathbf{y}_o^{\text{O}}$ of the outgoing agent o.

Definition 5.1: Neighbor of an agent

For an agent $i \in \overline{1, N} \setminus \{o\}$ of Σ, a neighbor ν of agent i is one of the following:

- another agent of Σ having a Voronoi cell contiguous to $\mathcal{V}_i(k)$;
- the outgoing agent o if its Voronoi cell is contiguous to $\mathcal{V}_i(k)$;
- a vertex of $\mathcal{V}_i(k)$ lying on $\partial\mathcal{Y}$.

The set of all neighbors of agent i at time k is denoted by $\text{N}_i(k)$.

The position $\mathbf{y}_o(k) \in \mathbb{R}^2$ and the objective $\mathbf{y}_o^{\text{O}}$ of the outgoing agent o define a hyperplane $\partial\mathcal{H}_o(k) = \{\mathbf{x} \in \mathbb{R}^2 \mid \boldsymbol{h}_o(k)\mathbf{x} = \theta_o(k)\}$ in $\mathbb{R}^2$, where $\boldsymbol{h}_o(k) \in \mathbb{R}^{1\times 2}$ and $\theta_o(k) \in \mathbb{R}$, which is known by each agent of Σ as per Assumption 5.1. Then, a new objective point, different from the Chebyshev center $\mathbf{c}_i(k)$ of the Voronoi cell $\mathcal{V}_i(k)$, is computed for each agent $i \in \overline{1, N} \setminus \{o\}$. This new objective point is the barycenter $\mathbf{y}_i^{\text{b}}(k) \in \mathbb{R}^2$ of the neighbors of agent i as defined in Definition 5.1:

$$\mathbf{y}_i^{\text{b}}(k) = \frac{\displaystyle\sum_{\nu \in \text{N}_i(k)} \omega_{\nu,i}(k)\mathbf{y}_\nu(k)}{\displaystyle\sum_{\nu \in \text{N}_i(k)} \omega_{\nu,i}(k)} \tag{5.11}$$

where $\omega_{\nu,i}(k)$ is a weight attributed to the neighbor $\nu \in \mathrm{N}_i(k)$ of agent $i \in \overline{1,N} \setminus \{o\}$ depending on its relative position to $\partial\mathcal{H}_o(k)$ and $\mathbf{y}_\nu(k) \in \mathbb{R}^2$ is the position of the neighbor $\nu \in \mathrm{N}_i(k)$. A procedure has then to be designed to attribute the weights $\omega_{\nu,i}(k)$ to the neighbors.

Let $d(\mathbf{y}_i(k), \partial\mathcal{H}_o(k))$ denote the distance of agent $i \in \overline{1,N} \setminus \{o\}$ to the hyperplane $\partial\mathcal{H}_o(k)$ defined as:

$$d(\mathbf{y}_i(k), \partial\mathcal{H}_o(k)) = \min_{\mathbf{y} \in \partial\mathcal{H}_o(k)} \|\mathbf{y}_i(k) - \mathbf{y}\|_2.$$

If the conditions:

- $d(\mathbf{y}_\nu(k), \partial\mathcal{H}_o(k)) \geq d(\mathbf{y}_i(k), \partial\mathcal{H}_o(k))$, with $\nu \in \mathrm{N}_i(k)$, i.e. the neighbor $\nu \in \mathrm{N}_i(k)$ is farther from $\partial\mathcal{H}_o(k)$ than agent i;
- $\boldsymbol{h}_o(k)(\mathbf{y}_\nu(k) - \mathbf{y}_o(k)) \cdot \boldsymbol{h}_o(k)(\mathbf{y}_i(k) - \mathbf{y}_o(k)) \geq 0$, with $\nu \in \mathrm{N}_i(k)$, i.e. the neighbor ν and agent i belong to the same half-space delimited by $\partial\mathcal{H}_o(k)$

are met, then $\omega_{\nu,i}(k) = \kappa$, where $\kappa \in \mathbb{R}$, with $\kappa > 1$. Else, $\omega_{\nu,i}(k) = 1$. With such a choice of weights, the barycenter ends up driving the remaining agent i away from the trajectory of the outgoing agent, thus leaving room for the outgoing agent to leave $\mathcal{Y}$ safely. This procedure is summarized in Algorithm 5.3.

Algorithm 5.3: Computation of the neighbors' barycenter of a remaining agent of Σ.

Input: The list of neighbors $\mathrm{N}_i(k)$ of agent $i \in \overline{1,N} \setminus \{o\}$, the position of the neighbors $\mathbf{y}_\nu(k)$, $\forall \nu \in \mathrm{N}_i(k)$, the position of the outgoing agent $\mathbf{y}_o(k)$, the hyperplane $\partial\mathcal{H}_o(k) = \{\mathbf{x} \in \mathbb{R}^2 \mid \boldsymbol{h}_o(k)\mathbf{x} \leq \theta_o(k)\}$

```
for ν ∈ N_i(k) do
    if h_o(k)(y_ν(k) − y_o(k)) · h_o(k)(y_i(k) − y_o(k)) ≥ 0 then
        if d(y_ν(k), ∂H_o(k)) > d(y_i(k), ∂H_o(k)) then
            ω_{ν,i}(k) ← κ;
        else
            ω_{ν,i}(k) ← 1;
        end
    else
        ω_{ν,i}(k) ← 1;
    end
end
Compute the neighbors' barycenter of agent i with (5.11);
```

Output: The neighbors' barycenter $\mathbf{y}_i^{\mathrm{b}}(k)$ of agent i

Example 5.1: Construction of a neighbor's barycenter

Let Σ be a MAS composed of 10 agents deployed in $\mathcal{Y} = \mathbb{B}^2(10 \cdot \mathbf{1}_{2\times 1})$ as presented in Figure 5.1. At a given time k, agent 3 has to leave Σ and the workspace $\mathcal{Y}$. It has for objective $\mathbf{y}_3^{\mathrm{O}} = \begin{bmatrix} -9 & -12 \end{bmatrix}^\top \in \mathbb{R}^2 \setminus \mathcal{Y}$, chosen arbitrarily and depicted by asterisks in Figure 5.1, which defines with $\mathbf{y}_3(k)$ the hyperplane $\partial\mathcal{H}_3(k) = \{\mathbf{x} \in \mathbb{R}^2 \mid \boldsymbol{h}_3(k)\mathbf{x} = \theta_3(k)\}$, represented as a solid line in Figure 5.1.

This example then presents how the neighbors' barycenters of the agents 2 and 5 are obtained. The hyperplanes $\partial\mathcal{D}_2(k)$ and $\partial\mathcal{D}_5(k)$, represented with dashed lines,

are the sets of all the points of $\mathbb{R}^2$ *at the same distance of the hyperplane* $\partial\mathcal{H}_3(k)$ *as agents 2 and 5, respectively, defined as:*

$$\partial\mathcal{D}_i(k) = \left\{\mathbf{x} \in \mathbb{R}^2 \;\middle|\; d(\mathbf{x}, \partial\mathcal{H}_3(k)) = d(\mathbf{y}_i(k), \partial\mathcal{H}_3(k))\right\}$$

where $i \in \{2, 5\}$. *They are presented in Figure 5.1 to graphically compare the distances but are not necessary for the real procedure.*

Agents 4, 5, 8, 9 and 10 have a Voronoi cell contiguous to $\mathcal{V}_2(k)$, *the Voronoi cell* $\mathcal{V}_3(k)$ *of the outgoing agent 3 is also contiguous to* $\mathcal{V}_2(k)$ *and* 11 *and* 12 *are vertices of* $\mathcal{V}_2(k)$ *lying on* $\partial\mathcal{Y}$. *Then, by Definition 5.1, the set of neighbors of agent 2 is* $\mathrm{N}_2(k) = \{3, 4, 5, 8, 9, 10, 11, 12\}$. *Agents 2, 6 and 9 have a Voronoi cell contiguous to* $\mathcal{V}_5(k)$ *and the Voronoi cell* $\mathcal{V}_3(k)$ *of the outgoing agent 3 is contiguous to* $\mathcal{V}_5(k)$. *Then, by Definition 5.1, the set of neighbors of agent 5 is* $\mathrm{N}_5(k) = \{2, 3, 6, 9\}$.

It can be seen in Figure 5.1 that $d(\mathbf{y}_\nu(k), \partial\mathcal{H}_3(k)) \geq d(\mathbf{y}_2(k), \partial\mathcal{H}_3(k))$ *and* $\boldsymbol{h}_3(k)(\mathbf{y}_\nu(k) - \mathbf{y}_3(k)) \cdot \boldsymbol{h}_3(k)(\mathbf{y}_2(k) - \mathbf{y}_3(k)) \geq 0$ *for all* $\nu \in \{8, 10, 11, 12\}$. *Then,* $\omega_{\nu,2}(k) = \kappa$ *for all* $\nu \in \{8, 10, 11, 12\}$, *while* $\omega_{\nu,2}(k) = 1$ *for all* $\nu \in \{3, 4, 5, 9\}$. *Again, it can be seen in Figure 5.1 that only neighbor 2 is located further from* $\partial\mathcal{H}_3(k)$ *than agent 5 among the neighbors that are on the same side of* $\partial\mathcal{H}_3(k)$ *as agent 5, thus* $d(\mathbf{y}_2(k), \partial\mathcal{H}_3(k)) \geq d(\mathbf{y}_5(k), \partial\mathcal{H}_3(k))$ *and* $\boldsymbol{h}_3(k)(\mathbf{y}_2(k) - \mathbf{y}_3(k)) \cdot \boldsymbol{h}_3(k)(\mathbf{y}_5(k) - \mathbf{y}_3(k)) \geq 0$. *Then* $\omega_{2,5}(k) = \kappa$, *while* $\omega_{\nu,5}(k) = 1$ *for all* $\nu \in \{3, 6, 9\}$.

From (5.11), the barycenters are:

$$\mathbf{y}_2^{\mathrm{b}}(k) = \frac{\kappa(\mathbf{y}_8(k) + \mathbf{y}_{10}(k) + \mathbf{y}_{11}(k) + \mathbf{y}_{12}(k)) + \mathbf{y}_3(k) + \mathbf{y}_4(k) + \mathbf{y}_5(k) + \mathbf{y}_9(k)}{4(\kappa + 1)}$$

$$\mathbf{y}_5^{\mathrm{b}}(k) = \frac{\kappa\mathbf{y}_2(k) + \mathbf{y}_3(k) + \mathbf{y}_6(k) + \mathbf{y}_9(k)}{\kappa + 3}$$

where $\mathbf{y}_{11}(k)$ *and* $\mathbf{y}_{12}(k)$ *are the coordinates in* $\mathbb{R}^2$ *of the vertices of* $\mathcal{V}_2(k)$ *lying on* $\partial\mathcal{Y}$.

When $\kappa = 3$, *the neighbors' barycenter of agents 2 and 5 are presented in Figure 5.1 with squares. The other neighbors' barycenter of agents 1, 4, 6, 7, 8, 9 and 10 can then be obtained the same way.*

Remark 5.2: Heterogeneous MAS
In the case of heterogeneous MAS, different weights κ_i could be considered for different type of vehicles. ◊

While the outgoing agent o is inside $\mathcal{Y}$, i.e. while $\mathbf{y}_o(k) \in \mathcal{Y}$, the objective point of the agents of Σ is changed from the Chebyshev center $\mathbf{c}_i(k)$ of the Voronoi cell $\mathcal{V}_i(k)$, with $i \in \overline{1, N} \setminus \{o\}$, to the neighbors' barycenter $\mathbf{y}_i^{\mathrm{b}}(k)$. With such an objective point, the agents of Σ are driven away from the trajectory of the outgoing agent o represented by the hyperplane $\partial\mathcal{H}_o(k)$.

According to Assumption 3.2, there exists a couple $\left(\mathbf{x}_i^{\mathrm{b}}(k), \mathbf{u}_i^{\mathrm{b}}(k)\right)$, with $i \in \overline{1, N} \setminus \{o\}$, such that $\left(\mathbf{x}_i^{\mathrm{b}}(k), \mathbf{u}_i^{\mathrm{b}}(k), \mathbf{y}_i^{\mathrm{b}}(k)\right)$ is an equilibrium point of (5.1) satisfying:

$$\begin{aligned} \mathbf{x}_i^{\mathrm{b}}(k) &= \boldsymbol{A}\mathbf{x}_i^{\mathrm{b}}(k) + \boldsymbol{B}\mathbf{u}_i^{\mathrm{b}}(k) \\ \mathbf{y}_i^{\mathrm{b}}(k) &= \boldsymbol{C}\mathbf{x}_i^{\mathrm{b}}(k). \end{aligned} \tag{5.12}$$

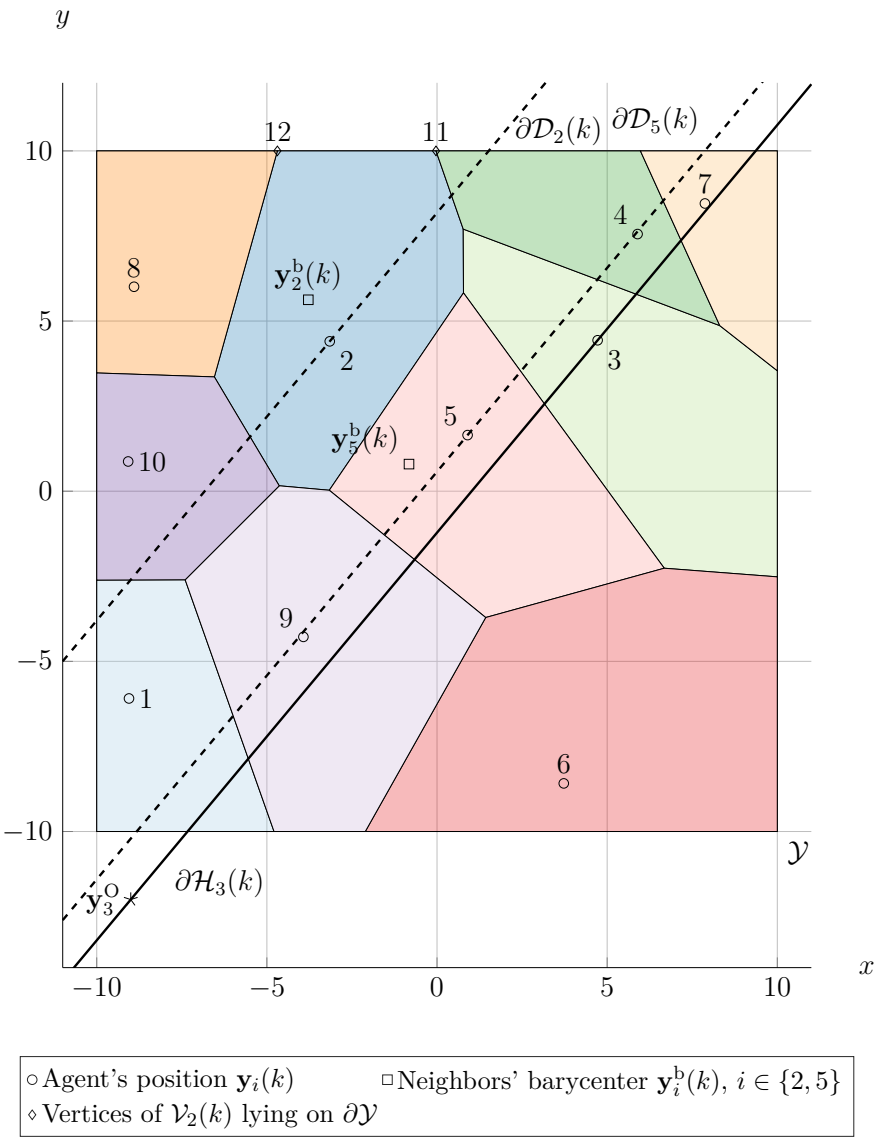


Figure 5.1: Example of construction of a neighbors' barycenter for MAS reconfiguration when one agent leaves the workspace.

Then, the agents of Σ compute their input $\mathbf{u}_i(k)$, with $i \in \overline{1, N} \setminus \{o\}$, by finding the solution of the optimization problem:

$$\underset{\substack{\mathbf{u}_i(k+l),\\ \forall l \in \overline{0, N_p-1}}}{\text{minimize}} \quad \sum_{l=0}^{N_p-1} \ell\big(\mathbf{x}_i(k+l), \mathbf{u}_i(k+l), \mathbf{x}_i^{\mathrm{b}}(k), \mathbf{u}_i^{\mathrm{b}}(k)\big) + V\big(\mathbf{x}_i(k+N_p), \mathbf{x}_i^{\mathrm{b}}(k)\big) \tag{5.13a}$$

subject to

$$\mathbf{x}_i(k+l+1) = \boldsymbol{A}\mathbf{x}_i(k+l) + \boldsymbol{B}\mathbf{u}_i(k+l), \qquad \forall l \in \overline{0, N_p-1}, \tag{5.13b}$$

$$\mathbf{x}_i(k+l) \in \mathcal{X}, \qquad \forall l \in \overline{0, N_p-1}, \tag{5.13c}$$

$$\mathbf{u}_i(k+l) \in \mathcal{U}, \qquad \forall l \in \overline{0, N_p-1}, \tag{5.13d}$$

$$\boldsymbol{C}\mathbf{x}_i(k+l) \in \mathcal{V}_i(k), \qquad \forall l \in \overline{0, N_p-1}, \tag{5.13e}$$

$$\mathbf{x}_i(k+N_p) \in \Omega_i(k) \tag{5.13f}$$

where $\ell\big(\mathbf{x}_i(k+l), \mathbf{u}_i(k+l), \mathbf{x}_i^{\mathrm{b}}(k), \mathbf{u}_i^{\mathrm{b}}(k)\big)$, with $l \in \overline{0, N_p - 1}$, is the stage cost:

$$\ell\big(\mathbf{x}_i(k+l), \mathbf{u}_i(k+l), \mathbf{x}_i^{\mathrm{b}}(k), \mathbf{u}_i^{\mathrm{b}}(k)\big) = \big\|\mathbf{x}_i(k+l) - \mathbf{x}_i^{\mathrm{b}}(k)\big\|_{\boldsymbol{Q}_i}^2 + \big\|\mathbf{u}_i(k+l) - \mathbf{u}_i^{\mathrm{b}}(k)\big\|_{\boldsymbol{R}_i}^2 \quad (5.14)$$

and $V\big(\mathbf{x}_i(k+N_p), \mathbf{x}_i^{\mathrm{b}}(k)\big)$ is the terminal cost:

$$V\big(\mathbf{x}_i(k+N_p), \mathbf{x}_i^{\mathrm{b}}(k)\big) = \big\|\mathbf{x}_i(k+N_p) - \mathbf{x}_i^{\mathrm{b}}(k)\big\|_{\boldsymbol{P}_i}^2. \quad (5.15)$$

The weighting matrices $\boldsymbol{Q}_i, \boldsymbol{P}_i \in \mathbb{R}^{n\times n}$ and $\boldsymbol{R}_i \in \mathbb{R}^{m\times m}$ in (5.14) and (5.15) are chosen such that $\boldsymbol{Q}_i = \boldsymbol{Q}_i^\top \succ 0$, $\boldsymbol{P}_i = \boldsymbol{P}_i^\top \succ 0$ and $\boldsymbol{R}_i = \boldsymbol{R}_i^\top \succ 0$. The prediction horizon N_p is a positive integer.

All the constraints of problem (5.13) have the same objective as the constraints of the nominal case problem (3.12) described in Section 3.2.2. Then, the meaning of the constraints of problem (5.13) are not described here.

As soon as the outgoing agent o is outside $\mathcal{Y}$, i.e. when $\mathbf{y}_o(k) \in \mathbb{R}^2 \setminus \mathcal{Y}$, the agents of Σ, which are still in the workspace $\mathcal{Y}$ resume their deployment following Algorithm 3.2. The algorithm followed by the agents of Σ during the extraction of an agent is summarized in Algorithm 5.4.

Algorithm 5.4: Decentralized algorithm followed by a remaining agent for the reconfiguration of the MAS when only one agent leaves the system.

```
k_0 ← 0;
for k ≥ k_0 do
    if agent o leaves Σ and y_o(k) ∈ 𝒴 then
        Compute the neighbors' barycenter y_i^b(k) of agent i ∈ 1,N \ {o} with
         Algorithm 5.3;
        Compute the couple (x_i^b(k), u_i^b(k)) with (5.12) such that
         (x_i^b(k), u_i^b(k), y_i^b(k)) is an equilibrium point of (5.1);
        Solve the optimization problem (5.8) to obtain the input signal u_i(k);
        Apply u_i(k) to the agent;
    else
        Follow Algorithm 3.2;
    end
end
```

5.1.3 Deployment results

5.1.3.1 Incoming agents

Let Σ be a multi-agent system composed of $N = 10$ agents deployed in the workspace:

$$\mathcal{Y} = \left\{ \mathbf{x} \in \mathbb{R}^2 \;\middle|\; \begin{bmatrix} 3 & 2 \\ 0 & 1 \\ -1 & 2 \\ -1 & -1 \\ 2 & -1 \end{bmatrix} \mathbf{x} \leq \begin{bmatrix} 24 \\ 6 \\ 9 \\ 15 \\ 12 \end{bmatrix} \right\}.$$

Three agents 11, 12 and 13 start respectively from:

$$\mathbf{y}_{11}(0) = \begin{bmatrix} -10 \\ -10 \end{bmatrix} \qquad \mathbf{y}_{12}(0) = \begin{bmatrix} -10 \\ 5 \end{bmatrix} \qquad \mathbf{y}_{13}(0) = \begin{bmatrix} 5 \\ -10 \end{bmatrix}$$

while the other agents start from random positions $\mathbf{y}_i(0) \in \mathcal{Y}$, with $i \in \overline{1, N}$. All the agents obey the single integrator dynamics (3.15) and the reconfiguration algorithm in the case of incoming agents is tested only with agents obeying this dynamics since, as for the nominal deployment case of Chapter 3, the results are quite similar in the UAV dynamics case.

The objective of the agents of Σ is to deploy into a Chebyshev configuration as in Chapter 3 while agents 11, 12 and 13 aim to join $\mathcal{Y}$ to integrate Σ and participate in the deployment. Their initial objectives to enter the workspace are, respectively:

$$\mathbf{y}^{\mathrm{E}}_{11}(0) = \begin{bmatrix} -7.125 \\ -7.125 \end{bmatrix} \qquad \mathbf{y}^{\mathrm{E}}_{12}(0) = \begin{bmatrix} -7.71 \\ 0.42 \end{bmatrix} \qquad \mathbf{y}^{\mathrm{E}}_{13}(0) = \begin{bmatrix} 1.56 \\ -8.28 \end{bmatrix}$$

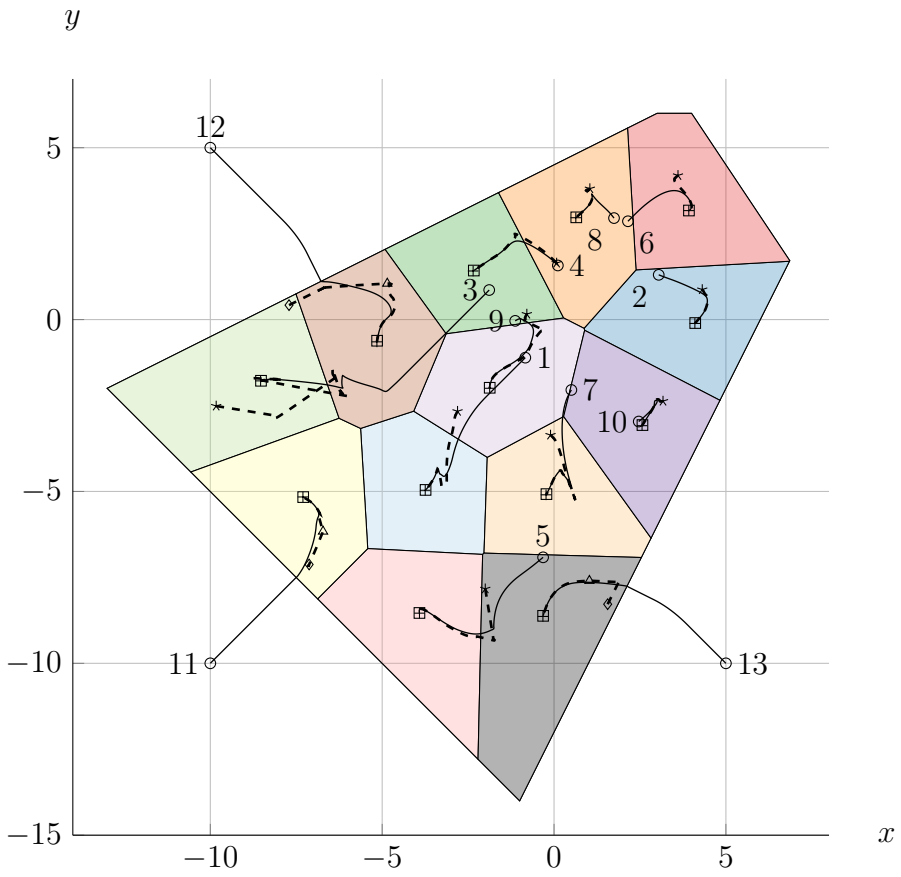
as obtained from (5.2). Then, from (5.3), since $\boldsymbol{A} = \boldsymbol{I}_2$, $\boldsymbol{B} = T_s\boldsymbol{I}_2$, where T_s is the sampling period, and $\boldsymbol{C} = \boldsymbol{I}_2$, $\mathbf{x}^{\mathrm{E}}_i(0) = \mathbf{y}^{\mathrm{E}}_i(0)$ and $\mathbf{u}^{\mathrm{E}}_i(0) = \mathbf{0}_{2\times 1}$ for all $i \in \{11, 12, 13\}$.

The sampling period used is $T_s = 0.2\,\mathrm{s}$ and the contraction factor for the terminal constraint is $\lambda_i = 0.9$ with a prediction horizon $N_p = 10$. The results exposed here are more succinct than those presented in the previous paragraphs since they are close to what happens in the nominal case in Section 3.2.4.

The considered dynamics is the same as the one presented in Section 3.2.4. The input set is then still $\mathcal{U} = \mathbb{B}^2(2 \cdot \mathbf{1}_{2\times 1})$, while the weighting matrices are chosen such that $\boldsymbol{Q}_i = \boldsymbol{R}_i = \boldsymbol{I}_2$ and $\boldsymbol{P}_i$ is the solution of the algebraic Riccati equation for all $i \in \overline{1, N}$. These tuning parameters are used for the deploying agents of Σ driven with the MPC (3.12). Meanwhile, for the incoming agents driven with the MPC (5.4), $\mathcal{U}_{\mathrm{E}} = \mathcal{U}$, $\boldsymbol{Q}^{\mathrm{E}}_i = \boldsymbol{R}^{\mathrm{E}}_i = \boldsymbol{I}_2$ and $\boldsymbol{P}^{\mathrm{E}}_i$ is the solution of the algebraic Riccati equation for all $i \in \overline{11, 13}$. Since the state vector of the incoming agents is exactly the position of these agents, as discussed in Paragraph 5.1.2.2, $\mathcal{X}_{\mathrm{E}} = \mathbb{R}^2$ and (5.4c) can be dropped. The solvers for optimization problems (3.12) and (5.4) are generated with CVXGEN (Mattingley and Boyd, 2012, 2013).

Then, Figure 5.2 presents the trajectory of the agents of Σ as well as the trajectory of the three incoming agents that are integrated to Σ when they enter the workspace $\mathcal{Y}$. These trajectories are presented in the final Voronoi configuration at $k = 150$. The trajectories of the agents are displayed as solid lines while the trajectory of the Chebyshev center of their Voronoi cell is presented as a dashed line. The initial position of the 10 agents of Σ and of the three incoming agents are represented by circles. The 10 agents of Σ begin to deploy by following the Chebyshev center of their cell, the initial centers being represented with stars. Each incoming agent follows its objective point $\mathbf{y}^{\mathrm{E}}_i(k)$, with $i \in \overline{11, 13}$, the initial objectives $\mathbf{y}^{\mathrm{E}}_i(0)$ being represented with diamonds. When an incoming agent reaches $\mathcal{Y}$, it joins the MAS Σ. The incoming agents integrate Σ at different time instants. Then, with Assumption 2.5, all 13 agents of Σ are able to compute their Voronoi cell and the associated Chebyshev center. The initial Chebyshev centers of the incoming agents 11, 12 and 13 are represented with a triangle. Then, these agents naturally participate in the deployment of Σ over $\mathcal{Y}$ until the MAS reaches a static Chebyshev configuration.

In Figure 5.2, it can be seen that the three agents 11, 12 and 13 can join Σ with a minimal impact on the deployment of the agents already present in $\mathcal{Y}$. The only impact can be seen on the position of the Chebyshev center of the agents close to the point of entry of agents 11, 12 and 13, i.e. agents 3 and 5.



○ Agent's initial position $\mathbf{y}_i(0)$ ◇ Incoming agent's initial objective $\mathbf{y}_i^{\mathrm{E}}(0)$
⋆ Initial Chebyshev center $\mathbf{c}_i(0)$ △ Incoming agent's initial Chebyshev center $\mathbf{c}_i(k)$
□ Agent's final position $\mathbf{y}_i(150)$ + Final Chebyshev center $\mathbf{c}_i(150)$

Figure 5.2: Trajectories of the agents of Σ, the agents joining Σ and their associated objectives.

This behavior is highlighted by the distances presented in Figure 5.3. The distances are defined as:

$$d_i(k) = \|\mathbf{y}_i(k) - \mathbf{c}_i(k)\|_2$$

for all $i \in \overline{1, N}$ and:

$$d_i(k) = \begin{cases} \left\|\mathbf{y}_i(k) - \mathbf{y}_i^{\mathrm{E}}(k)\right\|_2 & \text{if } \mathbf{y}_i(k) \in \mathbb{R}^2 \setminus \mathcal{Y} \\ \|\mathbf{y}_i(k) - \mathbf{c}_i(k)\|_2 & \text{otherwise} \end{cases}$$

for all $i \in \overline{11, 13}$. The graph presents pikes for the distances d_{11}, d_{12} and d_{13} at $t = 3\,\mathrm{s}$, $t = 4.8\,\mathrm{s}$ and $t = 3.6\,\mathrm{s}$, respectively. These pikes are due to the entrance of agents 11, 12 and 13 inside $\mathcal{Y}$ and the fact that they suddenly change their objective from $\mathbf{y}_i^{\mathrm{E}}(k)$, with $i \in \overline{11, 13}$, to $\mathbf{c}_i(k)$. These agents then participate in the deployment of Σ and each agent ultimately reaches its Chebyshev center. On the graph Figure 5.3,

a pike also appear on $d_3(k)$ at $t = 6\,\mathrm{s}$ following a sudden change in the position of the Chebyshev center $\mathbf{c}_3(k)$ due to the overall movement of the MAS Σ.

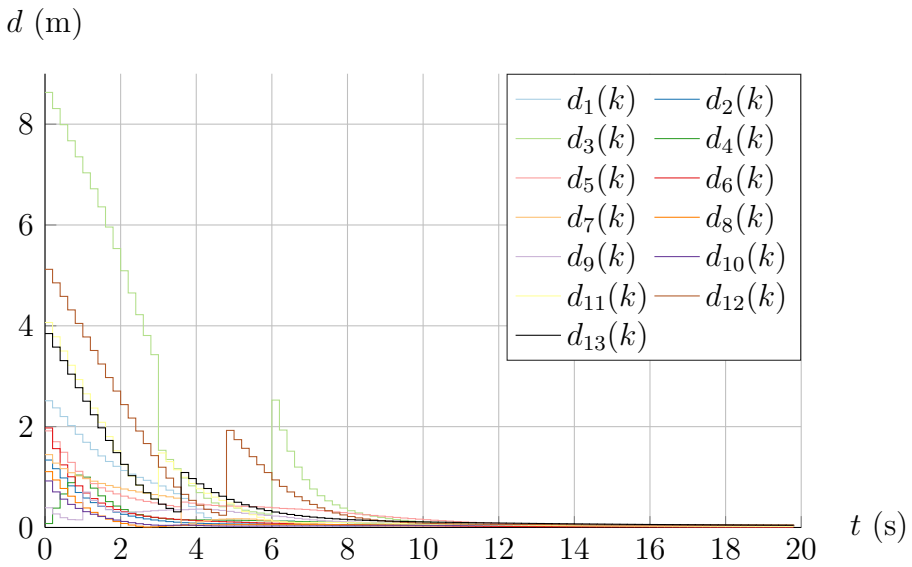


Figure 5.3: Distance of each agent of Σ to its Chebyshev center over time.

5.1.3.2 Outgoing agents

Let Σ be a multi-agent system composed of $N = 10$ agents deployed in the workspace:

$$\mathcal{Y} = \left\{ \mathbf{x} \in \mathbb{R}^2 \;\middle|\; \begin{bmatrix} 3 & 2 \\ 0 & 1 \\ -1 & 2 \\ -1 & -1 \\ 2 & -1 \end{bmatrix} \mathbf{x} \leq \begin{bmatrix} 24 \\ 6 \\ 9 \\ 15 \\ 12 \end{bmatrix} \right\}.$$

All the agents start from random positions $\mathbf{y}_i(0) \in \mathcal{Y}$, with $i \in \overline{1, N}$. All the agents obey the single integrator dynamics (3.15) and the reconfiguration algorithm in the case of one outgoing agent is tested only with agents obeying this dynamics since, as for the nominal deployment case of Chapter 3, the results are quite similar in the UAV dynamics case.

The objective of the agents of Σ is to deploy into a Chebyshev configuration as in Chapter 3. However, at time $t = 5\,\mathrm{s}$ agent 7 has to leave the workspace $\mathcal{Y}$. Its objective is to reach:

$$\mathbf{y}_7^{\mathrm{O}} = \begin{bmatrix} 10 \\ 10 \end{bmatrix}.$$

Then, agent 7 follows Algorithm 5.2 to get out of $\mathcal{Y}$ while the remaining agents of Σ track their neighbors' barycenter as per Algorithm 5.4. When agent 7 is outside $\mathcal{Y}$, the agents of Σ resume their deployment by tracking the Chebyshev center of their Voronoi cell.

The sampling period used is $T_s = 0.2\,\mathrm{s}$ and the contraction factor for the terminal constraint is $\lambda_i = 0.9$ for all $i \in \overline{1, N}$, with a prediction horizon $N_p = 10$. The input set is $\mathcal{U} = \mathcal{U}_{\mathrm{O}} = \mathbb{B}^2(2 \cdot \mathbf{1}_{2\times 1})$ and the weighting matrices for the MPCs (3.12),

(5.8) and (5.13) are $\boldsymbol{Q}_i = \boldsymbol{R}_i = \boldsymbol{I}_2$ and $\boldsymbol{P}_i$ is the solution of the algebraic Riccati equation for all $i \in \overline{1, N}$. Since the state vector of the outgoing agent is exactly the position of this agent, as discussed in Paragraph 5.1.2.3, $\mathcal{X}_\mathrm{O} = \mathbb{R}^2$, i.e. the constraint (5.8c) can be dropped when solving the MPC (5.8) problem. For the computation of the neighbors' barycenter, the value of κ appearing in Algorithm 5.3 is $\kappa = 3$. The solvers for optimization problems (3.12), (5.8) and (5.13) are generated with CVXGEN (Mattingley and Boyd, 2012, 2013).

Figure 5.4 presents the Voronoi tessellation at time $t = 5\,\mathrm{s}$, as well as the trajectories of all the agents $i \in \overline{1, N}$ as solid lines and the trajectories of their Chebyshev centers $\mathbf{c}_i(k)$ as dashed lines from $t = 0\,\mathrm{s}$ to $t = 5\,\mathrm{s}$. At $t = 5\,\mathrm{s}$, agent 7 leaves Σ and the workspace $\mathcal{Y}$. The behavior is identical to what can be observed in Section 3.2.4 and is not further described here. The only difference with the nominal case is that some agents (e.g. agent 8) are not able to reach their Chebyshev center before agent 7 starts leaving the workspace, which is not a problem since the reconfiguration can occur at any point during the deployment of Σ.

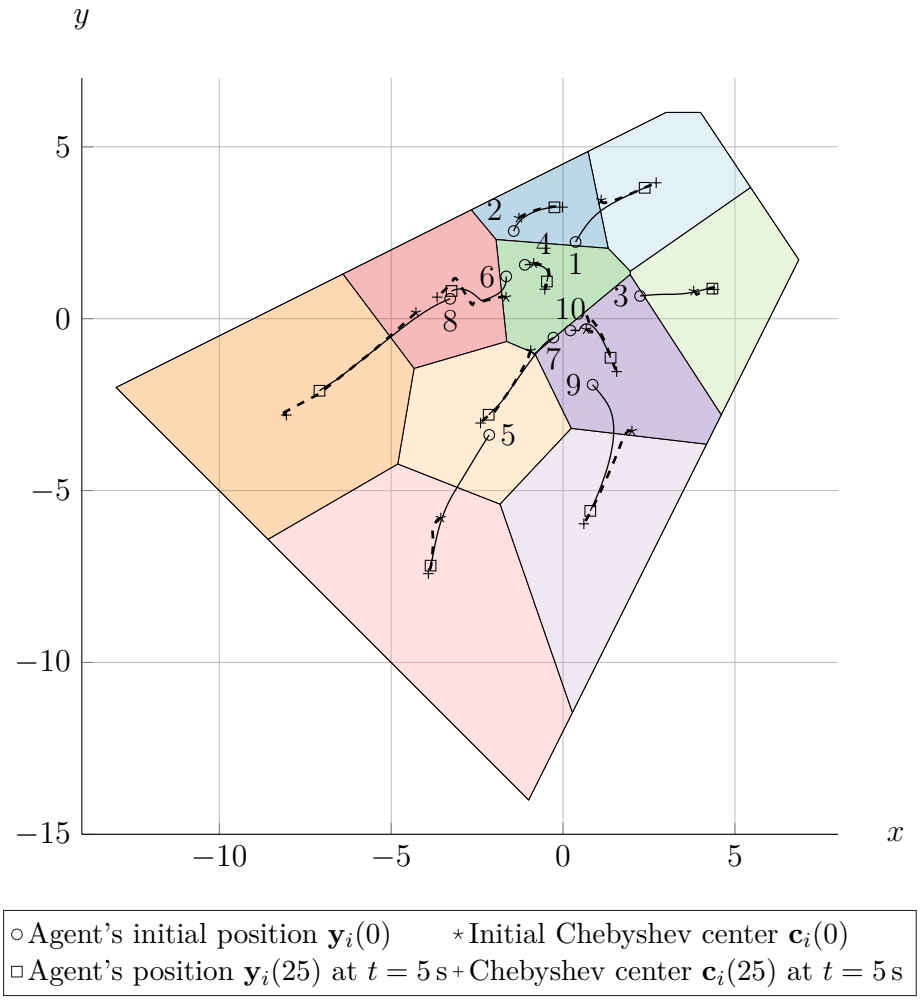


Figure 5.4: Trajectories of the agents of Σ during the first phase of the deployment.

Figure 5.5 presents the trajectories of all the agents $i \in \overline{1, N}$ as solid lines while agent 7 is leaving $\mathcal{Y}$ from $t = 5.2\,\mathrm{s}$ (i.e. $k = 26$) to $t = 8.8\,\mathrm{s}$ (i.e. $k = 44$), i.e. during the reconfiguration phase. It presents the trajectories of the neighbors' barycenters $\mathbf{y}_i^\mathrm{b}(k)$ for all $i \in \overline{1, N} \setminus \{7\}$ as dashed lines. In this figure, the Voronoi tessellation is

not presented for readability reasons. It can be seen in Figure 5.5 that the positions of the neighbors' barycenters vary greatly at each time instant in a reduced time frame which is still acceptable since this time frame is short. The agents are globally able to track their neighbors' barycenter while agent 7 leaves the workspace. The objective of the outgoing agent 7 is not presented in the figure for readability reasons.

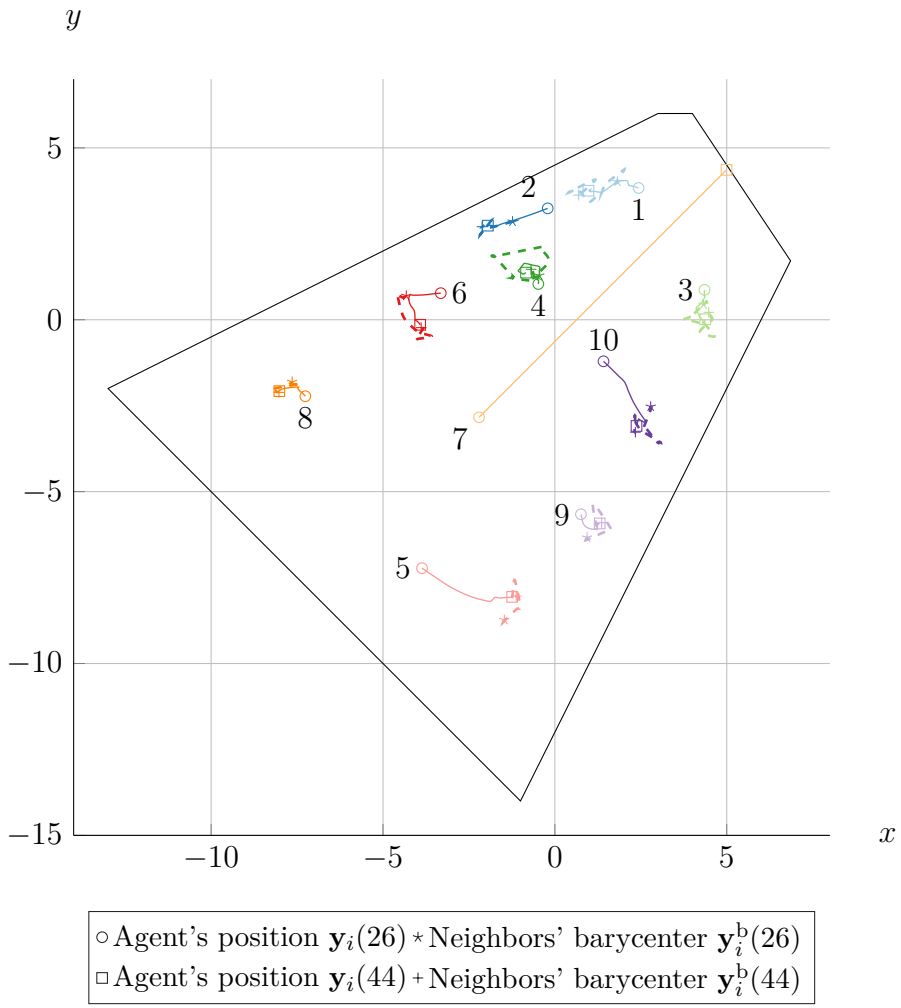


Figure 5.5: Trajectories of the agents of Σ and agent 7 during the reconfiguration phase of the deployment.

Finally, Figure 5.6 is similar to Figure 5.4 since it presents the same elements for the agents of Σ when agent 7 is out of $\mathcal{Y}$. The trajectories are presented from $t = 9\,\mathrm{s}$ (i.e. $k = 45$) to $t = 40\,\mathrm{s}$ (i.e. $k = 200$) inside the Voronoi tessellation at $t = 40\,\mathrm{s}$.

As for the other cases presented in this thesis, it is interesting to look at the distances between the agents and their objectives. This distance is defined as:

$$d_i(k) = \begin{cases} \left\|\mathbf{y}_i(k) - \mathbf{y}_i^{\mathrm{b}}(k)\right\|_2 & \text{if } \mathbf{y}_7(k) \in \mathcal{Y} \\ \left\|\mathbf{y}_i(k) - \mathbf{c}_i(k)\right\|_2 & \text{otherwise} \end{cases}$$

for all $i \in \overline{1, N} \setminus \{7\}$ and:

$$d_7(k) = \begin{cases} \left\|\mathbf{y}_7(k) - \mathbf{c}_7(k)\right\|_2 & \text{if } k < 25 \\ \left\|\mathbf{y}_7(k) - \mathbf{y}_7^{\mathrm{O}}(k)\right\|_2 & \text{otherwise.} \end{cases}$$

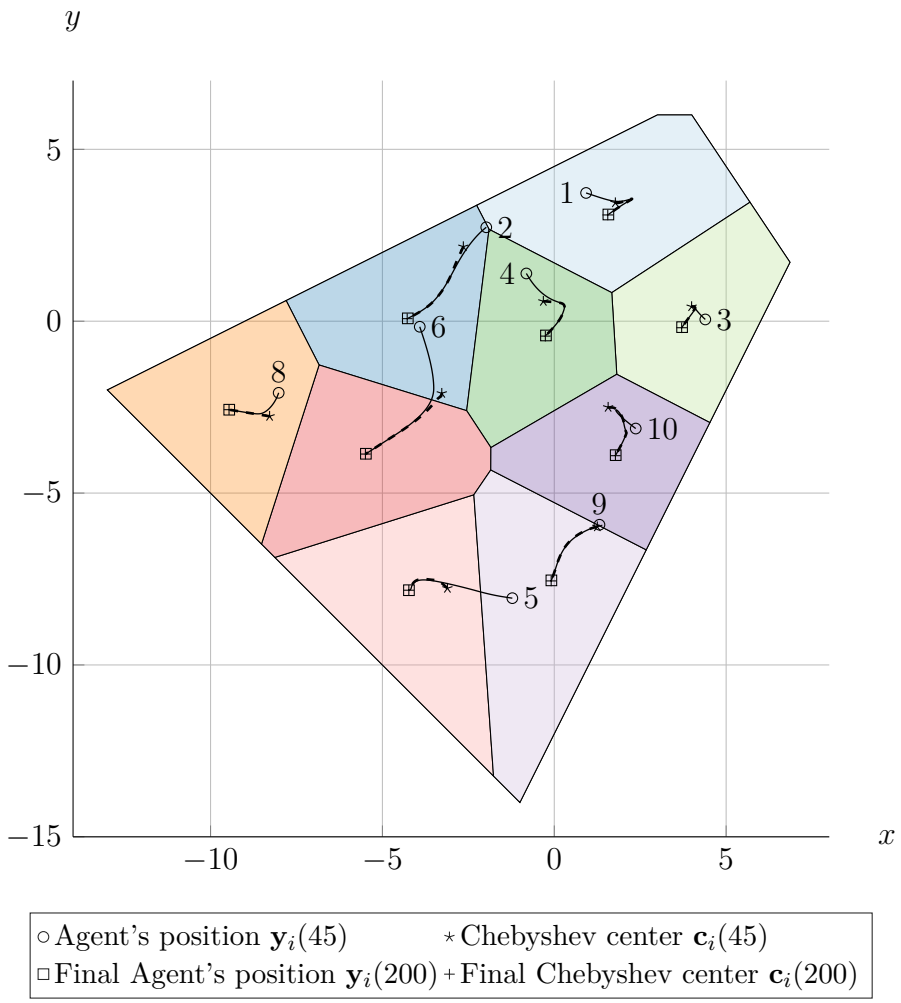


Figure 5.6: Trajectories of the agents of Σ during the third phase of the deployment.

Figure 5.7 presents the distances $d_i(k)$ for all $i \in \overline{1, N} \setminus \{7\}$. The behavior previously observed appears clearly on the value of the distances. Indeed, at $t = 5.2\,\mathrm{s}$, an important change on the distance to the objective point appears since each agent starts tracking its neighbors' barycenter instead of the Chebyshev center of its Voronoi cell. Then, at $t = 8.8\,\mathrm{s}$, a new important change occurs since the agents go back to the deployment objective instead of tracking their neighbors' barycenter. The agents then converge towards their Chebyshev center. During the time frame where agent 7 leaves the workspace $\mathcal{Y}$, it is obvious from Figure 5.7 and Figure 5.5 that the movement of the entire multi-agent system is not negligible. This is something that the algorithm provided in Section 5.2.2 aims to improve. Finally, Figure 5.8 presents the distance $d_7(k)$ which exhibits the fact that it is able to rally its objective point outside the workspace.

5.2 A safer way to deal with outgoing vehicles

5.2.1 Limitation of the first reconfiguration algorithm

In the following, Σ is the multi-agent system described in Paragraph 5.1.2.1. While Algorithm 5.4 and Algorithm 5.2 are effective to deal with the case of one agent

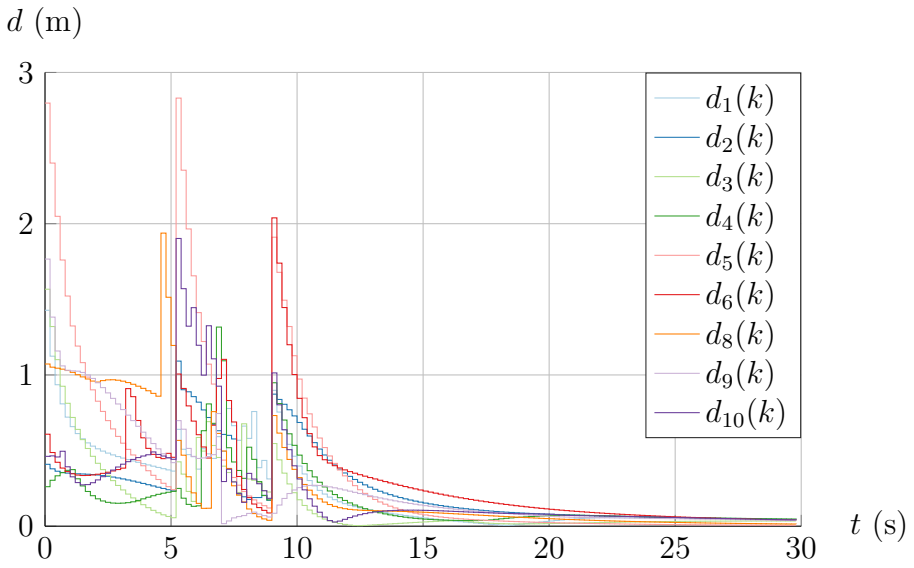


Figure 5.7: Distance of each agent of Σ to its Chebyshev center or neighbors' barycenter over time.

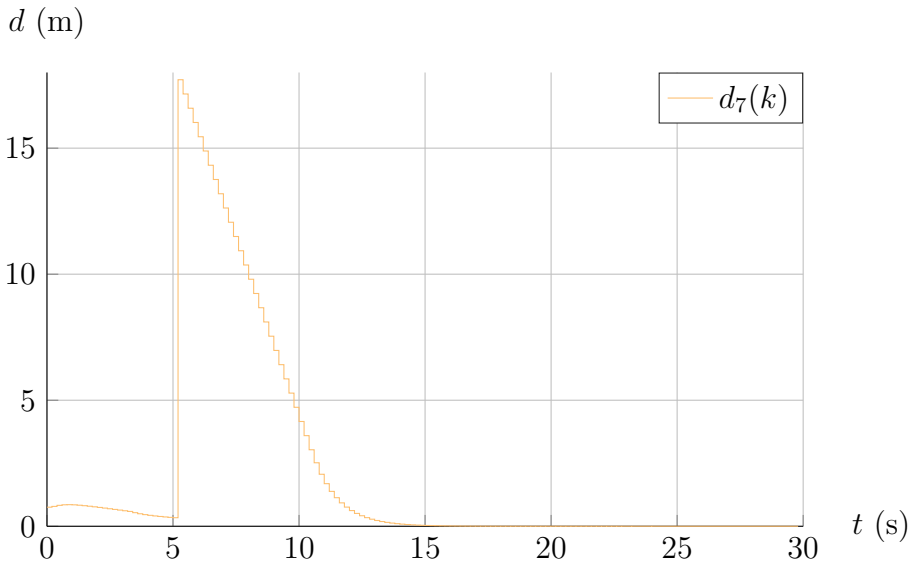


Figure 5.8: Distance of agent 7 to its objective over time.

leaving the MAS Σ, Chevet et al. (2020b) highlights a limitation that can arise when several agents leave Σ at the same time.

Indeed, the transient objective proposed in Paragraph 5.1.2.3 for an agent i of Σ, i.e. the neighbors' barycenter $\mathbf{y}_i^{\mathrm{b}}$, is relatively simple. The barycenter $\mathbf{y}_i^{\mathrm{b}}$ is computed only from the position of the neighbors of agent i which guarantees, by construction, that when one agent leaves the workspace, agent i is driven away from it. However, when two agents leave the workspace at the same time, there is no guarantee that the neighbors' barycenter does not end up lying on (or really close to) the trajectory of one of the outgoing agent.

Let $o \in \mathrm{O}(k) \subset \overline{1, N}$ index the agents leaving Σ at time instant k. In this case, where several agents leave the workspace, Algorithm 5.3 from the previous

section is naturally modified into Algorithm 5.5 described below. Let $\partial\mathcal{H}_o(k)$, with $o \in \mathrm{O}(k)$, be the hyperplane described by the position $\mathbf{y}_o(k)$ of the outgoing agent o and its target point $\mathbf{y}_o^{\mathrm{O}}$. For an agent $i \in \overline{1,N} \setminus \mathrm{O}(k)$, if the neighbor $\nu \in \mathrm{N}_i(k)$ is on the same side of all the hyperplanes $\partial\mathcal{H}_o(k)$ as agent i and farther from all these hyperplanes than agent i, then it is assigned the weight $\omega_{\nu,i}(k) = \kappa$, otherwise $\omega_{\nu,i}(k) = 1$. A limitation related to this objective point arises when at least two agents leave Σ. Indeed, if a remaining agent is located between the trajectories of these two outgoing agents, the neighbors' barycenter might not guarantee anymore that the remaining agent is driven away from the outgoing agents, leading to a collision risk. Example 5.2 then illustrates the limitation of the barycentric approach.

Algorithm 5.5: Computation of the neighbors' barycenter of a remaining agent of Σ when several agents leave the workspace.

Input: The list of neighbors $\mathrm{N}_i(k)$ of agent $i \in \overline{1,N} \setminus \mathrm{O}(k)$, the positions of the neighbors $\mathbf{y}_\nu(k)$, $\forall \nu \in \mathrm{N}_i(k)$, the positions of the outgoing agents $\mathbf{y}_o(k)$, $\forall o \in \mathrm{O}(k)$, the hyperplanes $\partial\mathcal{H}_o(k) = \{\mathbf{x} \in \mathbb{R}^2 \mid \boldsymbol{h}_o(k)\mathbf{x} = \theta_o(k)\}$, $\forall o \in \mathrm{O}(k)$

```
for ν ∈ N_i(k) do
    for o ∈ O(k) do
        if h_o(k)(y_ν(k) − y_o(k)) · h_o(k)(y_i(k) − y_o(k)) ≥ 0 then
            if d(y_ν(k), ∂H_o(k)) > d(y_i(k), ∂H_o(k)) then
                ω_{ν,i}(k) ← κ;
            else
                ω_{ν,i}(k) ← 1;
                Go to 1;
            end
        else
            ω_{ν,i}(k) ← 1;
            Go to 1;
        end
    end
end
Compute the neighbors' barycenter of agent i with (5.11);
```

Output: The neighbors' barycenter $\mathbf{y}_i^{\mathrm{b}}(k)$ of agent i

Example 5.2: Limit case of the reconfiguration algorithm in the case of several outgoing agents

The conditions are exactly the same as for Example 5.1. However, now, at a given time k, agents 2 and 3 leave Σ and the workspace $\mathcal{Y}$ as illustrated in Figure 5.9. They have for objectives $\mathbf{y}_2^{\mathrm{O}} = \begin{bmatrix}-12 & -6.2\end{bmatrix}^\top \in \mathbb{R}^2 \setminus \mathcal{Y}$ and $\mathbf{y}_3^{\mathrm{O}} = \begin{bmatrix}-9 & -12\end{bmatrix}^\top \in \mathbb{R}^2 \setminus \mathcal{Y}$ which define the hyperplanes $\partial\mathcal{H}_2(k)$ and $\partial\mathcal{H}_3(k)$, with $\mathbf{y}_2(k)$ and $\mathbf{y}_3(k)$, respectively.

Since there is no difference in the configuration with respect to Example 5.1, the neighbor set of agent 5 is still $\mathrm{N}_5(k) = \{2,3,6,9\}$. However, here, $\partial\mathcal{H}_2(k)$ and $\partial\mathcal{H}_3(k)$ are parallel. It is thus impossible for any neighbor of agent 5 to be farther from both $\partial\mathcal{H}_2(k)$ and $\partial\mathcal{H}_3(k)$ than agent 5. Then, from Algorithm 5.5, all the neighbors of agent 5 receive the weight $\omega_{\nu,5}(k) = 1$ for all $\nu \in \mathrm{N}_5(k)$.

The neighbors' barycenter $\mathbf{y}_5^{\mathrm{b}}(k)$ of agent 5 is then presented in Figure 5.9 as a

square. This objective is directly on the trajectory of agent 3, which could lead to a collision between agents 3 and 5 during the movement.

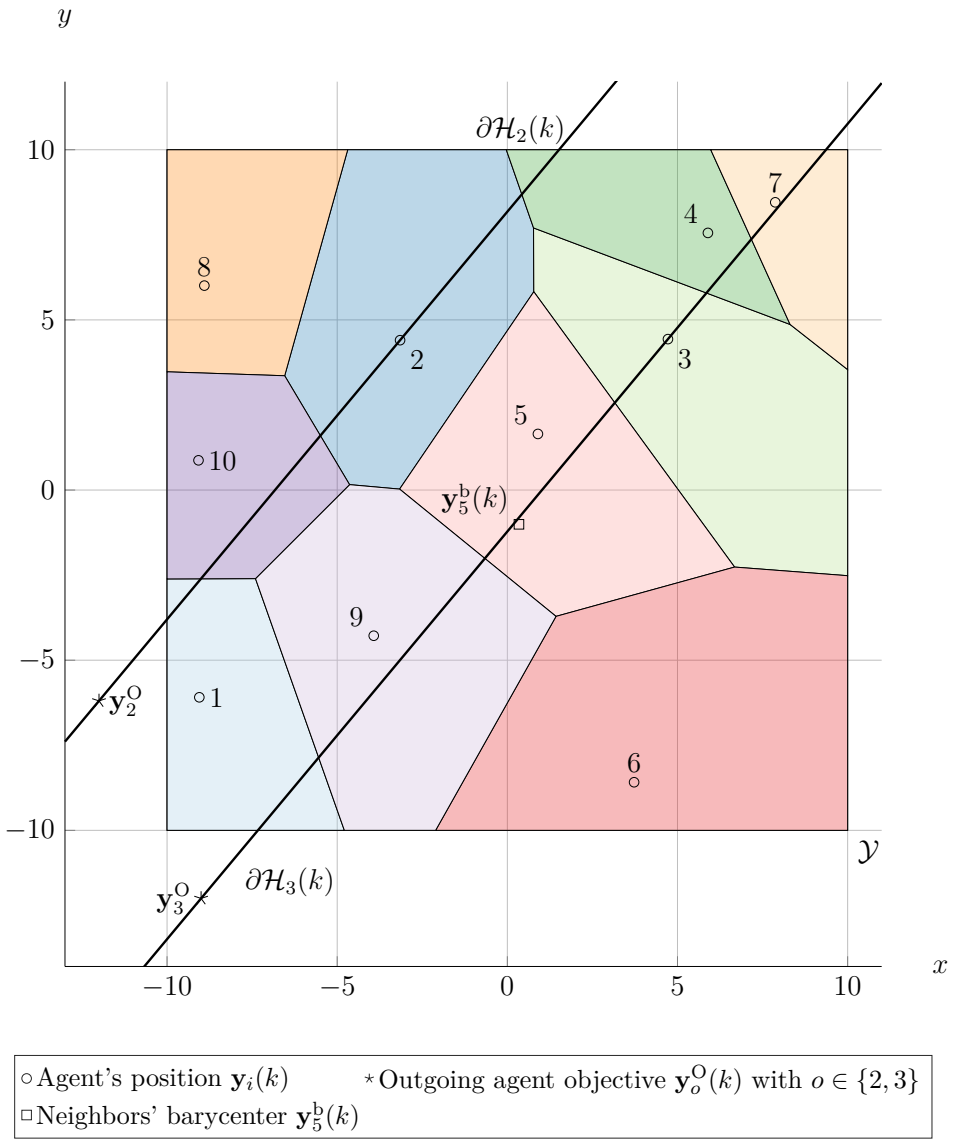


Figure 5.9: Limit case of the barycentric approach for the MAS reconfiguration when several agents leave the workspace.

Then, while the strategy of Paragraph 5.1.2.3 works well in the case of one outgoing agent, it is limited in the case of several agents leaving the MAS. Section 5.2.2 introduces a new transient objective to drive the agents of Σ safely away from all the outgoing agents as well as an improved algorithm to limit the overall movement of the entire MAS.

5.2.2 Improved reconfiguration algorithm

5.2.2.1 A new transient objective

Using the notations of the previous paragraph, the neighbors' barycenter of agent i of Σ, with $i \in \overline{1, N} \setminus \mathrm{O}(k)$, satisfies:

$$\sum_{\nu \in \mathrm{N}_i(k)} \omega_{\nu,i}(k)\big(\mathbf{y}_i^{\mathrm{b}}(k) - \mathbf{y}_\nu(k)\big) = \mathbf{0}_{2\times 1}.$$

Based on this relation, a new safe objective $\mathbf{y}_i^{\mathrm{s}}(k)$ is designed. Indeed, the neighbors' barycenter is the unconstrained minimum:

$$\mathbf{y}_i^{\mathrm{b}}(k) = \underset{\mathbf{y}\in\mathbb{R}^2}{\arg\min} \left\| \sum_{\nu \in \mathrm{N}_i(k)} \omega_{\nu,i}(k)(\mathbf{y} - \mathbf{y}_\nu(k)) \right\|_2^2. \tag{5.16}$$

Due to the lack of constraints in the previous minimization problem, there is no guarantee, as illustrated in Section 5.2.1, that the objective of the agents remaining inside the workspace $\mathcal{Y}$ allow them to be driven to safety and to avoid collision with the outgoing agents. Then, the new objective $\mathbf{y}_i^{\mathrm{s}}(k)$ is the solution of:

$$\begin{aligned} \mathbf{y}_i^{\mathrm{s}}(k) = \underset{\substack{\mathbf{y} \in \mathbb{R}^2 \\ \text{subject to} \\ \mathbf{y} \in \mathcal{R}_i(k)}}{\arg\min} \quad & \sum_{\nu \in \mathrm{N}_i(k)} \omega_{\nu,i}(k)^2 \|\mathbf{y} - \mathbf{y}_\nu(k)\|_2^2 \end{aligned} \tag{5.17}$$

where $\mathcal{R}_i(k)$ is a region where it is safe for agent i to evolve. In problem (5.17), the cost function is equivalent to the cost function of (5.16) since by the triangular inequality:

$$\left\| \sum_{\nu \in \mathrm{N}_i(k)} \omega_{\nu,i}(k)(\mathbf{y} - \mathbf{y}_\nu(k)) \right\|_2^2 \leq \left(\sum_{\nu \in \mathrm{N}_i(k)} \omega_{\nu,i}(k) \|\mathbf{y} - \mathbf{y}_\nu(k)\|_2 \right)^2$$

and by the Cauchy-Schwarz inequality:

$$\left(\sum_{\nu \in \mathrm{N}_i(k)} \omega_{\nu,i}(k) \|\mathbf{y} - \mathbf{y}_\nu(k)\|_2 \right)^2 \leq |\mathrm{N}_i(k)| \sum_{\nu \in \mathrm{N}_i(k)} \omega_{\nu,i}(k)^2 \|\mathbf{y} - \mathbf{y}_\nu(k)\|_2^2.$$

In addition, $|\mathrm{N}_i(k)|$ acting as a scaling factor, it can be omitted in the minimization problem (5.17).

The *region* $\mathcal{R}_i(k)$ is defined as the intersection of two sets:

- the *contracted Voronoi cell* $\mathcal{V}_i(k, \lambda_{\mathcal{V}_i})$ of agent i defined as:

  $$\mathcal{V}_i(k, \lambda_{\mathcal{V}_i}) = \mathbf{c}_i(k) \oplus \lambda_{\mathcal{V}_i}(\mathcal{V}_i(k) \oplus \{-\mathbf{c}_i(k)\})$$

  where $\mathbf{c}_i(k)$ is the Chebyshev center of the Voronoi cell $\mathcal{V}_i(k)$ and $\lambda_{\mathcal{V}_i} \in [0, 1)$ is a scaling factor;

- the *contracted working region* $\mathcal{W}_i(k, \lambda_{\mathcal{W}_i})$ defined from the hyperplanes $\partial\mathcal{H}_o(k)$, with $o \in \mathrm{O}(k)$, where $\lambda_{\mathcal{W}_i} \in [0, 1)$ is a scaling factor.

The contracted working region has been introduced in Chevet et al. (2020b). Let $\mathbf{y}_o(k)$, with $o \in \mathrm{O}(k)$, be the positions of the outgoing agents and $\mathbf{y}_o^{\mathrm{O}}$ their objectives in $\mathbb{R}^2 \backslash \mathcal{Y}$. These vectors define $|\mathrm{O}(k)|$ hyperplanes $\partial\mathcal{H}_o(k) = \{\mathbf{x} \in \mathbb{R}^2 \mid \boldsymbol{h}_o(k)\mathbf{x} = \theta_o(k)\}$ in $\mathbb{R}^2$. The working region of agent $i \in \overline{1, N} \setminus \mathrm{O}(k)$ is then defined as:

$$\mathcal{W}_i(k) = \left\{\mathbf{y} \in \mathbb{R}^2 \;\middle|\; \boldsymbol{h}_o(k)(\mathbf{y}_i(k) - \mathbf{y}_o(k)) \cdot \boldsymbol{h}_o(k)(\mathbf{y} - \mathbf{y}_o(k)) \geq 0,\ \forall o \in \mathrm{O}(k)\right\} \tag{5.18}$$

or as the set of all points on the same side of all the hyperplanes $\partial\mathcal{H}_o(k)$ as $\mathbf{y}_i(k)$. The definition of $\mathcal{W}_i(k)$ of (5.18) is that of a polyhedron. Then, in the following, the working region of agent i is represented equivalently as:

$$\mathcal{W}_i(k) = \left\{\mathbf{x} \in \mathbb{R}^2 \;\middle|\; \boldsymbol{H}_{\mathcal{W}_i}(k)\mathbf{x} \leq \boldsymbol{\theta}_{\mathcal{W}_i}(k)\right\} \tag{5.19}$$

with $\boldsymbol{H}_{\mathcal{W}_i}(k) \in \mathbb{R}^{s_i(k)\times 2}$ and $\boldsymbol{\theta}_{\mathcal{W}_i}(k) \in \mathbb{R}^{s_i(k)}$, where $s_i(k) \in \overline{1, |\mathrm{O}(k)|}$. The working region as defined in (5.19) is convex but can be unbounded.

Let $\mathbf{c}_{\mathcal{W}_i}(k)$ be the Chebyshev center of the intersection $\mathcal{W}_i(k) \cap \mathcal{Y}$ of the working region $\mathcal{W}_i(k)$ and the workspace $\mathcal{Y}$. The intersection $\mathcal{W}_i(k) \cap \mathcal{Y}$ is necessary to ensure that the Chebyshev center $\mathbf{c}_{\mathcal{W}_i}(k)$ exists, since $\mathcal{W}_i(k)$ is generally unbounded. Then, the contracted working region is defined as:

$$\mathcal{W}_i(k, \lambda_{\mathcal{W}_i}) = \mathbf{c}_{\mathcal{W}_i}(k) \oplus \lambda_{\mathcal{W}_i}(\mathcal{W}_i(k) \oplus \{-\mathbf{c}_{\mathcal{W}_i}(k)\}). \tag{5.20}$$

It can be noticed that the Chebyshev center $\mathbf{c}_{\mathcal{W}_i}(k)$ is used only for the definition of $\mathcal{W}_i(k, \lambda_{\mathcal{W}_i})$ and is not used *per se* in the following control algorithms.

Remark 5.3: Shared working region
Given the definition of the working region $\mathcal{W}_i(k)$ of agent $i \in \overline{1, N} \setminus \mathrm{O}(k)$, several agents can share the same working region (and as such, the same contracted working region) such that $\mathcal{W}_i(k) = \mathcal{W}_j(k)$ for two agents $i, j \in \overline{1, N} \setminus \mathrm{O}(k)$. ◊

Example 5.3: Construction of a contracted working region
Let Σ be the MAS of Example 5.2. Agents 2 and 3 leave the workspace $\mathcal{Y}$ towards the objectives $\mathbf{y}_2^{\mathrm{O}} = \begin{bmatrix}-12 & -8\end{bmatrix}^\top$ and $\mathbf{y}_3^{\mathrm{O}} = \begin{bmatrix}-9 & -12\end{bmatrix}^\top$ depicted by squares in Figure 5.10. At a given time k, the positions $\mathbf{y}_2(k)$ and $\mathbf{y}_3(k)$ of agents 2 and 3 combined with their objectives define two hyperplanes $\partial\mathcal{H}_2(k)$ and $\partial\mathcal{H}_3(k)$ represented as solid lines in Figure 5.10.

By expression (5.18), the working regions of agents 1, 4, 5, 7 and 9 are contained in-between $\partial\mathcal{H}_2(k)$ and $\partial\mathcal{H}_3(k)$ and, as per Remark 5.3, $\mathcal{W}_1(k) = \mathcal{W}_4(k) = \mathcal{W}_5(k) = \mathcal{W}_7(k) = \mathcal{W}_9(k)$. The working region $\mathcal{W}_1(k)$, i.e. the area between the two hyperplanes $\partial\mathcal{H}_2(k)$ and $\partial\mathcal{H}_3(k)$, is displayed in light blue in Figure 5.10. Moreover, the working region $\mathcal{W}_6(k)$ of agent 6 is the half-space below $\partial\mathcal{H}_3(k)$, which is displayed in light red in Figure 5.10. The working regions of agents 8 and 10 are the half-space above $\partial\mathcal{H}_2(k)$ such that $\mathcal{W}_8(k) = \mathcal{W}_{10}(k)$, which is displayed in light green in Figure 5.10.

The Chebyshev centers $\mathbf{c}_{\mathcal{W}_i}(k)$, with $i \in \overline{1, 10} \setminus \{2, 3\}$, of the intersection of the working regions $\mathcal{W}_i(k)$ with the workspace $\mathcal{Y}$ are displayed with stars in Figure 5.10. Then, choosing $\lambda_{\mathcal{W}_i} = 0.8$ for all $i \in \{1, 4, 5, 7, 9\}$, $\lambda_{\mathcal{W}_6} = 0.5$ and $\lambda_{\mathcal{W}_i} = 0.7$ for all $i \in \{8, 10\}$, the contracted working regions $\mathcal{W}_i(k, \lambda_{\mathcal{W}_i})$ for all $i \in \{1, 4, 5, 7, 9\}$ are displayed in dark blue, the region $\mathcal{W}_6(k, \lambda_{\mathcal{W}_6})$ is displayed in dark red and the regions $\mathcal{W}_i(k, \lambda_{\mathcal{W}_i})$ for all $i \in \{8, 10\}$ are displayed in dark green.

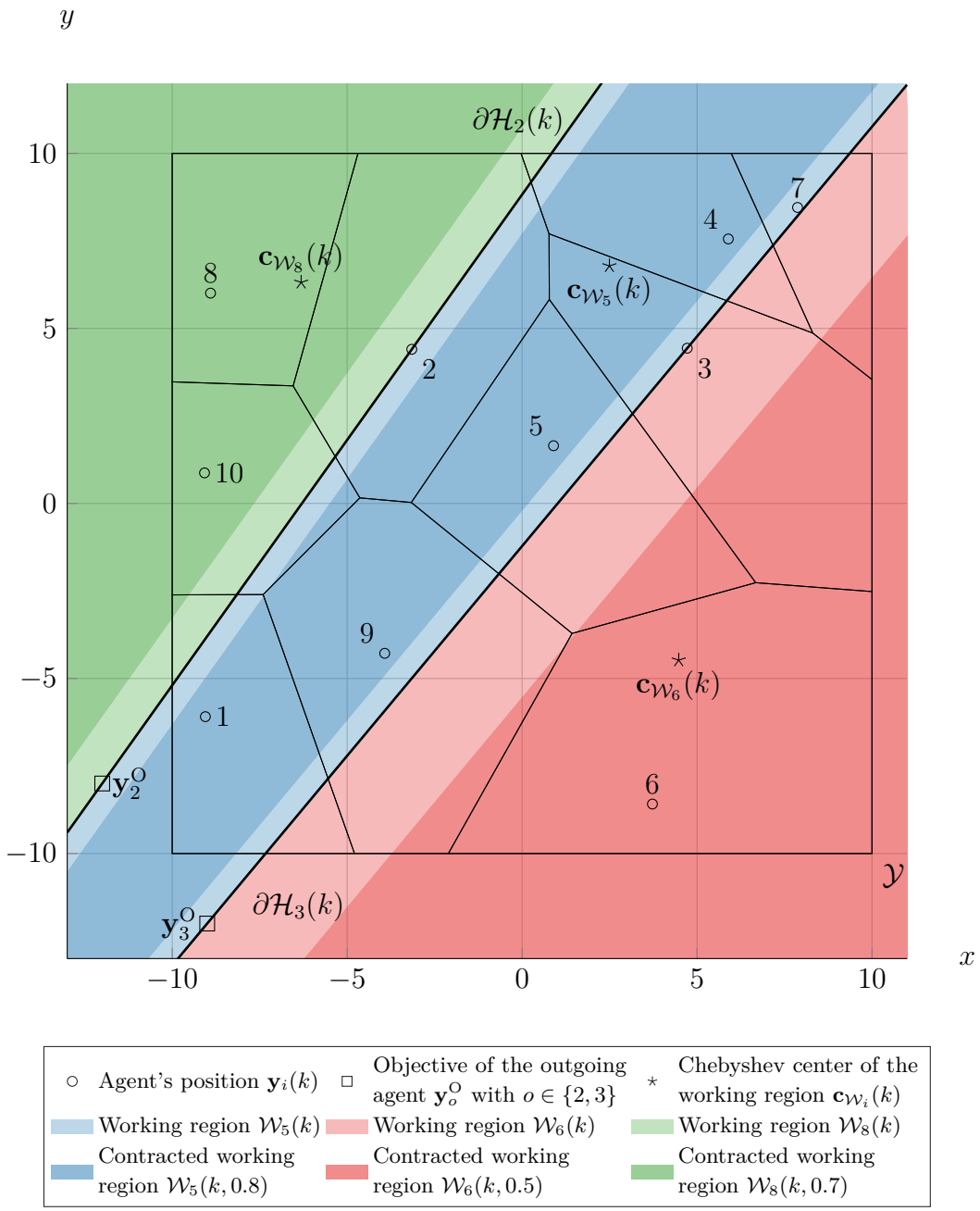


Figure 5.10: Construction of the contracted working regions.

With these definitions, the safe objective $\mathbf{y}_i^{\mathrm{s}}(k)$ is then the solution of (5.17) such that $\mathbf{y}_i^{\mathrm{s}}(k) \in \mathcal{V}_i(k, \lambda_{\mathcal{V}_i}) \cap \mathcal{W}_i(k, \lambda_{\mathcal{W}_i})$. Then, due to the fact that $\mathbf{y}_i^{\mathrm{s}}(k) \in \mathcal{W}_i(k, \lambda_{\mathcal{W}_i})$, it is guaranteed that agent i is driven away from the trajectories of all the outgoing agents $o \in \mathrm{O}(k)$.

In the following, the neighbors are defined as in Definition 5.1. As for the case of Paragraph 5.1.2.3, the weights $\omega_{\nu,i}(k)$, with $\nu \in \mathrm{N}_i(k)$, have to be attributed given the relative position of the neighbors of agent $i \in \overline{1, N} \setminus \mathrm{O}(k)$ to agent i and to the outgoing agents $o \in \mathrm{O}(k)$. The way these weights are attributed is close to what is presented in Algorithm 5.5. Let ω_h, ω_l and ω_e be three positive weights such that $\omega_h > \omega_l > \omega_e \geq 0$. The weight ω_h is attributed to so-called "heavy" neighbors, the weight ω_l to "light" neighbors and the weight ω_e to outgoing (or *exiting*) neighbors.

Then if the neighbor $\nu \in \mathrm{N}_i(k)$ is an outgoing agent, the neighbor receives the outgoing weight $\omega_{\nu,i}(k) = \omega_e$. If $\boldsymbol{h}_o(k)(\mathbf{y}_\nu(k) - \mathbf{y}_o(k)) \cdot \boldsymbol{h}_o(k)(\mathbf{y}_i(k) - \mathbf{y}_o(k)) \geq 0$ and $d(\mathbf{y}_\nu(k), \partial\mathcal{H}_o(k)) > d(\mathbf{y}_i(k), \partial\mathcal{H}_o(k))$ for all $o \in \mathrm{O}(k)$, the neighbor receives the heavy weight $\omega_{\nu,i}(k) = \omega_h$. Otherwise the neighbor receives the light weight $\omega_{\nu,i}(k) = \omega_l$. This procedure is summarized in Algorithm 5.6.

Algorithm 5.6: Attribution of the weights for the computation of the safe objective of a remaining agent of the MAS Σ.

Input: The list of neighbors $\mathrm{N}_i(k)$ of agent $i \in \overline{1, N} \setminus \mathrm{O}(k)$, the position of the neighbors $\mathbf{y}_\nu(k)$, $\forall \nu \in \mathrm{N}_i(k)$, the positions of the outgoing agents $\mathbf{y}_o(k)$, $\forall o \in \mathrm{O}(k)$, the hyperplanes $\partial\mathcal{H}_o(k) = \{\mathbf{x} \in \mathbb{R}^2 \mid \boldsymbol{h}_o(k)\mathbf{x} = \theta_o(k)\}$, $\forall o \in \mathrm{O}(k)$

```
for ν ∈ N_i(k) do
    if ν ∈ O(k) then
        ω_{ν,i}(k) ← ω_e;
    else
        for o ∈ O(k) do
            if h_o(k)(y_ν(k) − y_o(k)) · h_o(k)(y_i(k) − y_o(k)) ≥ 0 then
                if d(y_ν(k), ∂H_o(k)) > d(y_i(k), ∂H_o(k)) then
                    ω_{ν,i}(k) ← ω_h;
                else
                    ω_{ν,i}(k) ← ω_l;
                    Go to 1;
                end
            else
                ω_{ν,i}(k) ← ω_l;
                Go to 1;
            end
        end
    end
end
```

Example 5.4: Weight attributions to neighbors

Let Σ be the MAS of Example 5.3. Agents 2 and 3 leave the workspace $\mathcal{Y}$ towards the objectives $\mathbf{y}_2^{\mathrm{O}}$ and $\mathbf{y}_3^{\mathrm{O}}$ depicted by squares in Figure 5.11. At a given time k, the positions $\mathbf{y}_2(k)$ and $\mathbf{y}_3(k)$ of agents 2 and 3 combined with their objectives define two hyperplanes $\partial\mathcal{H}_2(k)$ and $\partial\mathcal{H}_3(k)$ represented as solid lines in Figure 5.11.

In Figure 5.11, the light blue area, defined for agent 5, is the set of all points $\mathbf{y}$ such that:

$$\begin{aligned} \boldsymbol{h}_2(k)(\mathbf{y} - \mathbf{y}_2(k)) \cdot \boldsymbol{h}_2(k)(\mathbf{y}_5(k) - \mathbf{y}_2(k)) \geq 0 \\ d(\mathbf{y}, \partial\mathcal{H}_2(k)) > d(\mathbf{y}_5(k), \partial\mathcal{H}_2(k)) \\ \boldsymbol{h}_3(k)(\mathbf{y} - \mathbf{y}_3(k)) \cdot \boldsymbol{h}_3(k)(\mathbf{y}_5(k) - \mathbf{y}_3(k)) \geq 0 \\ d(\mathbf{y}, \partial\mathcal{H}_3(k)) > d(\mathbf{y}_5(k), \partial\mathcal{H}_3(k)). \end{aligned}$$

Thus, if a neighbor of agent 5 is inside this area, it receives the weight ω_h and is called a "heavy" neighbor. The set of neighbors of agent 5 is $\mathrm{N}_5(k) = \{2, 3, 6, 9\}$.

There are no elements of $\mathrm{N}_5(k)$ *in the area described before thus no neighbors of agent 5 receive the weight* ω_h*. Agents 2 and 3 are outgoing agents and thus receive the weight* $\omega_{2,5}(k) = \omega_{3,5}(k) = \omega_e$*. Finally, agents 6 and 9 receive the weight* $\omega_{6,5}(k) = \omega_{9,5}(k) = \omega_l$ *and are called "light" agents.*

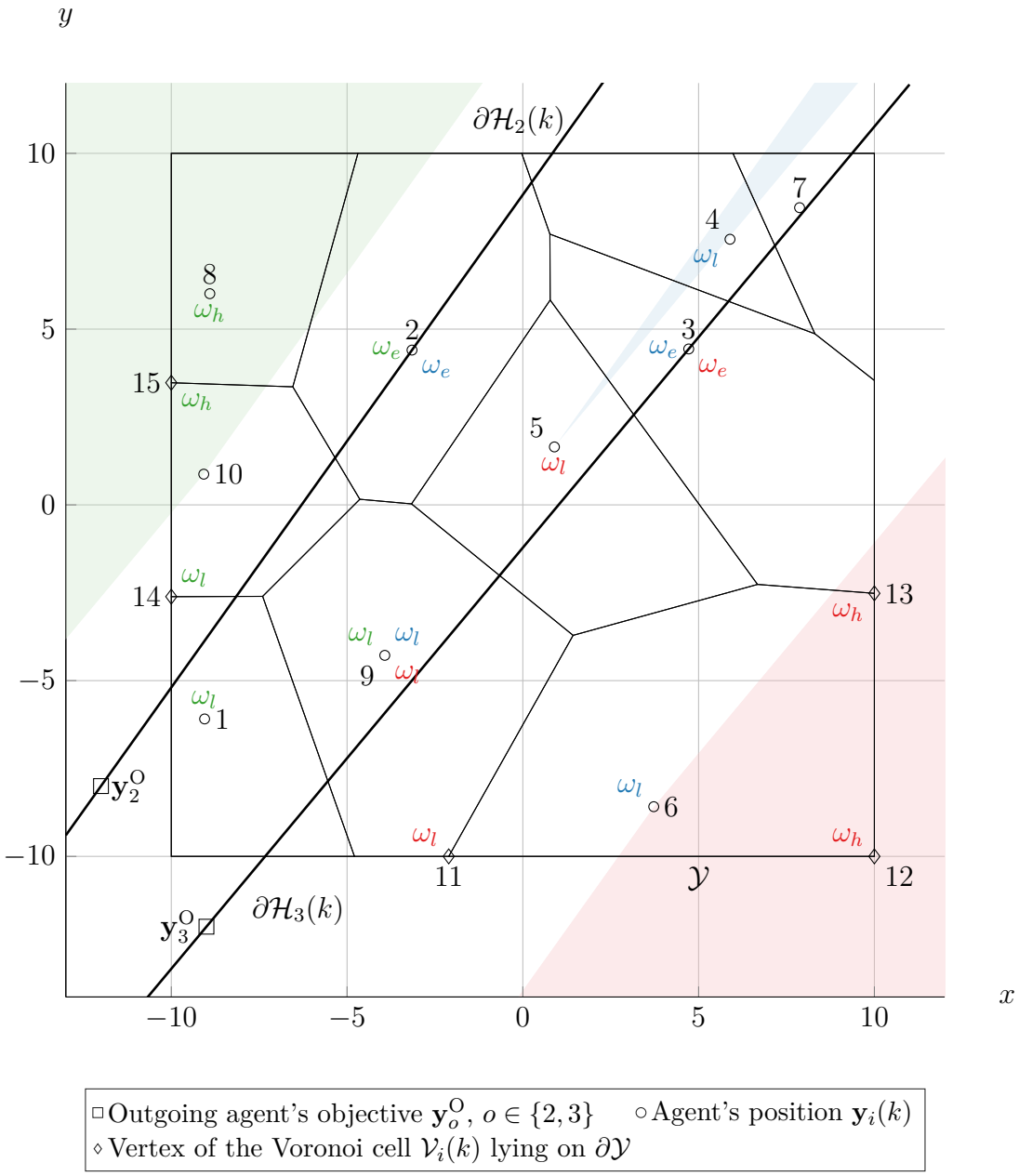


Figure 5.11: Attribution of weights to the neighbors of the agents of the MAS Σ.

The light red area of Figure 5.11, defined for agent 6, is the set of all points $\mathbf{y}$ *such that:*

$$\begin{aligned} \boldsymbol{h}_2(k)(\mathbf{y} - \mathbf{y}_2(k)) \cdot \boldsymbol{h}_2(k)(\mathbf{y}_6(k) - \mathbf{y}_2(k)) \geq 0 \\ d(\mathbf{y}, \partial\mathcal{H}_2(k)) > d(\mathbf{y}_6(k), \partial\mathcal{H}_2(k)) \\ \boldsymbol{h}_3(k)(\mathbf{y} - \mathbf{y}_3(k)) \cdot \boldsymbol{h}_3(k)(\mathbf{y}_6(k) - \mathbf{y}_3(k)) \geq 0 \\ d(\mathbf{y}, \partial\mathcal{H}_3(k)) > d(\mathbf{y}_6(k), \partial\mathcal{H}_3(k)). \end{aligned}$$

Thus, if a neighbor of agent 6 is inside this area, it receives the "heavy" weight ω_h. *The set of neighbors of agent 6 is* $\mathrm{N}_6(k) = \{3, 5, 9, 11, 12, 13\}$, *where 11, 12 and 13 are not agents but vertices of the Voronoi cell* $\mathcal{V}_6(k)$ *of agent 6 lying on* $\partial\mathcal{Y}$. *Vertices 12 and 13 are the only neighbors of agent 6 in the area described before, thus* $\omega_{12,6}(k) = \omega_{13,6}(k) = \omega_h$. *Agent 3 is an outgoing agent and thus receives the weight* $\omega_{3,6}(k) = \omega_e$. *Finally, agents 5 and 9 as well as vertex 11 receive the weight* $\omega_{5,6}(k) = \omega_{9,6}(k) = \omega_{11,6}(k) = \omega_l$.

Finally, in Figure 5.11, the light green area, defined for agent 10, is the set of all points $\mathbf{y}$ *such that:*

$$\begin{aligned} \boldsymbol{h}_2(k)(\mathbf{y} - \mathbf{y}_2(k)) \cdot \boldsymbol{h}_2(k)(\mathbf{y}_{10}(k) - \mathbf{y}_2(k)) \geq 0 \\ d(\mathbf{y}, \partial\mathcal{H}_2(k)) > d(\mathbf{y}_{10}(k), \partial\mathcal{H}_2(k)) \\ \boldsymbol{h}_3(k)(\mathbf{y} - \mathbf{y}_3(k)) \cdot \boldsymbol{h}_3(k)(\mathbf{y}_{10}(k) - \mathbf{y}_3(k)) \geq 0 \\ d(\mathbf{y}, \partial\mathcal{H}_3(k)) > d(\mathbf{y}_{10}(k), \partial\mathcal{H}_3(k)). \end{aligned}$$

Thus, if a neighbor of agent 10 is inside this area, it receives the weight ω_h. *The set of neighbors of agent 10 is* $\mathrm{N}_{10}(k) = \{1, 2, 8, 9, 14, 15\}$ *where 14 and 15 are not agents but vertices of the Voronoi cell* $\mathcal{V}_{10}(k)$ *of agent 10 lying on* $\partial\mathcal{Y}$. *Vertex 15 and agent 8 are the only neighbors of agent 10 in the area described before, therefore* $\omega_{8,10}(k) = \omega_{15,10}(k) = \omega_h$. *Agent 2 is an outgoing agent and thus receives the weight* $\omega_{2,10}(k) = \omega_e$. *Finally, agents 1 and 9 as well as vertex 14 receive the weight* $\omega_{1,10}(k) = \omega_{9,10}(k) = \omega_{14,10}(k) = \omega_l$.

The weights attributed for the computation of $\mathbf{y}_5^{\mathrm{s}}(k)$, $\mathbf{y}_6^{\mathrm{s}}(k)$ *and* $\mathbf{y}_{10}^{\mathrm{s}}(k)$ *are presented in Figure 5.11 in blue, red and green, respectively.*

The same procedure can be applied to the other agents of Σ *but is not be presented here. However, the construction of the safe objective* $\mathbf{y}_i^{\mathrm{s}}(k)$, *with* $i \in \overline{1, 10} \setminus \{2, 3\}$ *is not presented in this example since an additional layer to the reconfiguration algorithm concerning the computation of this objective point is presented in Paragraph 5.2.2.2.*

5.2.2.2 A new reconfiguration algorithm

In Paragraph 5.1.2.3, a reconfiguration algorithm where the remaining agents of the multi-agent system Σ follow a transient objective to avoid the trajectory of agents leaving the workspace $\mathcal{Y}$ is proposed. With the new safe objective allowing the agents to evolve inside a safe region proposed in Paragraph 5.2.2.1, this algorithm is efficient in allowing the agents of Σ remaining in the workspace $\mathcal{Y}$ to avoid the trajectory of the outgoing agents, it can be improved on some aspects. Indeed, with such a strategy, all the agents of Σ remaining in the workspace $\mathcal{Y}$ move until all the outgoing agents leave $\mathcal{Y}$ before resuming their deployment objective. The present paragraph then proposes an improved way to deal with MAS reconfiguration in the case of outgoing agents.

In order to avoid too much movement from the MAS, only the agents risking a collision with an outgoing agent follow the new transient safe objective $\mathbf{y}_i^{\mathrm{s}}(k)$, with $i \in \overline{1, N} \setminus \mathrm{O}(k)$, defined in the previous paragraph, while the other agents continue their deployment objective by following the Chebyshev center $\mathbf{c}_i(k)$ of their Voronoi cell $\mathcal{V}_i(k)$. To do so, each agent of Σ has to discriminate which outgoing agents are harmless to it and which are not.

Let $\mathrm{O}_i(k)$, with $i \in \overline{1,N}\backslash\mathrm{O}(k)$, denote the set of all the outgoing agents considered potentially harmful by agent i of Σ. For all outgoing agents $o \in \mathrm{O}(k)$, let:

$$\partial\mathcal{O}_o(k) = \left\{ \mathbf{x} \in \mathbb{R}^2 \;\middle|\; \left(\mathbf{y}_o^{\mathrm{O}} - \mathbf{y}_o(k)\right)^\top (\mathbf{x} - \mathbf{y}_o(k)) = 0 \right\} \tag{5.21}$$

which separates the plane $\mathbb{R}^2$ into two parts. An outgoing agent $o \in \mathrm{O}(k)$ is then considered harmful by agent $i \in \overline{1,N} \setminus \mathrm{O}(k)$ if $\mathbf{y}_i(k)$ and $\mathbf{y}_o^{\mathrm{O}}$ lie on the same side of $\partial\mathcal{O}_o(k)$, i.e. if $\left(\mathbf{y}_o^{\mathrm{O}} - \mathbf{y}_o(k)\right)^\top (\mathbf{y}_i(k) - \mathbf{y}_o(k)) \geq 0$. In this case, if $o \notin \mathrm{O}_i(k)$, the outgoing agent o is added to $\mathrm{O}_i(k)$, otherwise, $\mathrm{O}_i(k)$ does not change. However, as soon as $\mathbf{y}_i(k)$ and $\mathbf{y}_o^{\mathrm{O}}$ lie on different sides of $\partial\mathcal{O}_o(k)$, i.e. when $\left(\mathbf{y}_o^{\mathrm{O}} - \mathbf{y}_o(k)\right)^\top (\mathbf{y}_i(k) - \mathbf{y}_o(k)) < 0$, the outgoing agent o is removed from $\mathrm{O}_i(k)$. This procedure is summarized in Algorithm 5.7.

Algorithm 5.7: Construction of the set $\mathrm{O}_i(k)$ of the outgoing agents considered by the agent i.

Input: The positions $\mathbf{y}_o(k)$ and objectives $\mathbf{y}_o^{\mathrm{O}}$, $\forall o \in \mathrm{O}(k)$, of the outgoing agents, the position $\mathbf{y}_i(k)$ of agent i and the set $\mathrm{O}_i(k-1)$

```
for o ∈ O(k) do
    if (y_o^O − y_o(k))^⊤ (y_i(k) − y_o(k)) ≥ 0 and o ∉ O_i(k − 1) then
        O_i(k) ← O_i(k − 1) ∪ {o};
    else if (y_o^O − y_o(k))^⊤ (y_i(k) − y_o(k)) < 0 and o ∈ O_i(k − 1) then
        O_i(k) ← O_i(k − 1) \ {o};
    end
end
```

Output: The updated set $\mathrm{O}_i(k)$ of the agent $i \in \overline{1,N} \setminus \mathrm{O}(k)$

Remark 5.4: Behind and in front

In the following, an agent satisfying $\left(\mathbf{y}_o^{\mathrm{O}} - \mathbf{y}_o(k)\right)^\top (\mathbf{y}_i(k) - \mathbf{y}_o(k)) \geq 0$ is said to be *in front* of the outgoing agent $o \in \mathrm{O}(k)$.

An agent satisfying $\left(\mathbf{y}_o^{\mathrm{O}} - \mathbf{y}_o(k)\right)^\top (\mathbf{y}_i(k) - \mathbf{y}_o(k)) < 0$ is said to be *behind* the outgoing agent $o \in \mathrm{O}(k)$. ◊

Based on the set $\mathrm{O}_i(k)$ constructed with Algorithm 5.7, agent i of Σ then computes its safe objective $\mathbf{y}_i^{\mathrm{s}}(k)$ by solving (5.17). However, a slight modification to Algorithm 5.6 is done. Indeed, the line 5 is changed from "**for** $o \in \mathrm{O}(k)$ **do**" to "**for** $o \in \mathrm{O}_i(k)$ **do**" to take into account the new way of dealing with outgoing agents. The same procedure is done for the construction of the contracted working region. Indeed, the definition of the working region given in (5.18) is replaced with:

$$\mathcal{W}_i(k) = \left\{ \mathbf{y} \in \mathbb{R}^2 \;\middle|\; \boldsymbol{h}_o(k)(\mathbf{y}_i(k) - \mathbf{y}_o(k)) \cdot \boldsymbol{h}_o(k)(\mathbf{y} - \mathbf{y}_o(k)) \geq 0,\ \forall o \in \mathrm{O}_i(k) \right\}. \tag{5.22}$$

Example 5.5: Construction of the sets $\mathrm{O}_i(k)$

Let Σ be the multi-agent system of Example 5.4 in the exact same conditions. The agents 2 and 3 leave the workspace to join the points $\mathbf{y}_2^{\mathrm{O}}$ and $\mathbf{y}_3^{\mathrm{O}}$ depicted as squares in Figure 5.12. In Figure 5.12 both hyperplanes $\partial\mathcal{O}_2(k)$ and $\partial\mathcal{O}_3(k)$ are presented as solid lines. Then, if an agent i of Σ remaining inside $\mathcal{Y}$ belongs to the area colored in light blue, the set $\mathrm{O}_i(k)$ of the outgoing agents it considers is such that

$\mathrm{O}_i(k) = \{2, 3\}$, *which is verified for agents 1, 6, 8, 9 and 10. Indeed, these agents are in front of both agents 2 and 3 with respect to their objectives. On the other hand, if an agent* i *of* Σ *remaining inside* $\mathcal{Y}$ *belongs to the area colored in light red, its set* $\mathrm{O}_i(k)$ *is such that* $\mathrm{O}_i(k) = 3$, *which is verified only for agent 5. Indeed, agent 5 is in front of agent 3 but behind agent 2. Finally, if an agent* i *of* Σ *remaining inside* $\mathcal{Y}$ *belongs to the white area, its set* $\mathrm{O}_i(k)$ *is empty since the agent is behind both agents 2 and 3. This is verified for agents 4 and 7 such that* $\mathrm{O}_4(k) = \mathrm{O}_7(k) = \emptyset$.

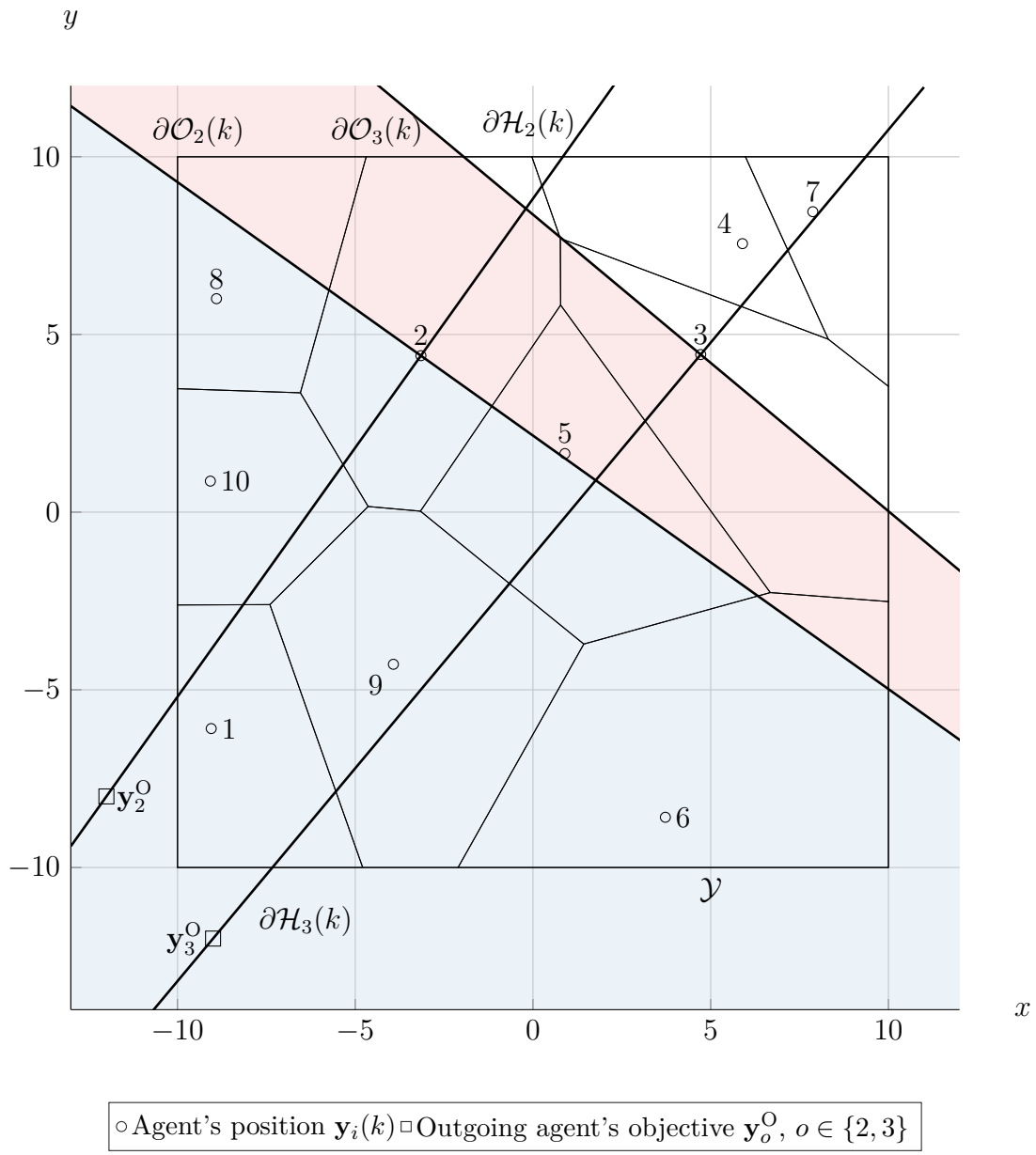


Figure 5.12: Construction of the sets $\mathrm{O}_i(k)$ for the agents of the MAS Σ.

Contrary to Example 5.4, the weight attributions for the neighbors of agent 5 is then slightly modified given the considered modification of Algorithm 5.6. Indeed, $d(\mathbf{y}_\nu(k), \partial\mathcal{H}_2(k))$, *with* $\nu \in \mathrm{N}_5(k)$, *is not considered anymore for the weight attribution procedure. However, this does not change the result given the configuration of the system.*

In addition, since the definition of the working region of an agent has been modified from (5.18) to (5.22), some conclusions of Example 5.3 are modified. Indeed, given the configuration of the MAS at time k, $\mathcal{W}_4(k)$ and $\mathcal{W}_7(k)$ are not defined since $\mathrm{O}_4(k) = \mathrm{O}_7(k) = \emptyset$. Moreover, since $\mathrm{O}_5(k) = \{3\} \neq \mathrm{O}_9(k)$, it is immediate that $\mathcal{W}_5(k) \neq \mathcal{W}_9(k)$.

There is no modification in the behavior of the outgoing agents with respect to what was presented in Paragraph 5.1.2.3, i.e. the outgoing agents then still follow the decentralized MPC algorithm of Algorithm 5.2 to leave the workspace $\mathcal{Y}$. However, for the remaining agents, Algorithm 5.4 is modified to incorporate all the improvements presented in this section.

If an agent i, with $i \in \overline{1,N} \setminus \mathrm{O}(k)$, remaining in the workspace $\mathcal{Y}$ is such that $\mathrm{O}_i(k) = \emptyset$, it follows Algorithm 3.2 without modification. However, if $\mathrm{O}_i(k) \neq \emptyset$, agent i follows the safe objective $\mathbf{y}_i^{\mathrm{s}}(k)$ with a decentralized MPC algorithm.

According to Assumption 3.2, let $(\mathbf{x}_i^{\mathrm{s}}(k), \mathbf{u}_i^{\mathrm{s}}(k))$, with $i \in \overline{1,N} \setminus \mathrm{O}(k)$, be a couple such that $(\mathbf{x}_i^{\mathrm{s}}(k), \mathbf{u}_i^{\mathrm{s}}(k), \mathbf{y}_i^{\mathrm{s}}(k))$ is an equilibrium point of (5.1), i.e.:

$$\begin{aligned} \mathbf{x}_i^{\mathrm{s}}(k) &= \boldsymbol{A}\mathbf{x}_i^{\mathrm{s}}(k) + \boldsymbol{B}\mathbf{u}_i^{\mathrm{s}}(k) \\ \mathbf{y}_i^{\mathrm{s}}(k) &= \boldsymbol{C}\mathbf{x}_i^{\mathrm{s}}(k). \end{aligned} \tag{5.23}$$

Then, the agents of Σ such that $\mathrm{O}_i(k) \neq \emptyset$ compute their input $\mathbf{u}_i(k)$, with $i \in \overline{1,N} \setminus \mathrm{O}(k)$, by finding the solution of problem (5.13), while replacing $\mathbf{x}_i^{\mathrm{b}}(k)$ and $\mathbf{u}_i^{\mathrm{b}}(k)$ by $\mathbf{x}_i^{\mathrm{s}}(k)$ and $\mathbf{u}_i^{\mathrm{s}}(k)$, respectively. The procedure for the reconfiguration of the MAS Σ in the case of outgoing agents is then summarized in Algorithm 5.8.

Algorithm 5.8: Decentralized algorithm followed by a remaining agent for the reconfiguration of the MAS when several agents leave the system.

```
1  for k ≥ 0 do
2      if O ≠ ∅ then
3          Compute O_i(k) of agent i ∈ 1,N \ O with Algorithm 5.7;
4          if O_i(k) = ∅ then
5              Follow Algorithm 3.2;
6          else
7              Attribute weights to agent i's neighbors with Algorithm 5.6
                (modified with O_i(k) instead of O at line 5);
8              Compute the safe objective y_i^s(k) of agent i by solving (5.17);
9              Compute the couple (x_i^s(k), u_i^s(k)) with (5.12) such that
                (x_i^s(k), u_i^s(k), y_i^s(k)) is an equilibrium point of (5.1);
10             Solve the optimization problem (5.13) (by replacing x_i^b(k) and
                u_i^b(k) by x_i^s(k) and u_i^s(k)) to obtain the input signal u_i(k);
11             Apply u_i(k) to agent i;
12         end
13     else
14         Follow Algorithm 3.2;
15     end
16 end
```

5.3 Reconfiguration in the case of outgoing agents

5.3.1 Comparison of the two algorithms in the case of one outgoing agent

In order to exhibit the improvement of the algorithm described in Section 5.2 with respect to the one presented in Paragraph 5.1.2.3, the exact same scenario as the one described in Paragraph 5.1.3.2 is considered. However, instead of following Algorithm 5.4 used when only one agent leaves the workspace, the remaining agents of Σ follow Algorithm 5.8 used when one or several agents leave the workspace. Agent 7 still follows Algorithm 5.2 and leaves Σ at $t = 5\,\mathrm{s}$.

The first phase of the deployment is identical to the one presented in Figure 5.4 since it only consists in the deployment of the MAS and is then not reported here.

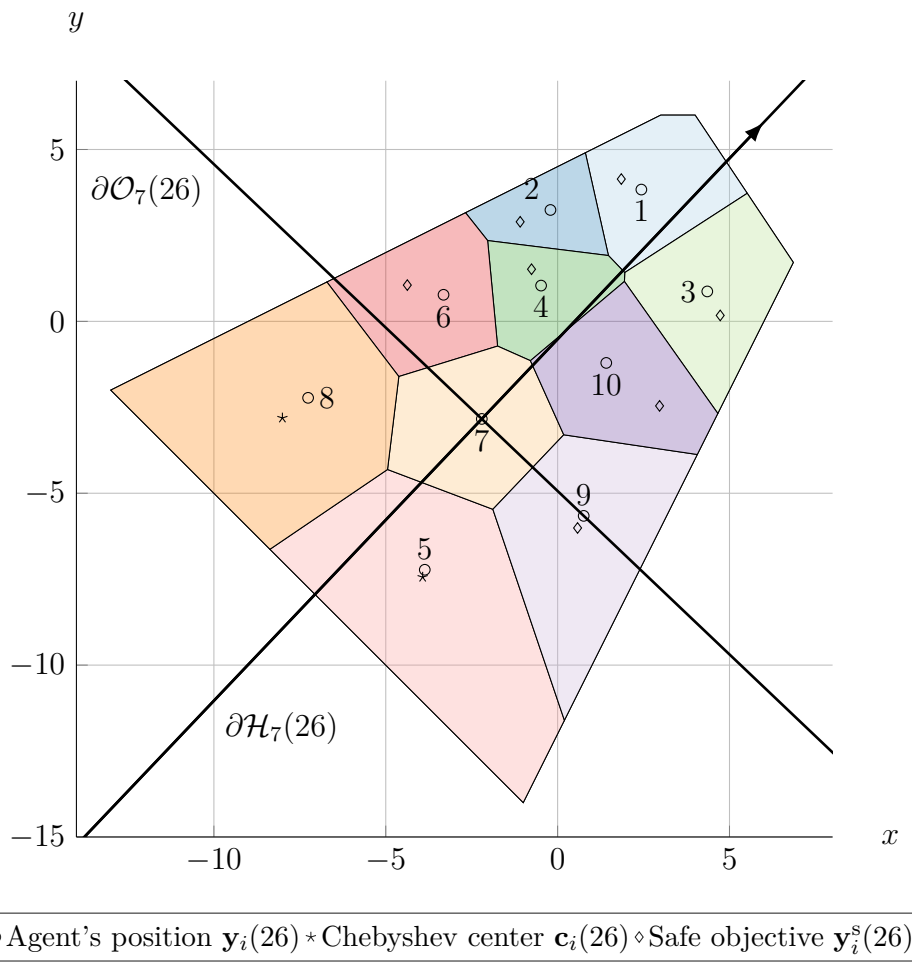


Figure 5.13: Position of the MAS Σ and of agent 7 at time $t = 5.2\,\mathrm{s}$.

Figure 5.13 presents the position of Σ and agent 7 at time $t = 5.2\,\mathrm{s}$ (i.e. $k = 26$). In this figure are presented the hyperplanes $\partial\mathcal{H}_7(26)$ and $\partial\mathcal{O}_7(26)$ described in Section 5.2. Since agent 7 moves towards the point $\mathbf{y}_7^{\mathrm{O}} = \begin{bmatrix}10 & 10\end{bmatrix}^\top$, agents 5 and 8 are behind agent 7 and, per Algorithm 5.7, $\mathrm{O}_5(26) = \mathrm{O}_8(26) = \emptyset$ since they are not in the same half-space delimited by $\partial\mathcal{O}_7(26)$ as $\mathbf{y}_7^{\mathrm{O}}$. Agents 5 and 8 then continue to

track their Chebyshev center $\mathbf{c}_5(k)$ and $\mathbf{c}_8(k)$, represented as stars in Figure 5.13, according to Algorithm 5.8. However, though agent 9 is in the limit case as it lies on $\partial\mathcal{O}_7(26)$, it is considered to be in front of agent 7 and $\mathrm{O}_i(26) = \{7\}$ for all $i \in \overline{1, N} \setminus \{5, 7, 8\}$ since they are in front of agents 7. They then start tracking the safe objective $\mathbf{y}_i^{\mathrm{s}}(26)$, with $i \in \overline{1, N} \setminus \{5, 7, 8\}$, represented as diamonds in Figure 5.13, according to Algorithm 5.8.

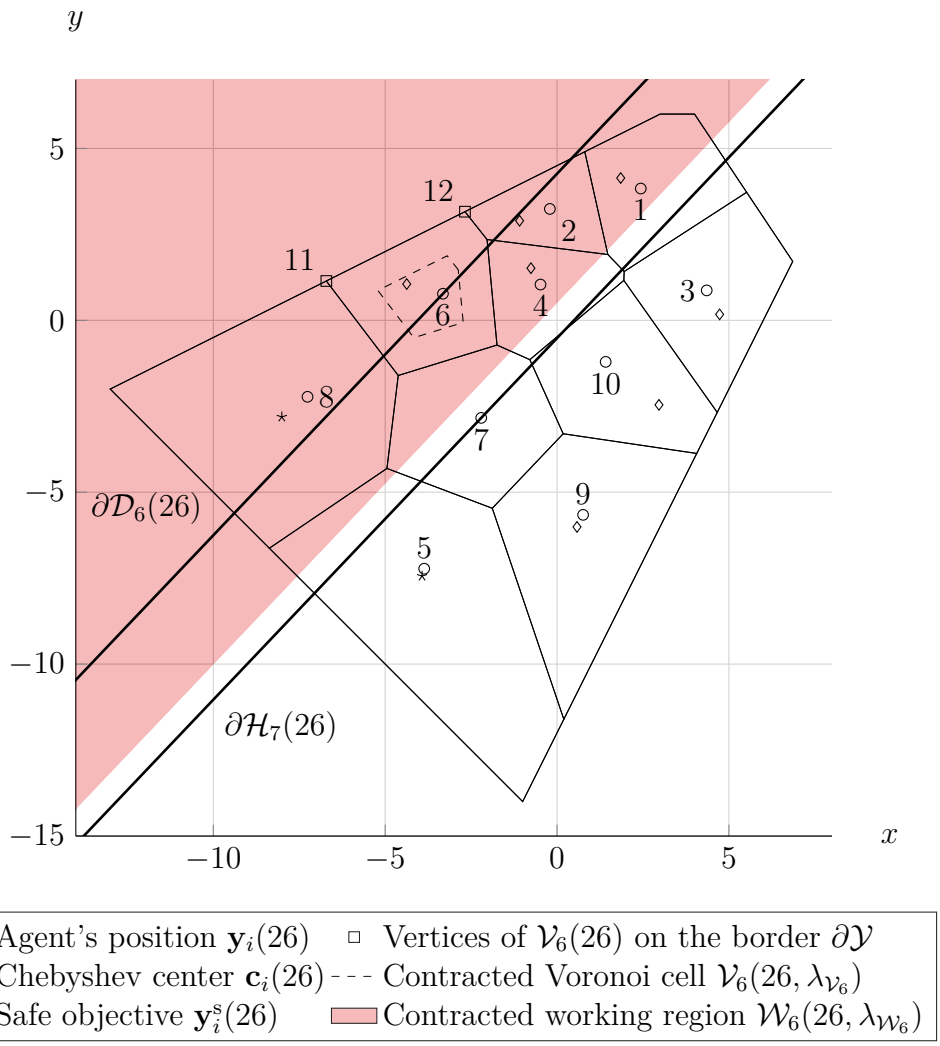


Figure 5.14: Construction of the safe objective of agent 6 at $t = 5.2\,\mathrm{s}$.

For the computation of the safe objectives $\mathbf{y}_i^{\mathrm{s}}(k)$, with $i \in \overline{1, N} \setminus \{5, 7, 8\}$, the weights $\lambda_{\mathcal{V}_i}$ and $\lambda_{\mathcal{W}_i}$ described in Paragraph 5.2.2.1 are chosen to be 0.5 and 0.8, respectively, for all the agents. Figure 5.14 details the construction of the safe objective $\mathbf{y}_6^{\mathrm{s}}(26)$, the procedure being the same for all the other agents. The line $\partial\mathcal{D}_6(26)$ is used to graphically see which neighbors of agent 6 receive a heavy weight ω_h and which neighbors receive a light weight ω_l. Indeed, all the points $\mathbf{y} \in \mathbb{R}^2$ above $\partial\mathcal{D}_6(26)$ are such that $d(\mathbf{y}, \partial\mathcal{H}_7(26)) \geq d(\mathbf{y}_6(26), \partial\mathcal{H}_7(26))$. The set of neighbors of agent 6 $\mathrm{N}_6(26)$ is $\mathrm{N}_6(26) = \{2, 4, 7, 8, 11, 12\}$, where 11 and 12 are vertices of the Voronoi cell $\mathcal{V}_6(26)$ lying on the border $\partial\mathcal{Y}$ of the workspace $\mathcal{Y}$. Then, per Algorithm 5.6, $\omega_{\nu,6}(26) = \omega_h$ for all $\nu \in \{8, 11, 12\}$, $\omega_{\nu,6}(26) = \omega_l$ for all $\nu \in \{2, 4\}$ and $\omega_{7,6}(26) = \omega_e$. The contracted Voronoi cell $\mathcal{V}_6(26, \lambda_{\mathcal{V}_6})$ of agent 6 is the area delimited by the dashed line in Figure 5.14. The contracted working region

$\mathcal{W}_6(26, \lambda_{\mathcal{W}_6})$ of agent 6 is the area colored in red in Figure 5.14. It can be noted that, as explained in Paragraph 5.2.2.1, $\mathcal{W}_6(26, \lambda_{\mathcal{W}_6}) = \mathcal{W}_i(26, \lambda_{\mathcal{W}_i})$ for all $i \in \{1, 2, 4\}$. Then, by choosing $\omega_h = 5$, $\omega_l = 1$ and $\omega_e = 0$, the safe objective $\mathbf{y}_6^{\mathrm{s}}(26)$ of agent 6 is obtained by solving (5.17).

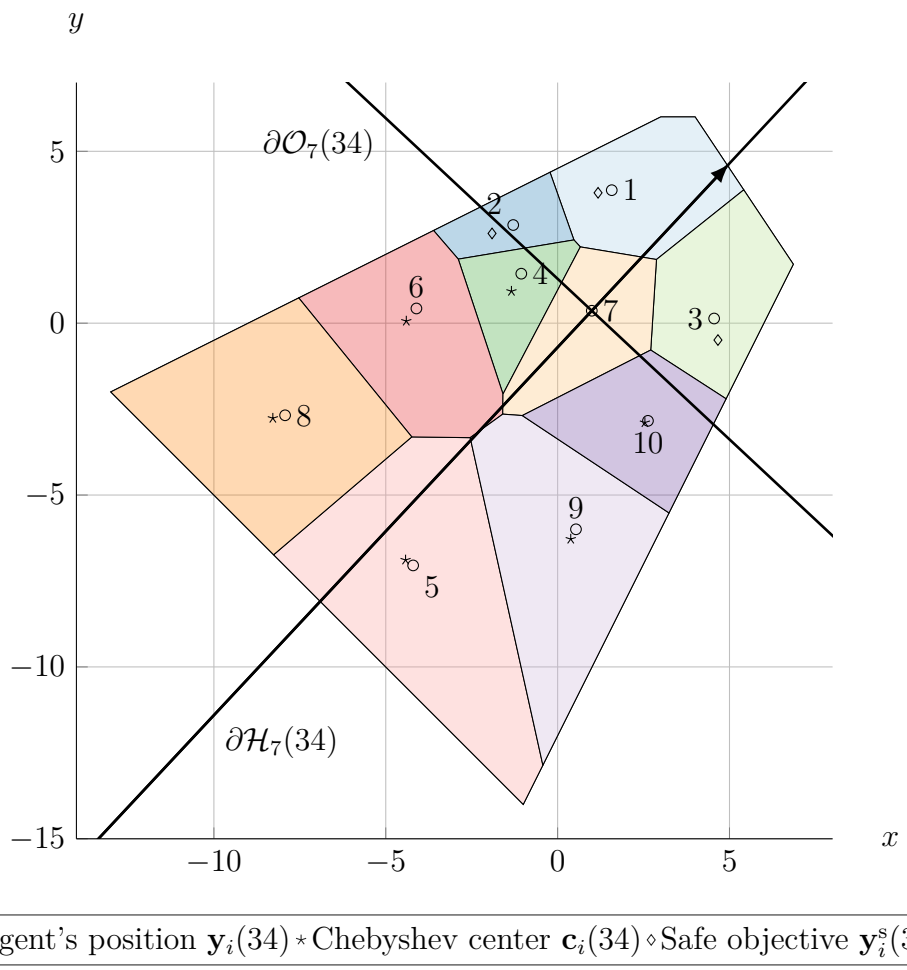


Figure 5.15: Position of the MAS Σ and of agent 7 at time $t = 6.8\,\mathrm{s}$.

As agent 7 moves towards its objective $\mathbf{y}_7^{\mathrm{O}}$ as shown in Figure 5.14, the agents tracking their safe objective $\mathbf{y}_i^{\mathrm{s}}(k)$, with $i \in \overline{1, N} \setminus \{5, 7, 8\}$, resume tracking their Chebyshev centers when they pass on the other side of $\partial\mathcal{O}_7(k)$, i.e. as soon as they are behind agent 7. For example, at time $t = 6.8\,\mathrm{s}$ ($k = 34$), agents 4, 6, 9 and 10 resumed tracking the Chebyshev center of their Voronoi cell as illustrated in Figure 5.15. The only agents still tracking their safe objective are agents 1, 2 and 3.

It is difficult to show the trajectories of the agents and of their objectives over time as it is done in Paragraph 5.1.3.2, since the movement cannot be decomposed into distinct phases. Indeed, all the agents change their objective asynchronously, contrary to the case of Paragraph 5.1.2.3, where they track their neighbors' barycenter as soon as agent 7 leaves Σ and starts leaving $\mathcal{Y}$ and resume tracking their Chebyshev center when agent 7 is out of $\mathcal{Y}$. Then, only the distances:

$$d_i(k) = \begin{cases} \|\mathbf{y}_i(k) - \mathbf{y}_i^{\mathrm{s}}(k)\|_2 & \text{if } \mathrm{O}_i(k) \neq \emptyset \\ \|\mathbf{y}_i(k) - \mathbf{c}_i(k)\|_2 & \text{otherwise} \end{cases}$$

for all $i \in \overline{1, N} \setminus \{7\}$ are shown in Figure 5.16. The distance $d_7(k)$ is identical to the one showed in Figure 5.8 since agent 7 follows the same algorithm as in Paragraph 5.1.3.2. From Figure 5.16, it can be seen that the overall movement of the multi-agent system Σ is reduced compared to what was achieved in Figure 5.7 with the algorithm from Paragraph 5.1.2.3. This reduction comes from the fact that the remaining agents do not wait for the outgoing agents to have left the workspace $\mathcal{Y}$ to resume their deployment by tracking the Chebyshev center of their Voronoi cell. Moreover, the changes occurring in the distances d_i have a reduced amplitude compared to those of Paragraph 5.1.3.2 and occur less often. Near the end of the simulation, a pike appears in $d_9(k)$ due to a sudden change in the position of $\mathbf{c}_9(k)$ resulting from the global movement of Σ.

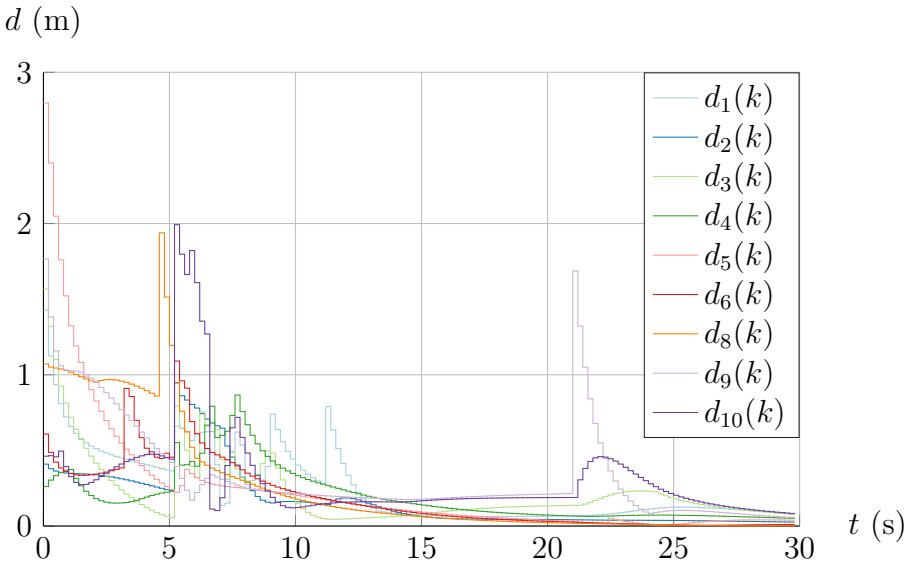


Figure 5.16: Distance of each agent of Σ to its Chebyshev center or safe objective over time.

Finally, Algorithm 5.8 not only provides a safe objective for the reconfiguration of Σ when one agent leaves the MAS, but it also reduces the overall movement of the MAS during reconfiguration. Such a result shows that Algorithm 5.8 would, for example, reduce the energy consumption of the agents of the MAS compared to what Algorithm 5.4 achieves while enabling it to perform a safe reconfiguration.

5.3.2 Reconfiguration in the case of multiple outgoing agents

The following section provides simulation results for the reconfiguration of a multi-agent system in the case of several agents leaving the MAS. The reconfiguration strategy for the remaining agents of Σ is the one described in Algorithm 5.8, while the outgoing agents use Algorithm 5.2. The simulations are run for the two dynamics considered along this thesis, single integrator and UAV dynamics.

5.3.2.1 Reconfiguration for single integrator dynamics

Let Σ be a multi-agent system composed of $N = 12$ agents obeying the single integrator dynamics (3.15). These agents are deployed in the workspace $\mathcal{X} = \mathcal{Y} =$

$\mathbb{B}^2(7.5 \cdot \mathbf{1}_{2\times1})$ with an input space $\mathcal{U} = \mathbb{B}^2(2 \cdot \mathbf{1}_{2\times1})$. The sampling rate used in (3.15) is $T_s = 0.2\,\mathrm{s}$. The prediction horizon for the model predictive controllers of Algorithm 5.2 and Algorithm 5.8 is $N_p = 10$ and the contraction factor for the terminal constraint is $\lambda_i = 0.9$ for all $i \in \overline{1, N}$. The weighting matrices for the MPC are $\boldsymbol{Q} = \boldsymbol{R} = \boldsymbol{I}_2$ and $\boldsymbol{P}$ is the solution of the algebraic Riccati equation.

For this simulation scenario, the agents of Σ start deploying inside $\mathcal{Y}$ from random positions $\mathbf{y}_i(0) \in \mathcal{Y}$, with $i \in \overline{1, N}$ represented by circles in Figure 5.17. At $t = 4\,\mathrm{s}$, agent 7 leaves Σ to join $\mathbf{y}_7^{\mathrm{O}} = \begin{bmatrix}0 & 10\end{bmatrix}^\top$. Then, while agent 7 is leaving $\mathcal{Y}$, agent 9 also leaves Σ at $t = 5.2\,\mathrm{s}$ to join $\mathbf{y}_9^{\mathrm{O}} = \begin{bmatrix}10 & 0\end{bmatrix}^\top$. Thus, during the deployment, the set $\mathrm{O}(k)$ of the outgoing agents is:

$$\mathrm{O}(k) = \begin{cases} \emptyset & \text{for} \quad k < 20 \\ \{7\} & \text{for} \quad 20 \leq k < 26. \\ \{7, 9\} & \text{for} \quad 26 \leq k \end{cases}$$

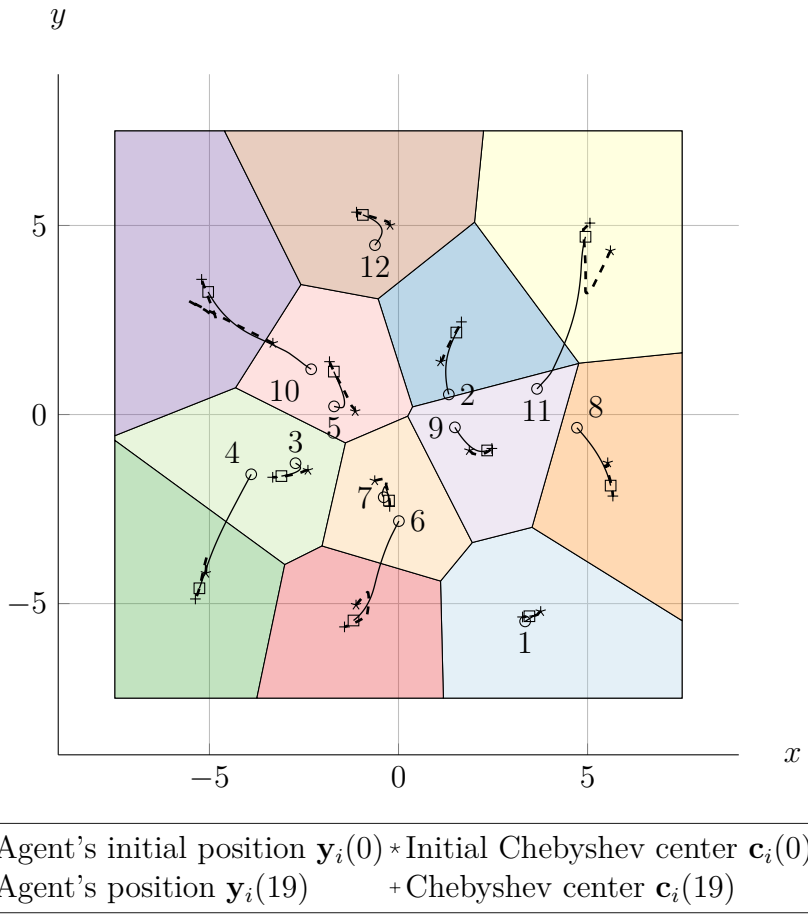


Figure 5.17: Trajectories of the agents of Σ during the first phase of the deployment.

While agents 7 and 9 leave the workspace, the other agents are in a reconfiguration phase before resuming their deployment to reach a Chebyshev configuration as presented in Algorithm 5.8. For the reconfiguration phase, the weights ω_h, ω_l and ω_e used to compute the safe objective $\mathbf{y}_i^{\mathrm{s}}(k)$ for all $i \in \overline{1, N} \setminus \mathrm{O}(k)$ are chosen to be 5, 1 and 0 respectively. The contraction factor to compute the contracted Voronoi cells $\mathcal{V}_i(k, \lambda_{\mathcal{V}_i})$ is $\lambda_{\mathcal{V}_i} = 0.5$ for all $i \in \overline{1, N} \setminus \mathrm{O}(k)$ and the contraction factor to compute the contracted working region $\mathcal{W}_i(k, \lambda_{\mathcal{W}_i})$ is $\lambda_{\mathcal{W}_i} = 0.8$ for all $i \in \overline{1, N} \setminus \mathrm{O}(k)$.

Figure 5.17 shows the trajectories of the agents of Σ as solid lines and of the Chebyshev center of their Voronoi cells during the first phase of the deployment as dashed lines, i.e. for all k such that $\mathrm{O}(k) = \emptyset$, in the Voronoi tessellation at $t = 3.8\,\mathrm{s}$. The observations that could be made on the MAS are the same as what is explained in Section 3.2.4.

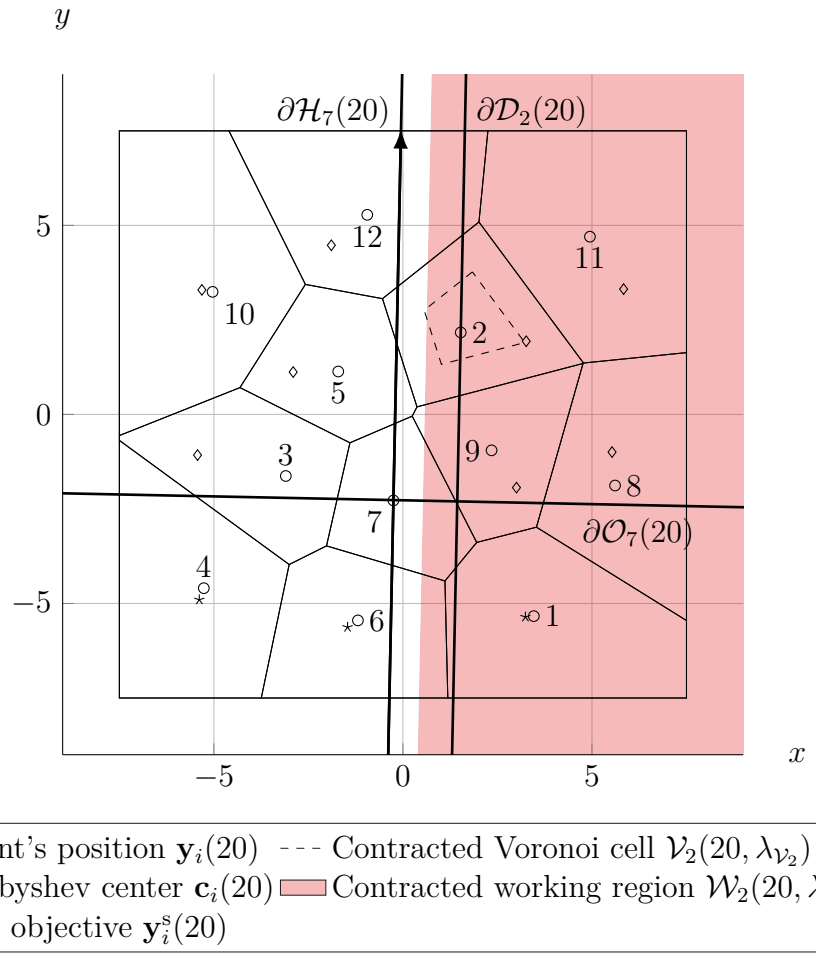


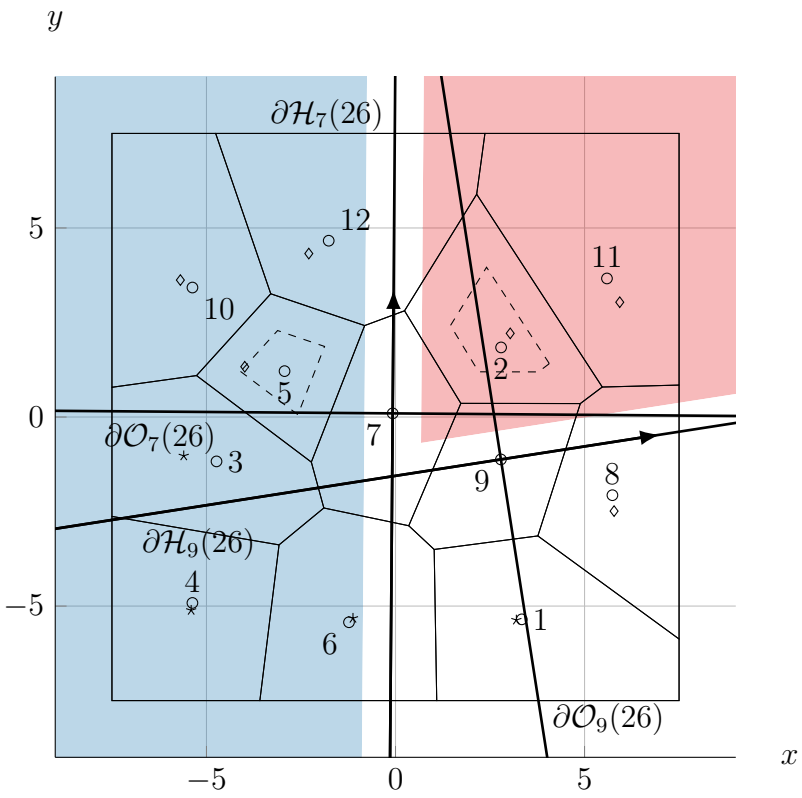
Figure 5.18: Construction of the safe objective of agent 2 at $t = 4\,\mathrm{s}$.

In Figure 5.18, the construction of the safe objective $\mathbf{y}_2^{\mathrm{s}}(20)$ of agent 2 is detailed. This agent is chosen since it is in front of agent 7 and has a collision risk with the outgoing agent. The light red area is the contracted working region $\mathcal{W}_2(20, \lambda_{\mathcal{W}_2})$ of agent 2 (which is also the contracted working region of agents 8, 9 and 11). The contracted Voronoi cell $\mathcal{V}_2(20, \lambda_{\mathcal{V}_2})$ of agent 2 is the area delimited by the dashed line. Then, the hyperplane $\partial\mathcal{D}_2(20)$ is:

$$\partial\mathcal{D}_2(20) = \left\{\mathbf{y} \in \mathbb{R}^2 \,\middle|\, d(\mathbf{y}, \partial\mathcal{H}_7(20)) = d(\mathbf{y}_2(20), \partial\mathcal{H}_7(20))\right\}$$

such that all the neighbors of agent 2 on the right of $\partial\mathcal{D}_2(20)$ receive the heavy weight ω_h. The neighbor set $\mathrm{N}_2(20)$ of agent 2 is $\mathrm{N}_2(20) = \{5, 8, 9, 11, 12\}$. Given the relative positions of the neighbors of agent 2 to $\partial\mathcal{D}_2(20)$, the weight terms $\omega_{\nu,2}(20) = \omega_h$ for all $\nu \in \{8, 9, 11\}$ and $\omega_{\nu,2}(20) = \omega_l$ for all $\nu \in \{5, 12\}$ are attributed. Then, the safe objective $\mathbf{y}_2^{\mathrm{s}}(20)$ is computed by solving (5.17) with the elements described before. The construction of the safe objectives $\mathbf{y}_i^{\mathrm{s}}(20)$ for all $i \in \{3, 5, 8, 9, 10, 11, 12\}$ is done by following the same procedure. It can also be seen from Figure 5.18 that agents 1,

4 and 6 continue tracking their Chebyshev center according to Algorithm 5.7 since they are behind the outgoing agent 7.



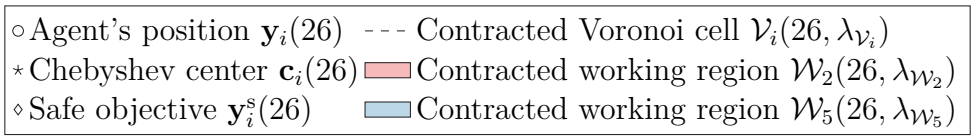


Figure 5.19: Construction of the safe objectives of agents 2 and 5 at $t = 5.2\,\mathrm{s}$.

While agent 7 is leaving $\mathcal{Y}$, agent 9 decides to leave Σ at $t = 5.2\,\mathrm{s}$. Then, Figure 5.19 presents the position of the agents at $t = 5.2\,\mathrm{s}$ and details the construction of the safe objectives of agents 2 and 5. According to Algorithm 5.7, at time $t = 5.2\,\mathrm{s}$, $\mathrm{O}_i(26) = \emptyset$ for $i \in \{1, 3, 4, 6\}$ since these agents are behind both outgoing agents, thus, agents 1, 3, 4 and 6 follow the Chebyshev center of their Voronoi cells. At time $t = 5.2\,\mathrm{s}$ the sets of the outgoing agents considered by agents 2 and 5 are, respectively, $\mathrm{O}_2(26) = \{7, 9\}$ and $\mathrm{O}_5(26) = \{7\}$ since agent 2 is in front of both outgoing agents, while agent 5 is in front of agent 7 and behind of agent 9. The neighbor sets $\mathrm{N}_2(26)$ and $\mathrm{N}_5(26)$ of agents 2 and 5 are $\mathrm{N}_2(26) = \{7, 8, 9, 11, 12\}$ and $\mathrm{N}_5(26) = \{3, 7, 10, 12\}$. Agent 11 is the only neighbor of agent 2 farther from $\partial\mathcal{H}_7(26)$ and $\partial\mathcal{H}_9(26)$ than agent 2 and on the same side of these hyperplanes as agent 2. Then, according to Algorithm 5.6, agent 11 receives the weight ω_h such that $\omega_{11,2}(26) = \omega_h$. Agents 8 and 12 are not on the same side of both $\partial\mathcal{H}_7(26)$ and $\partial\mathcal{H}_9(26)$ as agent 2, thus they receive the light weight ω_l such that $\omega_{\nu,2}(26) = \omega_l$ for $\nu \in \{8, 12\}$. Finally, since agents 7 and 9 are outgoing agents, they receive the weight ω_e such that $\omega_{\nu,2}(26) = \omega_e$ for $\nu \in \{7, 9\}$. The same way, $\omega_{\nu,5}(26) = \omega_h$ for $\nu \in \{3, 10\}$, $\omega_{12,5}(26) = \omega_l$ and $\omega_{7,5}(26) = \omega_e$. The areas inside the dashed lines are the contracted Voronoi cells $\mathcal{V}_2(26, \lambda_{\mathcal{V}_2})$ and $\mathcal{V}_5(26, \lambda_{\mathcal{V}_5})$. From the definition of

the contracted working region given in Paragraph 5.2.2.2 in (5.22), the contracted working region $\mathcal{W}_5(26, \lambda_{\mathcal{W}_5})$ of agent 5 is obtained by considering only the half-space defined by $\partial\mathcal{H}_7(26)$. This contracted working region is colored in blue in Figure 5.19. For its part, the contracted working region $\mathcal{W}_2(26, \lambda_{\mathcal{W}_2})$ of agent 2 is obtained by considering the half-spaces defined by both $\partial\mathcal{H}_7(26)$ and $\partial\mathcal{H}_9(26)$. This region is colored in red in Figure 5.19. Then, solving (5.17), the safe objectives $\mathbf{y}_2^{\mathrm{s}}(26)$ and $\mathbf{y}_5^{\mathrm{s}}(26)$ of agents 2 and 5 are found. The other safe objectives are found by following the same procedure.

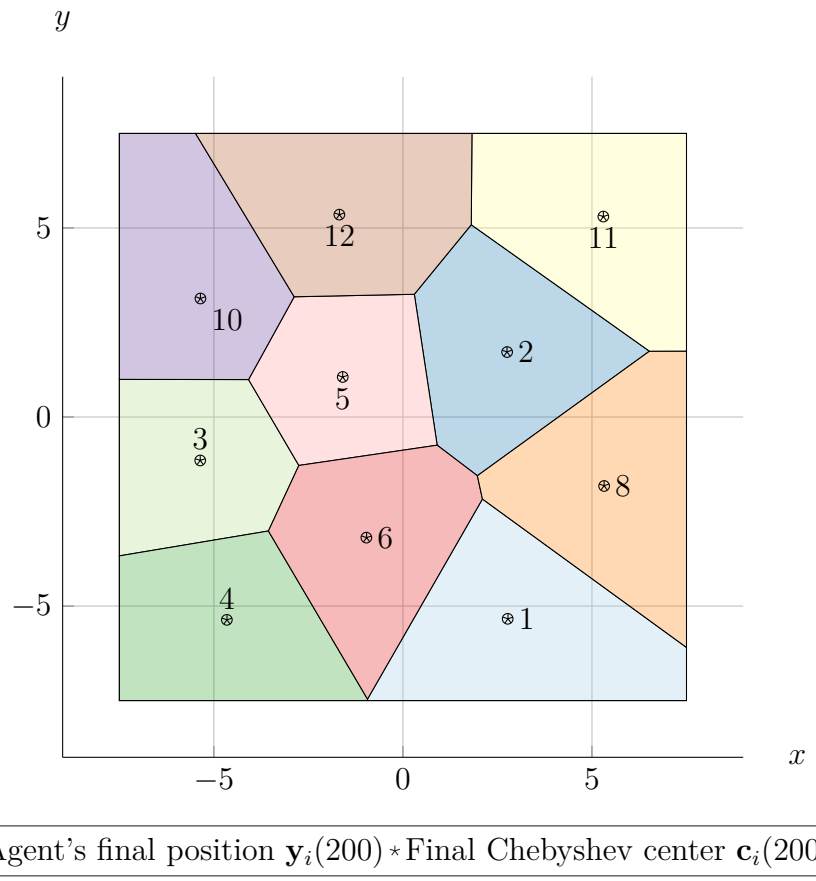


Figure 5.20: Final configuration of the MAS Σ at $t = 40\,\mathrm{s}$.

Agents 7 and 9 continue tracking their objectives $\mathbf{y}_7^{\mathrm{O}}$ and $\mathbf{y}_9^{\mathrm{O}}$ and leave $\mathcal{Y}$ at $t = 9\,\mathrm{s}$ and $t = 7.6\,\mathrm{s}$. In between the time the agents 7 and 9 leave Σ and the time they are outside $\mathcal{Y}$, the remaining agents of Σ avoid the trajectories of agents 7 and 9 by following their safe objectives $\mathbf{y}_i^{\mathrm{s}}(k)$, with $i \in \overline{1, N} \setminus \mathrm{O}(k)$. Moreover, as soon as they are behind both outgoing agents, the remaining agents resume tracking the Chebyshev center of their Voronoi cells $\mathbf{c}_i(k)$. The final Voronoi configuration at $t = 40\,\mathrm{s}$ is presented in Figure 5.20.

Finally, Figure 5.21 presents the distances:

$$d_i(k) = \begin{cases} \|\mathbf{y}_i(k) - \mathbf{y}_i^{\mathrm{s}}(k)\|_2 & \text{if } \mathrm{O}_i(k) \neq \emptyset \\ \|\mathbf{y}_i(k) - \mathbf{c}_i(k)\|_2 & \text{otherwise} \end{cases}$$

for all $i \in \overline{1, N} \setminus \{7, 9\}$, while Figure 5.22 presents the distances:

$$d_i(k) = \begin{cases} \left\|\mathbf{y}_i(k) - \mathbf{y}_i^{\mathrm{O}}(k)\right\|_2 & \text{if } i \in \mathrm{O}(k) \\ \|\mathbf{y}_i(k) - \mathbf{c}_i(k)\|_2 & \text{otherwise} \end{cases}$$

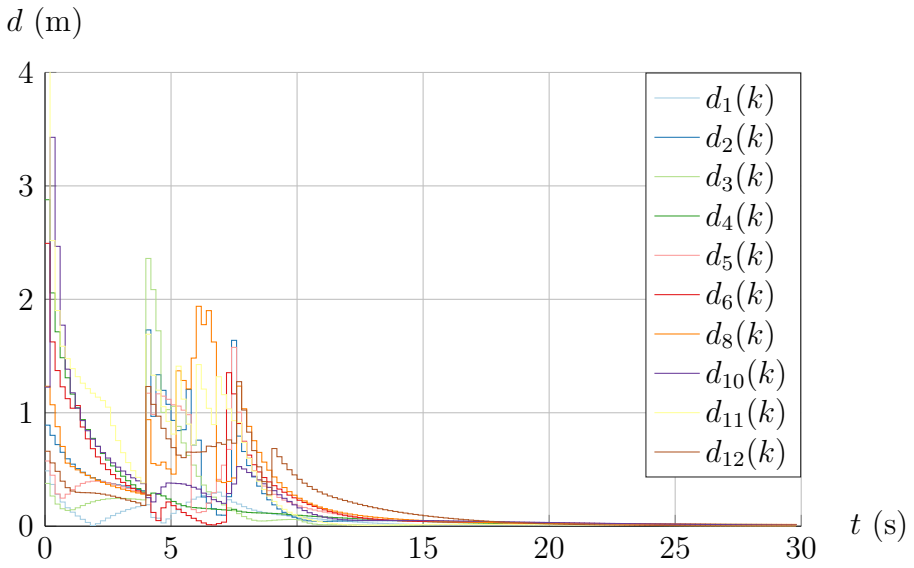


Figure 5.21: Distance of each agent of Σ to its Chebyshev center or safe objective over time.

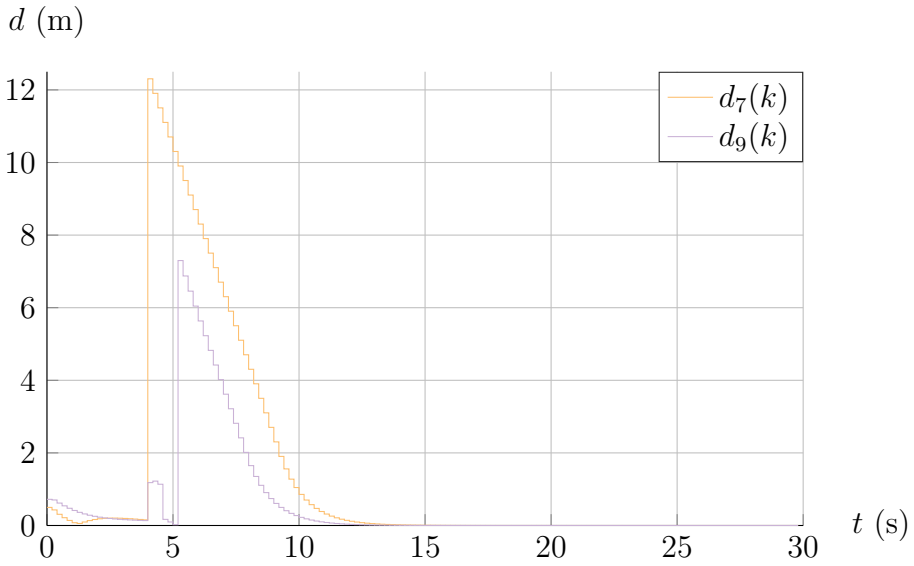


Figure 5.22: Distance of the agents leaving Σ to their objectives over time.

for $i \in \{7, 9\}$. From Figure 5.21, it can be seen that the reconfiguration strategy allows the agents to have a limited number of changes in their objectives during the reconfiguration phase. It also allows the system to return quickly to a reconfiguration strategy such that the MAS reaches a static Chebyshev configuration quickly after the last outgoing agent has left the workspace $\mathcal{Y}$. The main changes occur on the distances of the agents of Σ to their objectives at $t = 4\,\mathrm{s}$ and $t = 5.2\,\mathrm{s}$, i.e. when agents 7 and 9 leave Σ and when the agents of Σ resume tracking the Chebyshev center of their Voronoi cells. Figure 5.22 shows that the outgoing agents 7 and 9 are able to reach their objectives $\mathbf{y}_7^{\mathrm{O}}$ and $\mathbf{y}_9^{\mathrm{O}}$.

5.3.2.2 Reconfiguration for UAV dynamics

Let Σ be a multi-agent system composed of $N = 12$ agents obeying the UAV dynamics presented in Section 3.3. Then, for position control, the linearized dynamics of (3.30) are used, while the simulation is run with the nonlinear model of a quadrotor UAV (3.28). The MAS is deployed in the workspace $\mathcal{Y} = \mathbb{B}^2(7.5 \cdot \mathbf{1}_{2\times 1})$ with the input space $\mathcal{U} = \mathbb{B}^2\left(\frac{\pi}{6} \cdot \mathbf{1}_{2\times 1}\right)$. In addition to the output and input space, the state space is defined as:

$$\mathcal{X} = \left\{ \mathbf{x} \in \mathbb{R}^4 \;\middle|\; \boldsymbol{C}\mathbf{x} \in \mathcal{Y} \text{ and } \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x} \leq 2 \cdot \mathbf{1}_{4\times 1} \right\}$$

thus constraining the horizontal speed of each agent $i \in \overline{1, N}$ to $|v_{x,i}|, |v_{y,i}| \leq 2\,\mathrm{m} \cdot \mathrm{s}^{-1}$. The sampling rate used in the position subsystem of (3.30) is $T_s = 0.2\,\mathrm{s}$. The prediction horizon for the model predictive controllers of Algorithm 5.2 and Algorithm 5.8 is $N_p = 10$ and the contraction factor for the terminal constraint is $\lambda_i = 0.9$ for all $i \in \overline{1, N}$. The weighting matrices for the MPC are $\boldsymbol{Q} = \boldsymbol{I}_4$, $\boldsymbol{R} = \boldsymbol{I}_2$ and $\boldsymbol{P}$ is the solution of the algebraic Riccati equation. As in Section 3.3.3, it is necessary to introduce a Luenberger observer to estimate the state of the position subsystem. The observer gain is obtained as a linear quadratic regulator with weighting matrices $\boldsymbol{Q}_{\text{obs}} = 10\boldsymbol{I}_4$ and $\boldsymbol{R}_{\text{obs}} = \boldsymbol{I}_2$, giving:

$$\boldsymbol{L} = \begin{bmatrix} 1.0971 & 0 \\ 0.8303 & 0 \\ 0 & 1.0971 \\ 0 & 0.8303 \end{bmatrix}.$$

The example presented here follows the same idea as the one presented in Paragraph 5.3.2.1. The agents of Σ start deploying inside $\mathcal{Y}$ from random positions $\mathbf{y}_i(0) \in \mathcal{Y}$, while the other states have a null initial value except the altitude which is such that $z_i(0) = 5\,\mathrm{m}$ for all $i \in \overline{1, N}$. The estimated states $\widehat{\mathbf{x}}_i$ of all the agents $i \in \overline{1, N}$ are initialized such that $\widehat{\mathbf{x}}_i(0) = \mathbf{x}_i(0)$. At $t = 4\,\mathrm{s}$, agent 4 leaves Σ to join $\mathbf{y}_4^{\mathrm{O}} = \begin{bmatrix} 5 & 10 \end{bmatrix}^\top$. Then, while agent 4 is leaving $\mathcal{Y}$, agent 10 leaves Σ at $t = 5.2\,\mathrm{s}$ to rally $\mathbf{y}_{10}^{\mathrm{O}} = \begin{bmatrix} 10 & 10 \end{bmatrix}^\top$. Thus, during the deployment, the set $\mathrm{O}(k)$ of the outgoing agents is:

$$\mathrm{O}(k) = \begin{cases} \emptyset & \text{for} \quad k < 20 \\ \{4\} & \text{for} \quad 20 \leq k < 26. \\ \{4, 10\} & \text{for} \quad 26 \leq k \end{cases}$$

While agents 4 and 10 leave the workspace, the remaining agents of Σ enter a reconfiguration phase before resuming their deployment in a Chebyshev configuration following Algorithm 5.8. For the reconfiguration, the weights to compute the safe objective $\mathbf{y}_i^{\mathrm{s}}(k)$ for all $i \in \overline{1, N} \setminus \mathrm{O}(k)$ are chosen such that $\omega_h = 5$, $\omega_l = 1$ and $\omega_e = 0$. The contraction factors for the computation of the contracted Voronoi cells $\mathcal{V}_i(k, \lambda_{\mathcal{V}_i})$ is $\lambda_{\mathcal{V}_i} = 0.5$ and the contraction factor for the computation of the contracted working regions $\mathcal{W}_i(k, \lambda_{\mathcal{W}_i})$ is $\lambda_{\mathcal{W}_i} = 0.8$ for all $i \in \overline{1, N} \setminus \mathrm{O}(k)$. The way the safe objectives are computed is not described here since it is not different from what is presented in Paragraph 5.3.2.1.

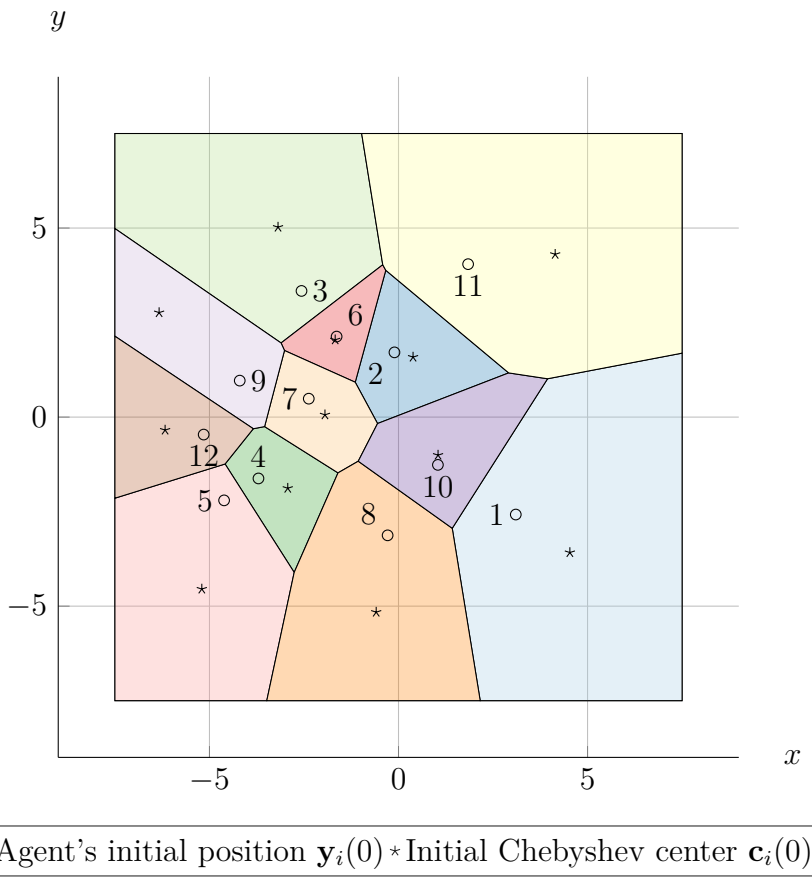


Figure 5.23: Initial configuration of the MAS Σ at $t = 0\,\mathrm{s}$.

Figure 5.23, Figure 5.24 and Figure 5.25 show the configuration of Σ at different time instants. The initial configuration at $t = 0\,\mathrm{s}$ is displayed in Figure 5.23, while the final configuration at $t = 40\,\mathrm{s}$ is presented in Figure 5.25. Then, Figure 5.24 illustrates the configuration at time $t = 5.4\,\mathrm{s}$, while agents 4 and 10 are leaving the workspace $\mathcal{Y}$. In these three figures, the position $\mathbf{y}_i(k)$, with $i \in \overline{1, N}$, of the agents is represented by a circle while the Chebyshev centers $\mathbf{c}_i$ are represented by stars and the safe objectives $\mathbf{y}_i^{\mathrm{s}}$ by diamonds. Moreover, in Figure 5.24 are shown as solid lines the hyperplanes $\partial\mathcal{H}_4(27)$ and $\partial\mathcal{H}_{10}(27)$ defined by the outgoing agents 4 and 10 and their objectives $\mathbf{y}_4^{\mathrm{O}}$ and $\mathbf{y}_{10}^{\mathrm{O}}$ as well as the hyperplanes $\partial\mathcal{O}_4(27)$ and $\partial\mathcal{O}_{10}(27)$, respectively orthogonal to $\partial\mathcal{H}_4(27)$ and $\partial\mathcal{H}_{10}(27)$, allowing to obtain graphically the sets $\mathrm{O}_i(27)$ for all $i \in \overline{1, N} \setminus \mathrm{O}(27)$. Since the construction of these hyperplanes is based on the estimated position $\widehat{\mathbf{y}}_i(k)$, with $i \in \mathrm{O}(k)$, of the outgoing agents, the estimated positions of all the agents are shown in Figure 5.24 with squares. The estimated positions are not presented in Figure 5.23 and Figure 5.25 since $\widehat{\mathbf{y}}_i(0) = \mathbf{y}_i(0)$ for all $i \in \overline{1, N}$ and, according to Figure 5.27, $\widehat{\mathbf{y}}_i(200) \approx \mathbf{y}_i(200)$. From Figure 5.24, it can be seen that the agents $i \in \{2, 3, 11\}$ are in front of the outgoing agents 4 and 10, that the agents $i \in \{1, 6, 9\}$ are in front of agent 10 but behind agent 4 while agents $i \in \{5, 7, 8, 12\}$ are behind both outgoing agents. Then $\mathrm{O}_i(27) = \{4, 10\}$ for $i \in \{2, 3, 11\}$, $\mathrm{O}_i(27) = \{10\}$ for $i \in \{1, 6, 9\}$ and $\mathrm{O}_i(27) = \emptyset$ for $i \in \{5, 7, 8, 12\}$. These sets are used to compute the safe objectives $\mathbf{y}_i^{\mathrm{s}}(27)$ of agents 1, 2, 3, 6, 9 and 11 as presented extensively before.

In Figure 5.24, it can be seen that at $t = 5.4\,\mathrm{s}$, agent 2 and agent 4 are close.

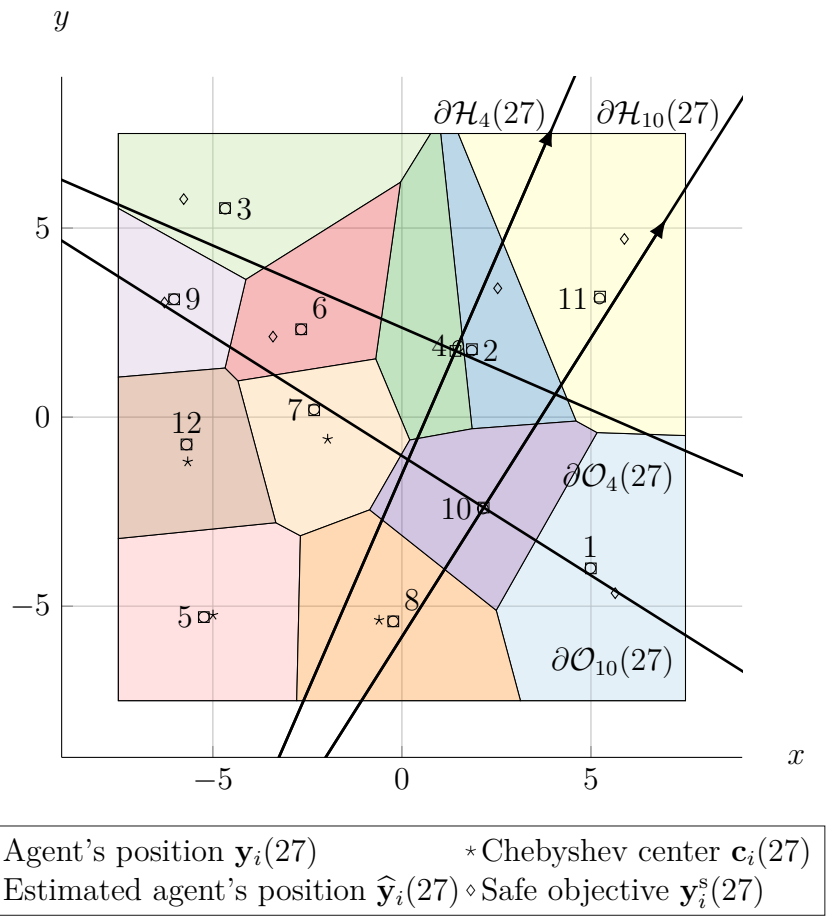


Figure 5.24: Configuration of the MAS Σ at $t = 5.4\,\mathrm{s}$.

This is due to the fact that the contracted working region $\mathcal{W}_2(27, \lambda_{\mathcal{W}_2})$ is too wide, allowing for the safe objective to be too close to the trajectory of the outgoing agents. Indeed, this region is obtained by contracting the region in between the hyperplanes $\partial\mathcal{H}_4(27)$ and $\partial\mathcal{H}_{10}(27)$ which is tight. Then, a contraction factor $\lambda_{\mathcal{W}_2} = 0.8$ is not really effective in reducing the size of the area in which the safe objective $\mathbf{y}_2^{\mathrm{s}}(27)$ can be found. To obtain a safe objective $\mathbf{y}_2^{\mathrm{s}}(27)$ farther from $\partial\mathcal{H}_4(27)$ that what is obtained in Figure 5.24, a smaller contraction factor $\lambda_{\mathcal{W}_2}$ has to be chosen.

Finally, Figure 5.26 presents the distances:

$$d_i(k) = \begin{cases} \|\mathbf{y}_i(k) - \mathbf{y}_i^{\mathrm{s}}(k)\|_2 & \text{if } \mathrm{O}_i(k) \neq \emptyset \\ \|\mathbf{y}_i(k) - \mathbf{c}_i(k)\|_2 & \text{otherwise} \end{cases}$$

for all $i \in \overline{1, N} \setminus \{4, 10\}$. The distances of agents 4 and 10 to their objective points are not shown since the behavior of the agents is similar to what is achieved for the deployment of a multi-UAV system as shown in Section 3.3.3. For its part, Figure 5.27, shows the norm of the estimation error:

$$\varepsilon_i(k) = \|\mathbf{y}_i(k) - \widehat{\mathbf{y}}_i(k)\|_2$$

for all $i \in \overline{1, N}$.

A behavior similar to the one seen in the single integrator dynamics case is observed. When agent 4 leaves Σ at $t = 4\,\mathrm{s}$, the change of objective for all the agents

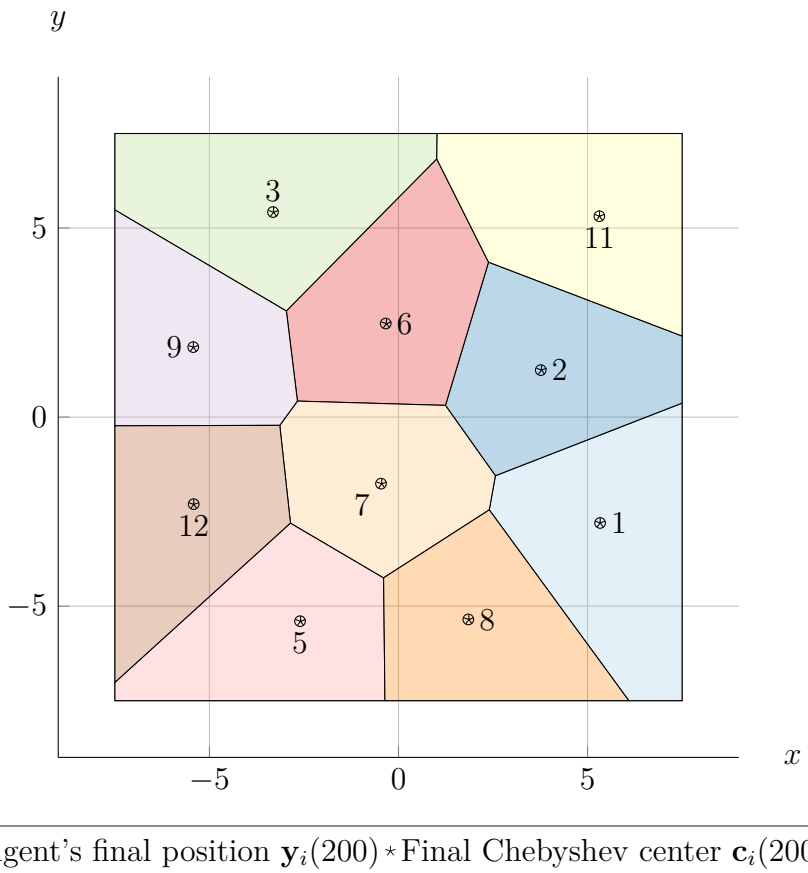


Figure 5.25: Final configuration of the MAS Σ at $t = 40\,\mathrm{s}$.

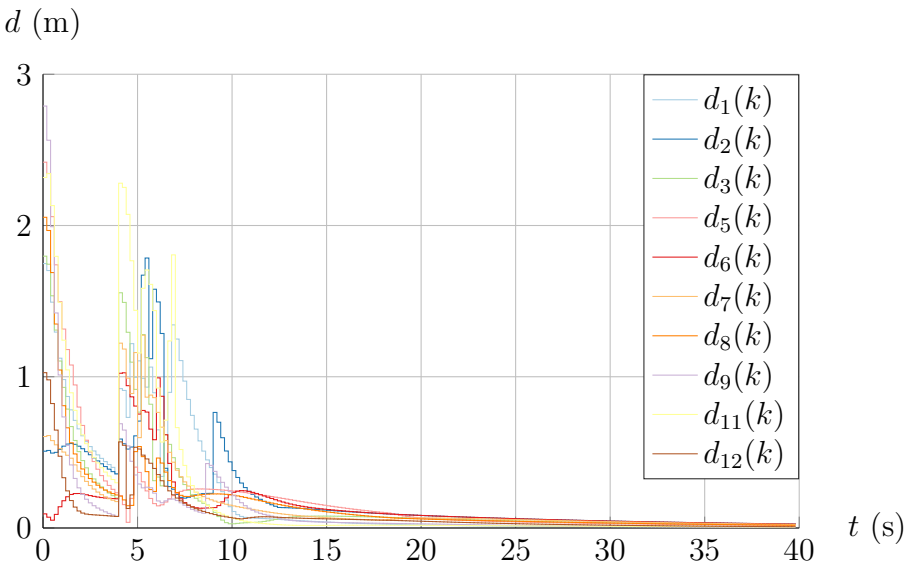


Figure 5.26: Distance of each agent of Σ to its Chebyshev center or safe objective over time.

of Σ, except the ones behind agent 4, is seen in Figure 5.26 with a sudden change in the distance between the agents and their objectives. Then, the agents start rallying their safe objectives until agent 10 leaves Σ at $t = 5.2\,\mathrm{s}$. While agents 4 and 10 leave the workspace $\mathcal{Y}$, the agents of Σ either track their safe objective or resume

tracking the Chebyshev center of their Voronoi cell. When both the outgoing agents are outside $\mathcal{Y}$, all the agents have resumed deploying into a Chebyshev configuration which they are able to reach since the distance of the agents to their Chebyshev centers converges to 0 m. The norm $\varepsilon_i(k)$, with $i \in \overline{1, N}$, displayed in Figure 5.27 shows that the estimation error remains small except for the two outgoing agents when they start rallying their objectives $\mathbf{y}_4^{\mathrm{O}}$ and $\mathbf{y}_{10}^{\mathrm{O}}$. While the values of $\varepsilon_4(k)$ and $\varepsilon_{10}(k)$ remain reasonably small, the sudden change is due to the attitude and speed of the outgoing agents which make the approximation of the nonlinear model of a UAV (3.27) by a linear model approaching its limits.

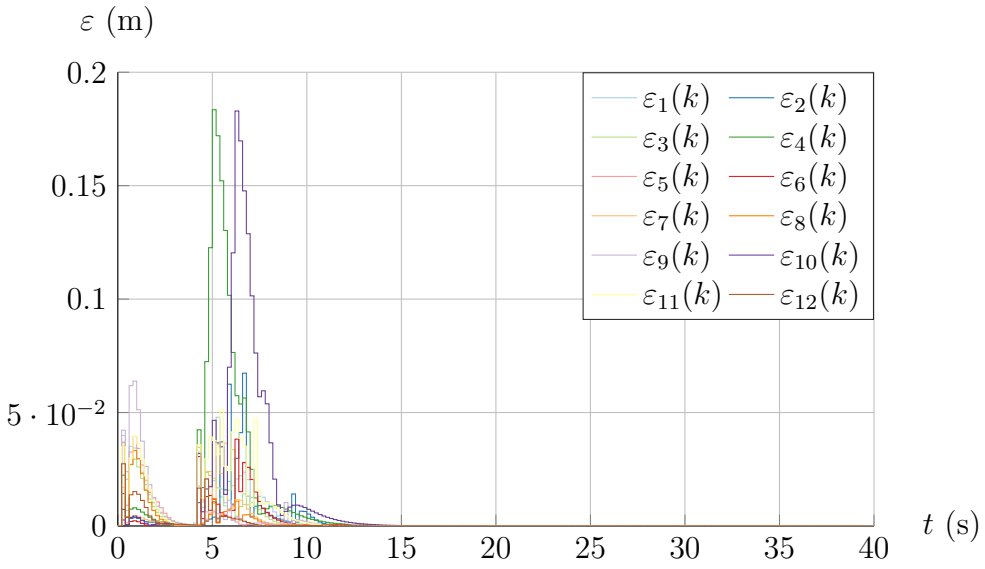


Figure 5.27: Norm of the difference between the estimation position and the real position of each agent of Σ over time.

5.4 Conclusion

This chapter deals with the reconfiguration of a multi-agent system (MAS) performing a Voronoi-based deployment in a two-dimensional convex bounded area when the number of agents in the MAS changes. Two reconfiguration strategies are presented, one when the number of agents increases because agents join the MAS, and the other when the number of agents decreases because agents need to leave the MAS.

The reconfiguration strategy in the case of incoming agents is a natural extension of the deployment strategy of Chapter 3 and allows to integrate agents seamlessly to the MAS. The agents that integrate the MAS then participate in its deployment into a static Chebyshev configuration. This strategy can be further improved by combining it with the robust deployment strategies of Chapter 4.

For its part, the reconfiguration strategy in the case of outgoing agents is more complex. It stems from problems arising in fault-tolerant formation control and is meant to allow agents to leave safely the workspace in which the MAS is deployed without colliding with the agents remaining in the workspace. The strategies presented in this chapter are based on the use of an objective different from the Chebyshev center for the agents remaining in the workspace, and allowing them

to avoid the trajectories of the outgoing agents. This objective is calculated as a weighted barycenter of the remaining agents. A first strategy able to deal with the case of a single outgoing agent is introduced. However, this strategy is limited in the case of multiple outgoing agents. Another reconfiguration algorithm is then proposed. It is based on a new transient objective, again computed as a weighted barycenter, but constrained to remain inside a region obtained as the intersection of a contracted version of the Voronoi cell of the agent and a contracted version of the area contained between the trajectories of the outgoing agents, called here working region. When there is no risk of collision, the remaining agents of the MAS resume tracking their Chebyshev center in order to deploy into a static Chebyshev configuration.

Despite being only mentioned in this thesis, the strategy is easily adaptable in the case of a fault occurring on one of the agents. An example of use of this reconfiguration when the outgoing agents are faulty can be found in Chevet et al. (2020b). To cover a wider spectrum of faults on the agents, the reconfiguration strategy proposed in this chapter could be combined with the robust deployment strategies proposed in Chapter 4. Moreover, the examples presented in this thesis always assume contraction factors for the construction of the reconfiguration objectives to be identical for all the agents. This can lead to dangerous situations, as presented in the UAV reconfiguration example of Paragraph 5.3.2.2, where the region to which the reconfiguration objective belongs to is not tight enough and allows for an objective to be too close to the trajectory of an outgoing agent. This kind of situation could be avoided by developing an efficient strategy to adapt the value of the contraction factors depending on the relative position of the agents remaining in the workspace to the trajectories of the outgoing agents.

Chapter 6

Concluding remarks and future work

6.1 Conclusion

The present thesis introduces several deployment and reconfiguration strategies for a multi-agent system (MAS). The proposed control strategies are based on decentralized model predictive control (MPC) laws and lead the MAS into a static configuration over a convex bounded two-dimensional area. The configuration is such that each agent lies on the Chebyshev center of a Voronoi cell of which it is the generator. Then, the contributions of this thesis are threefold. It first studies the deployment of a MAS in the nominal case and applies it to a fleet of quadrotor unmanned aerial vehicles (UAVs). It secondly studies the deployment of a MAS when the agents are subject to perturbations, either bounded deterministic or unbounded stochastic perturbations. Finally, reconfiguration strategies are proposed to deal with the case of agents joining or leaving the MAS.

Based on the work of Nguyen (2016), a centralized and a decentralized MPC algorithms are proposed in Chapter 3 to allow a MAS to deploy into a two-dimensional convex bounded area. Each agent is constrained in the MPC optimization problem to remain inside a Voronoi cell, of which it is the generator, and tracks the Chebyshev center of this Voronoi cell. For the case of agents obeying single integrator dynamics, a proof of feasibility of the decentralized MPC optimization problem is proposed, together with the convergence of the MAS deployment into a static Chebyshev configuration.

To deal with the case of a fleet of quadrotor UAVs, a cascaded control structure where the decentralized MPC is used to control the position of the UAVs is proposed. Simulations results showing the efficiency of the deployment when the UAVs are modeled as nonlinear systems are presented.

However, in real world applications, the agents are often subject to perturbations, either external (caused by the environment they evolve in), or internal (caused by faults or noises). With this in mind, a decentralized output-feedback tube-based MPC algorithm is designed for the deployment of a MAS subject to input and output bounded deterministic perturbations. In this framework, the position of the agents is not punctiform anymore but contained inside a box. Thus, a new box-based guaranteed Voronoi tessellation is then proposed in Chapter 2 to ensure that the agents are constrained to evolve inside an area in which there is no risk of collision with the other agents. To drive the agents in the guaranteed Voronoi tessellation, the proposed tube-based MPC strategy decomposes the computation of the control input into two parts. The first one is obtained by solving a nominal MPC optimization problem that neglects the bounded perturbations. The second one is obtained by multiplying the difference between the estimated state and the unperturbed nominal state of the system by a gain matrix. The state-feedback is designed to keep the

actual state within a tube envelope around the nominal state and ensures that the actual state deviation is bounded in the presence of uncertain dynamics with bounded deterministic perturbations. In addition, to obtain the estimated state, an observer is introduced in the control loop. For these reasons, in Chapter 4, two linear/bilinear matrix inequality constrained optimization problems are proposed in order to obtain the gains of a Luenberger observer and of the state-feedback part of the tube-based MPC controller.

In some cases (i.e. systems subject to process and measurement noises), the perturbations acting on a system can be unbounded and have a stochastic nature. Thus, Chapter 4 introduces an output-feedback chance-constrained MPC algorithm for the Voronoi-based deployment of a MAS subject to unbounded stochastic perturbations. Indeed, in this context, it is necessary to enforce that the constraints appearing in the MPC optimization problem are satisfied with a given probability. Then, using the stochastic properties of these signals, an optimization-based procedure is proposed to relax the probabilistic constraints into algebraic constraints. A proof of feasibility of this procedure is provided as well as an explicit solution to the optimization problem, allowing the online relaxation of constraints. With the proposed procedure, the controller is able to deal with time-varying stochastic properties of the perturbation signals.

For both the bounded deterministic and unbounded stochastic perturbations cases, the efficiency of the deployment strategies are illustrated on two types of multi-agent systems. To do so, two MAS are considered for each strategy. A first multi-agent system is composed of agents obeying single integrator dynamics, while the other is composed of quadrotor UAVs modeled with nonlinear dynamics. In the case of a fleet of quadrotor UAVs, the robust model predictive controller is used as the position controller of the agents.

For operational reasons, the MAS may need to incorporate new agents to participate in its deployment. Then, Chapter 5 proposes a natural extension of the decentralized deployment algorithm of Chapter 3 to allow new agents to join the MAS and to deploy over the convex bounded two-dimensional area. However, while the MAS may need to include new agents, the converse is also true, and a reconfiguration strategy is needed to allow agents to leave the multi-agent system and the deployment area while the remaining agents avoid colliding with them.

A first reconfiguration strategy is introduced to deal with the case of a single outgoing agent. While this agent leaves the deployment area using a decentralized model predictive controller, the remaining agents track a transient objective allowing them to be driven away from the trajectory of the outgoing agent. This transient objective is the weighted barycenter of the neighbors of an agent. If a neighbor of a remaining agent is farther away from the outgoing agent's trajectory than the remaining agent, it receives a higher weight than the other neighbors. A second reconfiguration strategy is then presented to deal with the case of several agents leaving the MAS simultaneously. With this new strategy, each remaining agent decides if it continues tracking the Chebyshev center of its Voronoi cell or if it tracks a new transient objective depending on its relative position to both the outgoing agents and their objectives outside the deployment area. The new transient objective is then obtained by solving an optimization problem, constraining it to belong to an area in which it is ensured that no risk of collision with the outgoing agents exists for the remaining agents. These two reconfiguration strategies are compared in simulation for the case of one agent leaving a MAS composed of agents obeying single

integrator dynamics. Then, the efficiency of the second reconfiguration strategy in the case of several agents leaving the deployment area simultaneously is illustrated.

6.2 Future directions

All along this thesis, directions opened by the proposed work have been discussed. A synthetic view of these perspectives is presented here.

The Voronoi-based deployment and reconfiguration strategies introduced throughout this thesis are two-dimensional since they are meant to be applied in any kind of multi-agent system composed of vehicles, some of them, as unmanned ground or surface vehicles, which cannot move in more than two dimensions. However, unmanned aerial or underwater vehicles are able to move in the three directions of space. Then, a generalization of the proposed control strategies for the Voronoi-based deployment and reconfiguration of multi-agent systems in a convex bounded three-dimensional area could be studied. Such a generalization would cover both the nominal and perturbed case. Moreover, an extension of the reconfiguration strategies in the case of outgoing agents introduced in Chapter 5 would be needed. Indeed, the working regions of the remaining agents are based on the trajectories of the outgoing agents. However, in three dimensions, an infinite number of hyperplanes contain this trajectory. Thus, an efficient scheme should be introduced to select the appropriate hyperplanes to define the remaining agents' working regions.

To generalize the deployment strategy in the nominal case, following the discussion at the end of Chapter 3, the recursive feasibility and stability of model predictive control under time-varying constraints has to be further investigated. To do so, it may be necessary to adopt a centralized or distributed control strategy instead of a decentralized one. Then, to provide more degrees of freedom to the optimization problem on which the model predictive controller is based, the pseudo-Voronoi tessellation proposed in Chapter 2 could be used. With this new tessellation, the border of a cell generated by two agents passes by a point located on the line segment joining these two agents. This location would be added as a decision variable to the problem.

When dealing with systems subject to bounded deterministic perturbations, robust control strategies are used. In the present thesis, a Luenberger observer is used to estimate the state of the agents of the MAS. Future work on the deployment of a multi-agent system subject to bounded deterministic perturbations should study the use of a more robust observer such as Kalman filters or set-membership state estimation techniques. The latter method could be combined with the linear/bilinear matrix inequality constrained gain tuning procedure proposed in Chapter 4 to compute the gains necessary for the decentralized output-feedback tube-based MPC as well as the associated invariant sets online. Indeed, the MPC optimization problems are subject to time-varying constraints which influence the shape and size of the invariant sets. Thus, computing them online would allow to take this time dependence into account. An analysis of the computation time of such a strategy is necessary.

The aforementioned gain tuning procedure for the output-feedback tube-based model predictive controller consists in solving two linear/bilinear matrix inequality constrained optimization problems sequentially. Further studies of a common tuning

procedure, where both gains are obtained from a single polynomial matrix inequality constrained optimization problem, should be carried out.

Since the output-feedback decentralized chance-constrained MPC strategy introduced in Chapter 4 is based, as the tube-based MPC strategy, on the estimation of the state with a Luenberger observer, the same future direction concerning the use of more robust observers holds. Moreover, the unbounded stochastic perturbations considered in this case are normally distributed. Thus, the adaptation of the chance-constrained strategy to other probability distributions should be investigated. Such investigations would study the possibility to use the stochastic properties of the distributions to relax the probabilistic constraints appearing in the MPC optimization problem into algebraic constraints.

Another direction for the deployment of a MAS subject to unbounded stochastic perturbations is to use tube-based chance-constrained MPC techniques. Indeed, the algebraic relaxation procedure of Chapter 4 is meant to find a probabilistic bound to the perturbations. Then, a bounded set containing a fraction (ideally close to 1) of the unbounded stochastic perturbations could be defined. Thus, the tube-based MPC strategy introduced for bounded perturbations could be used for the deployment of a MAS subject to unbounded stochastic perturbations.

The reconfiguration strategies of Chapter 5 have been introduced for nominal agents. Then, considering perturbations, either bounded deterministic or unbounded stochastic, the robust control strategies of Chapter 4 could be used for the reconfiguration of a MAS subject to perturbations. Moreover, the deployment and reconfiguration strategy in the case of multiple outgoing agents is based on the definition of safe operating regions for the remaining agents. These safe regions are obtained by contracting areas to which a remaining agent belongs. However, in the strategy proposed in Chapter 5, the contraction factors are chosen arbitrarily by the user. Thus, further improvements of this algorithm should include an efficient scheme to adapt the value of the contraction factors depending on the relative position of the remaining agents to the outgoing agents.

In the present thesis, the proposed deployment and reconfiguration strategies are tested in simulation. Thus, future work should focus on the implementation of the control algorithms on real multi-vehicle systems.

Bibliography

Mahyar Abdolhosseini, Youmin Zhang, and Camille Alain Rabbath. An efficient model predictive control scheme for an unmanned quadrotor helicopter. *Journal of Intelligent & Robotic Systems*, 70(1-4):27–38, 2013. 83

Saeed Ahmadizadeh, Dragan Nešić, Dean R. Freestone, and David B. Grayden. On synchronization of networks of Wilson–Cowan oscillators with diffusive coupling. *Automatica*, 71:169–178, 2016. 7

Javier Alonso Mora, Tobias Nägeli, Roland Siegwart, and Paul Beardsley. Collision avoidance for aerial vehicles in multi-agent scenarios. *Autonomous Robots*, 39(1):101–121, 2015. xxvi, 3

Ignacio Alvarado. *Model predictive control for tracking constrained linear systems*. Ph.D. thesis, Universidad de Sevilla, 2007. xxxi, 14, 104, 145

Ignacio Alvarado, Daniel Limón, Teodoro Álamo, Mirko Fiacchini, and Eduardo F. Camacho. Robust tube based MPC for tracking of piece-wise constant references. In *46th IEEE Conference on Decision and Control*, pages 1820–1825. IEEE, 2007. 98

Arslan Ahmed Amin and Khalid Mahmood Hasan. A review of fault tolerant control systems: advancements and applications. *Measurement*, 143:58–68, 2019. 148

Franco Angelini, Cosimo Della Santina, Manolo Garabini, Matteo Bianchi, Gian Maria Gasparri, Giorgio Grioli, Manuel Giuseppe Catalano, and Antonio Bicchi. Decentralized trajectory tracking control for soft robots interacting with the environment. *IEEE Transactions on Robotics*, 34(4):924–935, 2018. xxvii, 5

Franz Aurenhammer. Voronoi diagrams–a survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345–405, 1991. xxix, 9

Muharrem Ayar, Rodrigo D. Trevizan, Arturo S. Bretas, Haniph Latchman, and Serhat Obuz. A robust decentralized control framework for enhancing smart grid transient stability. In *2017 IEEE Power & Energy Society General Meeting*, pages 1–5. IEEE, 2017. xxvii, 5

Valentin Baillard, Alexandre Goy, Nicolas Vasselin, and Cristina Stoica Maniu. Potential field based optimization of a prey-predator multi-agent system. In *9th Vienna Conference on Mathematical Modelling*, 2018. xxviii, 7

Efstathios Bakolas and Panagiotis Tsiotras. Optimal partitioning for spatiotemporal coverage in a drift field. *Automatica*, 49(7):2064–2073, 2013. 11

Giuseppe Basile and Giovanni Marro. Controlled and conditioned invariant subspaces in linear system theory. *Journal of Optimization Theory and Applications*, 3(5):306–315, 1969. 35

Adel Belkadi, Hernan Abaunza, Laurent Ciarletta, Pedro Castillo, and Didier Theilliol. Distributed path planning for controlling a fleet of UAVs: Application to a team of quadrotors. *IFAC-PapersOnLine*, 50(1):15983–15989, 2017. 6

Adel Belkadi, Hernan Abaunza, Laurent Ciarletta, Pedro Castillo, and Didier Theilliol. Design and implementation of distributed path planning algorithm for a fleet of UAVs. *IEEE Transactions on Aerospace and Electronic Systems*, 55(6):2647–2657, 2019. 6

Fabio L. Bellifemine, Giovanni Caire, and Dominic P.A. Greenwood. *Developing Multi-Agent Systems with JADE*, volume of *Wiley Series in Agent Technology*. John Wiley & Sons, 2007. xxvi, 2

Alberto Bemporad, Manfred Morari, Vivek Dua, and Efstratios N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002. xxix, 8

Ali Bidram, Frank L. Lewis, and Ali Davoudi. Distributed control systems for small-scale power networks: Using multiagent cooperative control theory. *IEEE Control Systems Magazine*, 34(6):56–77, 2014. 6

Lars Blackmore, Masahiro Ono, and Brian C. Williams. Chance-constrained optimal path planning with obstacles. *IEEE Transactions on Robotics*, 27(6): 1080–1094, 2011. 123, 124

Franco Blanchini and Stefano Miani. *Set-Theoretic Methods in Control*, volume of *Systems & Control: Foundations & Applications*. Birkhäuser, 2nd edition, 2015. 13, 19, 23, 24, 35, 36, 98

Mogens Blanke, Christian W. Frei, Franta Kraus, Ron J. Patton, and Marcel Staroswiecki. What is fault-tolerant control? *IFAC Proceedings Volumes*, 33 (11):41–52, 2000. 148

Mogens Blanke, Michel Kinnaert, Jan Lunze, Marcel Staroswiecki, and Jochen Schröder. *Diagnosis and Fault-Tolerant Control*. Springer, 3rd edition, 2016. 149

Samir Bouabdallah, Pierpaolo Murrieri, and Roland Siegwart. Design and control of an indoor micro quadrotor. In *IEEE International Conference on Robotics and Automation*, volume 5, pages 4393–4398. IEEE, 2004. 78

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 7th edition, 2009. 8, 11, 23, 53, 54, 62, 134

Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*, volume 15 of *SIAM Studies in Applied Mathematics*. SIAM, 1994. 20, 21, 22, 23, 103

David Bremner, Komei Fukuda, and Ambros Marzetta. Primal-dual methods for vertex and facet enumeration. *Discrete & Computational Geometry*, 20(3): 333–357, 1998. 28

Efim M. Bronstein. Approximation of convex sets by polytopes. *Journal of Mathematical Sciences*, 153(6):727–762, 2008. 25

Francesco Bullo, Jorge Cortés, and Sonia Martínez. *Distributed Control of Robotic Networks: A Mathematical Approach to Motion Coordination Algorithms*, volume 27 of *Princeton Series in Applied Mathematics*. Princeton University Press, 2009. xxvi, 2, 3

Davide Calvaresi, Amro Najjar, Michael Schumacher, and Kary Främling. *Explainable, Transparent Autonomous Agents and Multi-Agent Systems*, volume 11763 of *Lecture Notes in Computer Science*. Springer, 2019. xxvi, 2

Eduardo F. Camacho and Carlos Bordons. *Model Predictive Control*, volume of *Advanced Textbooks in Control and Signal Processing*. Springer, 2nd edition, 2007. xxix, 8

Mark Cannon, Basil Kouvaritakis, Saša V. Raković, and Qifeng Cheng. Stochastic tubes in model predictive control with probabilistic constraints. *IEEE Transactions on Automatic Control*, 56(1):194–200, 2010. xxix, 8, 123

Mark Cannon, Qifeng Cheng, Basil Kouvaritakis, and Saša V. Raković. Stochastic tube MPC with state estimation. *Automatica*, 48(3):536–541, 2012. xxix, 8, 13, 98, 123, 129, 146

Yongcan Cao and Wei Ren. Distributed coordinated tracking with reduced interaction via a variable structure approach. *IEEE Transactions on Automatic Control*, 57(1):33–48, 2011. 6

Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial informatics*, 9(1):427–438, 2012. xxviii, 6

Ashwin Carvalho, Yiqi Gao, Stéphanie Lefèvre, and Francesco Borrelli. Stochastic predictive control of autonomous vehicles in uncertain environments. In *12th International Symposium on Advanced Vehicle Control*, pages 712–719, 2014. 124

Pedro Castillo, Rogelio Lozano, and Alejandro E. Dzul. *Modelling and Control of Mini-Flying Machines*, volume of *Advances in Industrial Control*. Springer, 2005. 78

Pavel Čech, Petr Tučník, Vladimír Bureš, and Martina Husráková. Modelling complexity of economic system with multi-agent systems. In *5th International Conference on Knowledge Management and Information Sharing*, pages 464–469, 2013. xxvi, 2

Abbas Chamseddine, Youmin Zhang, and Camille Alain Rabbath. Trajectory planning and re-planning for fault tolerant formation flight control of quadrotor unmanned aerial vehicles. In *American Control Conference*, pages 3291–3296. IEEE, 2012. 149

Paula Chanfreut, José M. Maestre, and Eduardo F. Camacho. Coalitional model predictive control on freeways traffic networks. *IEEE Transactions on Intelligent Transportation Systems*, 2020. xxvi, xxviii, 3, 6, 8

Nesrine Changuel, Bessem Sayadi, and Michel Kieffer. Control of multiple remote servers for quality-fair delivery of multimedia contents. *IEEE Journal on Selected Areas in Communications*, 32(4):746–759, 2014. xxvii, 4

Gong Cheng, Jason Gu, Tao Bai, and Osama Majdalawieh. A new efficient control algorithm using potential field: extension to robot path tracking. In *Canadian Conference on Electrical and Computer Engineering*, volume 4, pages 2035–2040. IEEE, 2004. 7

Thomas Chevet, Maria Makarov, Cristina Stoica Maniu, Israel Hinostroza, and Pierre Tarascon. State estimation of an octorotor with unknown inputs. Application to radar imaging. In *21st International Conference on System Theory, Control and Computing*, pages 723–728. IEEE, 2017. 145

Thomas Chevet, Cristina Stoica Maniu, Cristina Vlad, and Youmin Zhang. Voronoi-based UAVs formation deployment and reconfiguration using MPC techniques. In *International Conference on Unmanned Aircraft Systems*, pages 9–14. IEEE, 2018. xxxi, xxxiii, 14, 15, 67, 148, 150, 155

Thomas Chevet, Cristina Stoica Maniu, Cristina Vlad, and Youmin Zhang. Guaranteed Voronoi-based deployment for multi-agent systems under uncertain measurements. In *18th European Control Conference*, pages 4016–4021. IEEE, 2019. xxxi, 14, 44, 98, 99, 107

Thomas Chevet, Cristina Stoica Maniu, Cristina Vlad, Youmin Zhang, and Eduardo F. Camacho. Chance-constrained MPC for Voronoi-based multi-agent system deployment. In *IFAC World Congress*, pages 7051–7056. IFAC, 2020a. xxxii, 15, 114, 123, 124, 126, 130

Thomas Chevet, Cristina Vlad, Cristina Stoica Maniu, and Youmin Zhang. Decentralized MPC for UAVs formation deployment and reconfiguration with multiple outgoing agents. *Journal of Intelligent & Robotic Systems*, 97(1):155–170, 2020b. xxxi, xxxiii, 14, 15, 39, 67, 79, 80, 148, 150, 166, 170, 192

Howie Choset. Coverage for robotics – A survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31(1-4):113–126, 2001. xxix, 9, 62

Howie Choset and Joel Burdick. Sensor based planning, Part I: The generalized Voronoi graph. In *IEEE International Conference on Robotics and Automation*, pages 1649–1655. IEEE, 1995. xxix, 9, 44

Charles K. Chui and Guanrong Chen. *Kalman Filtering*. Springer, 5th edition, 2017. 145

Alfredo Colosimo. Multi-agent simulations of population behavior: A promising tool for systems biology. In Mariano Bizzarri, editor, *Systems Biology*, volume 1702 of *Methods in Molecular Biology*, pages 307–326. Springer, 2018. xxvi, 2

Stefano R. Cominesi, Marcello Farina, Luca Giulioni, Bruno Picasso, and Riccardo Scattolini. A two-layer stochastic model predictive control scheme for microgrids. *IEEE Transactions on Control Systems Technology*, 26(1):1–13, 2017. 6

Luca Consolini, Fabio Morbidi, Domenico Prattichizzo, and Mario Tosques. Leader–follower formation control of nonholonomic mobile robots with input constraints. *Automatica*, 44(5):1343–1349, 2008. 5

Jorge Cortés, Sonia Martínez, Timur Karatas, and Francesco Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255, 2004. xxvi, xxix, xxx, 3, 9, 10, 62, 63

Iain D. Couzin, Jens Krause, Nigel R. Franks, and Simon A. Levin. Effective leadership and decision-making in animal groups on the move. *Nature*, 433(7025): 513–516, 2005. xxvi, 2

Li Dai, Yuanqing Xia, Yulong Gao, Basil Kouvaritakis, and Mark Cannon. Cooperative distributed stochastic MPC for systems with state estimation and coupled probabilistic constraints. *Automatica*, 61:89–96, 2015. 13, 124, 146

Li Dai, Yuanqing Xia, Yulong Gao, and Mark Cannon. Distributed stochastic MPC of linear systems with additive uncertainty and coupled probabilistic constraints. *IEEE Transactions on Automatic Control*, 62(7):3474–3481, 2016. 8, 124

Li Dai, Yuanqing Xia, Yulong Gao, and Mark Cannon. Distributed stochastic MPC for systems with parameter uncertainty and disturbances. *International Journal of Robust and Nonlinear Control*, 28(6):2424–2441, 2018. 13, 124, 146

Raffaello D'Andrea and Geir E. Dullerud. Distributed control design for spatially interconnected systems. *IEEE Transactions on Automatic Control*, 48(9): 1478–1495, 2003. xxvi, 3

George B. Dantzig. Fourier-motzkin elimination and its dual. Technical report, Stanford University, 1972. 30

Paul Davidsson. Agent based social simulation: A computer science view. *Journal of artificial societies and social simulation*, 5(1), 2002. xxvi, 2

Boris Delaunay. Sur la sphère vide. *Bulletin de l'Académie des sciences de l'URSS*, 6:793–800, 1934. 51

Scott A. DeLoach, Mark F. Wood, and Clint H. Sparkman. Multiagent systems engineering. *International Journal of Software Engineering and Knowledge Engineering*, 11(03):231–258, 2001. xxvi, 2

Emmanuele DiBenedetto. *Real Analysis*, volume of *Birkhäuser Advanced Texts Basler Lehrbücher*. Springer, 2nd edition, 2016. 56, 121

Dimos V. Dimarogonas, Emilio Frazzoli, and Karl H. Johansson. Distributed event-triggered control for multi-agent systems. *IEEE Transactions on Automatic Control*, 57(5):1291–1297, 2011. xxvi, 2

Aris L. Dimeas and Nikos D. Hatziargyriou. Operation of a multiagent system for microgrid control. *IEEE Transactions on Power Systems*, 20(3):1447–1455, 2005. xxvi, 3

G. Lejeune Dirichlet. Über die Reduction der positiven quadratischen Formen mit drei unbestimmten ganzen Zahlen. *Journal für die reine und angewandte Mathematik*, 40:209–227, 1850. xxix, 9, 41

Sebastian d'Oleire Oltmanns, Irene Marzolff, Klaus Daniel Peter, and Johannes B. Ries. Unmanned aerial vehicle (UAV) for monitoring soil erosion in Morocco. *Remote Sensing*, 4(11):3390–3416, 2012. xxix, 9, 61

Sergey V. Drakunov and Vadim I. Utkin. Sliding mode control in dynamic systems. *International Journal of Control*, 55(4):1029–1037, 1992. 98

S.S. Dughman and John Anthony Rossiter. A survey of guaranteeing feasibility and stability in MPC during target changes. *IFAC-PapersOnLine*, 48(8):813–818, 2015. 92, 93

W. Palin Elderton. Tables for testing the goodness of fit of theory to observation. *Biometrika*, 1(2):155–163, 1902. 59, 130, 136

William Evans and Jeff Sember. Guaranteed Voronoi diagrams of uncertain sites. In *20th Canadian Conference on Computational Geometry*, pages 207–210, 2008. 44

Marcello Farina, Luca Giulioni, and Riccardo Scattolini. Stochastic linear Model Predictive Control with chance constraints – A review. *Journal of Process Control*, 44:53–67, 2016. 13, 123, 124, 130

J. Alexander Fax and Richard M. Murray. Information flow and cooperative control of vehicle formations. *IEEE Transactions on Automatic Control*, 49(9): 1465–1476, 2004. 7

Filiberto Fele, José M. Maestre, S. Mehdy Hashemy, David Muñoz de la Peña, and Eduardo F. Camacho. Coalitional model predictive control of an irrigation canal. *Journal of Process Control*, 24(4):314–325, 2014. xxviii, 8

Filiberto Fele, José M. Maestre, and Eduardo F. Camacho. Coalitional control: Cooperative game theory and control. *IEEE Control Systems Magazine*, 37(1): 53–69, 2017. 8

Filiberto Fele, Ezequiel Debada, José M. Maestre, and Eduardo F. Camacho. Coalitional control for self-organizing agents. *IEEE Transactions on Automatic Control*, 63(9):2883–2897, 2018. xxviii, 8

Jacques Ferber and Olivier Gutknecht. A meta-model for the analysis and design of organizations in multi-agent systems. In *International Conference on Multi Agent Systems*, pages 128–135. IEEE, 1998. xxvi, 2

Mirko Fiacchini and Mazen Alamir. Computing control invariant sets is easy. *arXiv e-prints*, art. arXiv:1708.04797, 2017. 36

Jan Fiala, Michal Kočvara, and Michael Stingl. PENLAB: A MATLAB solver for nonlinear semidefinite optimization. *arXiv e-prints*, art. arXiv:1311.5240, 2013. 21

Pedro Flores Palmeros, Pedro Castillo, and Fernando Castanos. Backstepping-based controller for flight formation. In *International Conference on Unmanned Aircraft Systems*, pages 263–269. IEEE, 2019. xxviii, 7

Bernard Flury. *A First Course in Multivariate Statistics*, volume of *Springer Texts in Statistics*. Springer, 1st edition, 1997. 58, 59

Gene F. Franklin, J. David Powell, and Michael L. Workman. *Digital Control of Dynamic Systems*. Addison-Wesley, 3rd edition, 1997. 120, 121

Komei Fukuda and Alain Prodon. Double description method revisited. In *Franco-Japanese and Franco-Chinese Conference on Combinatorics and Computer Science*, pages 91–111. Springer, 1995. 28

Damien Galzi and Yuri Shtessel. UAV formations control using high order sliding modes. In *American Control Conference*, pages 4249–4254. IEEE, 2006. 63

Yan Gao, Gengyuan Liu, Marco Casazza, Yan Hao, Yan Zhang, and Biagio F. Giannetti. Economy-pollution nexus model of cities at river basin scale based on multi-agent simulation: A conceptual framework. *Ecological Modelling*, 379: 22–38, 2018. xxvi, 2

Yiqi Gao, Andrew Gray, H. Eric Tseng, and Francesco Borrelli. A tube-based robust nonlinear predictive control approach to semiautonomous ground vehicles. *Vehicle System Dynamics*, 52(6):802–823, 2014. 98

Francisco Gavilan, Rafael Vazquez, and Eduardo F. Camacho. Chance-constrained model predictive control for spacecraft rendezvous with disturbance estimation. *Control Engineering Practice*, 20(2):111–122, 2012. xxxi, xxxii, 14, 123, 124, 126, 130, 133, 138, 145

Khaled A. Ghamry and Youmin Zhang. Fault-tolerant cooperative control of multiple UAVs for forest fire detection and tracking mission. In *3rd Conference on Control and Fault-Tolerant Systems*, pages 133–138. IEEE, 2016. 149

Chris Godsil and Gordon F. Royle. *Algebraic Graph Theory*, volume 207 of *Graduate Texts in Mathematics*. Springer, 1st edition, 2001. 7

Ramón González, Mirko Fiacchini, José Luis Guzmán, Teodoro Álamo, and Francisco Rodríguez. Robust tube-based predictive control for mobile robots in off-road conditions. *Robotics and Autonomous Systems*, 59(10):711–726, 2011. 98

Michael Grant and Stephen Boyd. Graph implementations for nonsmooth convex programs. In Vincent D. Blondel, Stephen Boyd, and Hidenori Kimura, editors, *Recent Advances in Learning and Control*, volume 371 of *Lecture Notes in Control and Information Sciences*, pages 95–110. Springer, 2008. 21

Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx, mar 2014. 21

Michael Green and David J.N. Limebeer. *Linear Robust Control*. Courier Corporation, 2012. 98

Daniel Grzonka, Agnieszka Jakobik, Joanna Kołodziej, and Sabri Pllana. Using a multi-agent system and artificial intelligence for monitoring and improving the cloud performance and security. *Future Generation Computer Systems*, 86: 1106–1117, 2018. 2

Ahmed Taimour Hafez and Mohamed A. Kamel. Fault-tolerant control for cooperative unmanned aerial vehicles formation via fuzzy logic. In *International Conference on Unmanned Aircraft Systems*, pages 1261–1266. IEEE, 2016. 149

Johan Hagelbäck and Stefan J. Johansson. Using multi-agent potential fields in real-time strategy games. In *Seventh International Conference on Autonomous Agents and Multi-agent Systems*, pages 631–638, 2008. xxviii, 7

Brian C. Hall. *Lie Groups, Lie Algebras, and Representations: An Elementary Introduction*, volume 222 of *Graduate Texts in Mathematics*. Springer, 2nd edition, 2015. 20

Paul R. Halmos. *Measure Theory*, volume 18 of *Graduate Texts in Mathematics*. Springer, 1st edition, 1950. 56

Jinlu Han and YangQuan Chen. Multiple UAV formations for cooperative source seeking and contour mapping of a radiative signal field. *Journal of Intelligent & Robotic Systems*, 74(1-2):323–332, 2014. xxix, 9, 61

Tianfang Han, Chuntao Zhang, Yan Sun, and Xiaomin Hu. Study on environment-economy-society relationship model of liaohe river basin based on multi-agent simulation. *Ecological modelling*, 359:135–145, 2017. xxvi, 2

Tomoaki Hashimoto. Probabilistic constrained model predictive control for linear discrete-time systems with additive stochastic disturbances. In *52nd Conference on Decision and Control*, pages 6434–6439. IEEE, 2013. 123

Johan Hatleskog. *Voronoi-based deployment for multi-agent systems*. Master's thesis, Norwegian University of Science and Technology (NTNU), 2018. xxix, xxx, 9, 11, 12, 13, 63, 70, 71, 73, 92, 94

Didier Henrion, Johan Löfberg, Michal Kočvara, and Michael Stingl. Solving polynomial static output feedback problems with PENBMI. In *44th IEEE Conference on Decision and Control*, pages 7581–7586. IEEE, 2005. 21

Martin Herceg, Michal Kvasnica, Colin N. Jones, and Manfred Morari. Multi-Parametric Toolbox 3.0. In *12th European Control Conference*, pages 502–510. IEEE, 2013. 54

Christina Herzog, Jean-Marc Pierson, and Laurent Lefèvre. Modelling technology transfer in green IT with multi-agent system. In Rachid Benlamri and Michael Sparer, editors, *Leadership, Innovation and Entrepreneurship as Driving*

Forces of the Global Economy, volume of *Springer Proceedings in Business and Economics*, pages 3–16. Springer, 2017. xxvi, 2

João P. Hespanha, Payam Naghshtabrizi, and Yonggang Xu. A survey of recent results in networked control systems. *Proceedings of the IEEE*, 95(1):138–162, 2007. xxvi, 2

Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 2nd edition, 2013. 20

Zeng-Guang Hou, Long Cheng, and Min Tan. Decentralized robust adaptive control for the multiagent system consensus problem using neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(3): 636–647, 2009. xxix, 8

Andrew Howard, Maja J. Matarić, and Gaurav S. Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In Hajime Asama, Tamio Arai, Toshio Fukuda, and Tsutomu Hasegawa, editors, *Distributed Autonomous Robotic Systems 5*, pages 299–308. Springer, 2002. xxix, 9, 62

Qinglei Hu, Hongyang Dong, Youmin Zhang, and Guangfu Ma. Tracking control of spacecraft formation flying with collision avoidance. *Aerospace Science and Technology*, 42:353–364, 2015. 63

Yongzhao Hua, Xiwang Dong, Qingdong Li, and Zhang Ren. Distributed fault-tolerant time-varying formation control for second-order multi-agent systems with actuator failures and directed topologies. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 65(6):774–778, 2017. 149

Sunan Huang, Rodney Swee Huat Teo, Jennifer Lai Pheng Kwan, Wenqi Liu, and Siarhei Michailovich Dymkou. Distributed UAV loss detection and auto-replacement protocol with guaranteed properties. *Journal of Intelligent & Robotic Systems*, 93(1-2):303–316, 2019. 149

Alberto Isidori. *Nonlinear Control Systems*, volume of *Communications and Control Engineering*. Springer, 3rd edition, 1995. 83

Stefan Ivić, Bojan Crnković, and Igor Mezić. Ergodicity-based cooperative multiagent area coverage via a potential field. *IEEE Transactions on Cybernetics*, 47(8):1983–1993, 2016. xxviii, 7

Meng Ji, Abubakr Muhammad, and Magnus Egerstedt. Leader-based multi-agent coordination: Controllability and optimal control. In *American Control Conference*, pages 1358–1363. IEEE, 2006. xxix, 8

Jian Jin and Lie Tang. Optimal coverage path planning for arable farming on 2D surfaces. *Transactions of the ASABE*, 53(1):283–295, 2010. xxix, 9, 61

Mohamed A. Kamel, Youmin Zhang, and Xiang Yu. Fault-tolerant cooperative control of multiple wheeled mobile robots under actuator faults. *IFAC-PapersOnLine*, 48(21):1152–1157, 2015. 149

Mohamed A. Kamel, Xiang Yu, and Youmin Zhang. Formation control and coordination of multiple unmanned ground vehicles in normal and faulty situations: A review. *Annual Reviews in Control*, 2020. xxvi, 3, 39, 149

Edward W. Kamen and Jonathan K. Su. *Introduction to Optimal Estimation*, volume of *Advanced Textbooks in Control and Signal Processing*. Springer, 1999. 121

Erkan Kayacan, Herman Ramon, and Wouter Saeys. Robust trajectory tracking error model-based predictive control for unmanned ground vehicles. *IEEE/ASME Transactions on Mechatronics*, 21(2):806–814, 2015. 98

Mohsen Khalili, Xiaodong Zhang, Yongcan Cao, Marios M. Polycarpou, and Thomas Parisini. Distributed fault-tolerant control of multiagent systems: An adaptive learning approach. *IEEE Transactions on Neural Networks and Learning Systems*, 2019. 149

M. Reyasudin Basir Khan, Razali Jidin, and Jagadeesh Pasupuleti. Multi-agent based distributed control architecture for microgrid energy management and optimization. *Energy Conversion and Management*, 112:288–307, 2016. 6, 8

Ernesto Kofman, Hernan Haimovich, and María M. Seron. A systematic method to obtain ultimate bounds for perturbed systems. *International Journal of Control*, 80(2):167–178, 2007. 36

Philipp N. Köhler, Matthias A. Müller, and Frank Allgöwer. A distributed economic MPC framework for cooperative control under conflicting objectives. *Automatica*, 96:368–379, 2018. 92, 93

Milan Korda, Ravi Gondhalekar, Frauke Oldewurtel, and Colin N. Jones. Stochastic MPC framework for controlling the average constraint violation. *IEEE Transactions on Automatic Control*, 59(7):1706–1721, 2014. 123

Basil Kouvaritakis and Mark Cannon. *Model Predictive Control*, volume of *Advanced Textbooks in Control and Signal Processing*. Springer, 2016. xxix, 8, 98, 123

Alexander B. Kurzhanski and István Vályi. *Ellipsoidal Calculus for Estimation and Control*, volume of *Systems & Control: Foundations & Applications*. Birkhäuser, 1997. 23

Andrea S. Laliberte and Albert Rango. Texture and scale in object-based analysis of subdecimeter resolution unmanned aerial vehicle (UAV) imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 47(3):761–770, 2009. xxix, 9, 61

Wilbur Langson, Ioannis Chryssochoos, Saša V. Raković, and David Q. Mayne. Robust model predictive control using tubes. *Automatica*, 40(1):125–133, 2004. 98

Rie B. Larsen, Bilge Atasoy, and Rudy R. Negenborn. Model predictive control for simultaneous planning of container and vehicle routes. *European Journal of Control*, 2020. xxvi, 3

Vu Tuan Hieu Le. *Robust predictive control by zonotopic set-membership estimation*. Ph.D. thesis, Supélec, 2012. 145

Vu Tuan Hieu Le, Cristina Stoica, Teodoro Álamo, Eduardo F. Camacho, and Didier Dumur. *Zonotopes: From Guaranteed State-Estimation to Control*, volume of *Automation, control and industrial engineering*. John Wiley & Sons, 2013. 20, 145

Jing Li, Shuo Chen, Fangbing Zhang, Erkang Li, Tao Yang, and Zhaoyang Lu. An adaptive framework for multi-vehicle ground speed estimation in airborne videos. *Remote Sensing*, 11(10):1241, 2019. xxix, 9, 61

Peng Li, Xiang Yu, Xiaoyan Peng, Zhiqiang Zheng, and Youmin Zhang. Fault-tolerant cooperative control for multiple UAVs based on sliding mode techniques. *Science China Information Sciences*, 60(7), 2017a. 63

Shengbo Eben Li, Yang Zheng, Keqiang Li, Yujia Wu, J. Karl Hedrick, Feng Gao, and Hongwei Zhang. Dynamical modeling and distributed control of connected and automated vehicles: Challenges and opportunities. *IEEE Intelligent Transportation Systems Magazine*, 9(3):46–58, 2017b. 6

Wei Li and Christos G. Cassandras. Distributed cooperative coverage control of sensor networks. In *44th Conference on Decision and Control*, pages 2542–2547. IEEE, 2005. 9, 62

Xiaochen Li, Wenji Mao, Daniel Zeng, and Fei-Yue Wang. Agent-based social simulation and modeling in social computing. In *International Conference on Intelligence and Security Informatics*, pages 401–412. Springer, 2008. xxvi, 2

Daniel Limón, Ignacio Alvarado, Teodoro Álamo, and Eduardo F. Camacho. MPC for tracking piecewise constant references for constrained linear systems. *Automatica*, 44(9):2382–2387, 2008. 93

Daniel Limón, Ignacio Alvarado, Teodoro Álamo, and Eduardo F. Camacho. Robust tube-based MPC for tracking of constrained linear systems with additive disturbances. *Journal of Process Control*, 20(3):248–260, 2010. 98, 102, 103, 105, 106, 112, 116

Daniel Limón, Antonio Ferramosca, Ignacio Alvarado, and Teodoro Álamo. Nonlinear MPC for tracking piece-wise constant reference signals. *IEEE Transactions on Automatic Control*, 63(11):3735–3750, 2018. 92, 93

Shu Lin, Bart De Schutter, Yugeng Xi, and Hans Hellendoorn. Efficient network-wide model-based predictive control for urban traffic networks. *Transportation Research Part C: Emerging Technologies*, 24:122–140, 2012. xxvi, 3

Wei Liu, Wei Gu, Wanxing Sheng, Xiaoli Meng, Zaijun Wu, and Wu Chen. Decentralized multi-agent system-based cooperative frequency control for autonomous microgrids with communication constraints. *IEEE Transactions on Sustainable Energy*, 5(2):446–456, 2014. xxvii, 5

Zhixiang Liu, Chi Yuan, Xiang Yu, and Youmin Zhang. Leader-follower formation control of unmanned aerial vehicles in the presence of obstacles and actuator faults. *Unmanned Systems*, 4(3):197–211, 2016. 5

Zonglin Liu and Olaf Stursberg. Recursive feasibility and stability of MPC with time-varying and uncertain state constraints. In *18th European Control Conference*, pages 1766–1771. IEEE, 2019. 93

Stuart Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982. xxx, 10, 62

Michel Loève. *Probability Theory I*, volume 45 of *Graduate Texts in Mathematics*. Springer, 4th edition, 1977. 56

Michel Loève. *Probability Theory II*, volume 46 of *Graduate Texts in Mathematics*. Springer, 4th edition, 1978. 56

Johan Löfberg. YALMIP: A toolbox for modeling and optimization in MATLAB. In *IEEE International Conference on Robotics and Automation*, pages 284–289. IEEE, 2004. 21

Johan Löfberg. Oops! I cannot do it again: Testing for recursive feasibility in MPC. *Automatica*, 48(3):550–555, 2012. 92

Johan Löfberg. YALMIP. https://yalmip.github.io/download/, jan 2020. 21

Thillainathan Logenthiran, Dipti Srinivasan, Ashwin M. Khambadkone, and Htay Nwe Aung. Multiagent system for real-time operation of a microgrid in real-time digital simulator. *IEEE Transactions on Smart Grid*, 3(2):925–933, 2012. xxvi, 3

Shuai Lu, Nader Samaan, Ruisheng Diao, Marcelo Elizondo, Chunlian Jin, Ebony Mayhorn, Yu Zhang, and Harold Kirkham. Centralized and decentralized control for demand response. In *Innovative Smart Grid Technology*, pages 1–8. IEEE, 2011. xxvii, 5

David G. Luenberger. Observing the state of a linear system. *IEEE Transactions on Military Electronics*, 8(2):74–80, 1964. 86, 97, 122

Daniel Lyons, Jan-Peter Calliess, and Uwe D. Hanebeck. Chance constrained model predictive control for multi-agent systems with coupling constraints. In *American Control Conference*, pages 1223–1230. IEEE, 2012. 8, 124

Jan M. Maciejowski. *Predictive Control: with Constraints*. Pearson Education, 2002. xxix, 8

James MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297, 1967. 10

José M. Maestre and Rudy R. Negenborn. *Distributed Model Predictive Control Made Easy*, volume 69 of *Intelligent Systems, Control and Automation: Science and Engineering*. Springer, 2014. xxvi, xxviii, xxix, 2, 6, 8

Vladimir Malakhov, Kirill Nesytykh, and Tatiana Dubynina. A multi-agent approach for the intersectoral modeling of the Russian economy. In *Tenth International Conference Management of Large-Scale System Development*, pages 1–5. IEEE, 2017. xxvi, 2

Martina Mammarella, Elisa Capello, Hyeongjun Park, Giorgio Guglieri, and Marcello Romano. Tube-based robust model predictive control for spacecraft proximity operations in the presence of persistent disturbance. *Aerospace Science and Technology*, 77:585–594, 2018. 98

Jiři Matoušek and Bernd Gärtner. *Understanding and Using Linear Programming*, volume of *Universitext*. Springer, 2007. 27

Jacob Mattingley and Stephen Boyd. CVXGEN: A code generator for embedded convex optimization. *Optimization and Engineering*, 13(1):1–27, 2012. 74, 88, 115, 141, 160, 163

Jacob Mattingley and Stephen Boyd. CVXGEN: Code generation for convex optimization. http://www.cvxgen.com/docs/index.html, dec 2013. 74, 88, 115, 141, 160, 163

David Q. Mayne, James B. Rawlings, Christopher V. Rao, and Pierre O.M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000. xxix, 8

David Q. Mayne, María M. Seron, and Saša V. Raković. Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica*, 41 (2):219–224, 2005. xxix, xxxi, 8, 12, 14, 98

David Q. Mayne, Saša V. Raković, Rolf Findeisen, and Frank Allgöwer. Robust output feedback model predictive control of constrained linear systems. *Automatica*, 42(7):1217–1222, 2006. xxix, 8, 12, 97, 98, 107, 144

Dory Merhy. *Contribution to ellipsoidal and zonotopic set-membership state estimation*. Ph.D. thesis, Université Paris-Saclay, 2019. 23

Luis Merino, Fernando Caballero, J. Ramiro Martínez de Dios, Iván Maza, and Aníbal Ollero. An unmanned aircraft system for automatic forest fire monitoring and measurement. *Journal of Intelligent & Robotic Systems*, 65(1-4): 533–548, 2012. xxix, 9, 61

Mehran Mesbahi and Magnus Egerstedt. *Graph Theoretic Methods in Multiagent Networks*, volume 33. Princeton University Press, 2010. 7

Nathan Michel, Sylvain Bertrand, Sorin Olaru, Giorgio Valmorbida, and Didier Dumur. Design and flight experiments of a tube-based model predictive controller for the AR.Drone 2.0 quadrotor. *IFAC-PapersOnLine*, 52(22):112–117, 2019. 98, 113

Luis I. Minchala Avila, Luis E. Garza Castañón, Adriana Vargas Martínez, and Youmin Zhang. A review of optimal control techniques applied to the energy management and control of microgrids. *Procedia Computer Science*, 52:780–787, 2015. xxvi, 3

Mehran Mirshams and Mohsen Khosrojerdi. Attitude control of an underactuated spacecraft using tube-based MPC approach. *Aerospace Science and Technology*, 48:140–145, 2016. 98

Miad Moarref and Luis Rodrigues. An optimal control approach to decentralized energy-efficient coverage problems. *IFAC Proceedings Volumes*, 47(3):6038–6043, 2014. xxx, 9, 11, 62, 63

Irinel-Constantin Morărescu and Mirko Fiacchini. Topology preservation for multi-agent networks: design and implementation. In Alexandre Seuret, Laurentiu Hetel, Jamal Daafouz, and Karl H. Johansson, editors, *Delays and Networked Control Systems*, volume 6 of *Advances in Delays and Dynamics*, pages 253–269. Springer, 2016. 5

Kristian Hengster Movric and Frank L. Lewis. Cooperative optimal control for multi-agent systems on directed graph topologies. *IEEE Transactions on Automatic Control*, 59(3):769–774, 2013. xxix, 8

Richard M. Murray. Recent research in cooperative control of multivehicle systems. *Journal of Dynamic Systems, Measurement, and Control*, 129(5):571–583, 2007. 9

Roger B. Myerson. *Game Theory, Analysis of Conflict*. Harvard University Press, 1991. 8

Rudy R. Negenborn, Bart De Schutter, and Johannes Hellendoorn. Multi-agent model predictive control for transportation networks: Serial versus parallel schemes. *Engineering Applications of Artificial Intelligence*, 21(3):353–366, 2008. xxvi, 3

Francesco Nex and Fabio Remondino. UAV for 3D mapping applications: a review. *Applied Geomatics*, 6(1):1–15, 2014. xxix, 9, 61

Minh Tri Nguyen. *Commande prédictive sous contraintes de sécurité pour des systèmes dynamiques Multi-Agents*. Ph.D. thesis, Université Paris-Saclay, 2016. xxix, xxx, 3, 9, 11, 12, 13, 55, 62, 63, 65, 70, 86, 94, 193

Minh Tri Nguyen and Cristina Stoica Maniu. Voronoi based decentralized coverage problem: From optimal control to model predictive control. In *24th Mediterranean Conference on Control and Automation*, pages 1307–1312. IEEE, 2016. xxvi, 3, 12, 63

Minh Tri Nguyen, Cristina Stoica Maniu, and Sorin Olaru. Optimization-based control for multi-agent deployment via dynamic Voronoi partition. *IFAC-PapersOnLine*, 50(1):1828–1833, 2017. 12, 63

Alexandros Nikou and Dimos V. Dimarogonas. Decentralized tube-based model predictive control of uncertain nonlinear multiagent systems. *International Journal of Robust and Nonlinear Control*, 29(10):2799–2818, 2019. 8, 98

Yugang Niu, Daniel W.C. Ho, and James Lam. Robust integral sliding mode control for uncertain stochastic systems with time-varying delay. *Automatica*, 41 (5):873–880, 2005. 123

Sorin Olaru, José A. De Doná, María M. Seron, and Florin Stoican. Positive invariant sets for fault tolerant multisensor control schemes. *International Journal of Control*, 83(12):2622–2640, 2010. 36, 37, 38

Sorin Olaru, Alexandra Grancharova, and Fernando Lobo Pereira. *Developments in Model-Based Optimization and Control*, volume 464 of *Lecture Notes in Control and Information Sciences*. Springer, 2015. xxix, 8

Reza Olfati Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on Automatic Control*, 51(3):401–420, 2006. 6, 7

Reza Olfati Saber and Richard M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, 2004. xxviii, 7, 63

Reza Olfati Saber, J. Alex Fax, and Richard M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1): 215–233, 2007. xxvii, 4

Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. MIT press, 1994. xxviii, 7

Elena Panteley and Antonio Loría. Synchronization and dynamic consensus of heterogeneous networked systems. *IEEE Transactions on Automatic Control*, 62 (8):3758–3773, 2017. 7

Sotiris Papatheodorou, Yiannis Stergiopoulos, and Anthony Tzes. Distributed area coverage control with imprecise robot localization. In *24th Mediterranean Conference on Control and Automation*, pages 214–219. IEEE, 2016. xxx, xxxi, 11, 12, 14, 99

Sotiris Papatheodorou, Anthony Tzes, and Yiannis Stergiopoulos. Collaborative visual area coverage. *Robotics and Autonomous Systems*, 92:126–138, 2017. xxvi, xxx, xxxi, 3, 9, 11, 14, 99

Daniel Pickem, Paul Glotfelter, Li Wang, Mark Mote, Aaron Ames, Eric Feron, and Magnus Egerstedt. The robotarium: A remotely accessible swarm robotics research testbed. In *IEEE International Conference on Robotics and Automation*, pages 1699–1706. IEEE, 2017. 74

Jeroen Ploeg, Nathan Van De Wouw, and Henk Nijmeijer. Lp string stability of cascaded systems: Application to vehicle platooning. *IEEE Transactions on Control Systems Technology*, 22(2):786–793, 2013. xxvi, 3

Boris T. Polyak, Sergey A. Nazin, Cécile Durieu, and Eric Walter. Ellipsoidal parameter or state estimation under model uncertainty. *Automatica*, 40(7): 1171–1179, 2004. 23

Alexander Poznyak, Andrey Polyakov, and Vadim Azhmyakov. *Attractive Ellipsoids in Robust Control*, volume of *Systems & Control: Foundations & Applications*. Birkhäuser, 2014. 23

Maria Prandini, Simone Garatti, and John Lygeros. A randomized approach to stochastic model predictive control. In *51st Conference on Decision and Control*, pages 7315–7320. IEEE, 2012. 123

Franco P. Preparata and Michael Ian Shamos. *Computational Geometry: An Introduction*, volume of *Texts and Monographs in Computer Science*. Springer, 1985. 51

James A. Primbs and Chang Hwan Sung. Stochastic receding horizon control of constrained linear systems with state and control multiplicative noise. *IEEE Transactions on Automatic Control*, 54(2):221–230, 2009. 123

Ionela Prodan. *Constrained control of dynamical Multi-Agent systems*. Ph.D. thesis, Supélec, 2012. xxvii, xxviii, 5, 7

Ionela Prodan, Sorin Olaru, Cristina Stoica, and Silviu-Iulian Niculescu. Predictive control for tight group formation of multi-agent systems. *IFAC Proceedings Volumes*, 44(1):138–143, 2011. xxvi, xxvii, 3, 4, 8, 98

Nicanor Quijano, Carlos Ocampo Martinez, Julian Barreiro Gomez, German Obando, Andres Pantoja, and Eduardo Mojica Nava. The role of population games and evolutionary dynamics in distributed control systems: The advantages of evolutionary game theory. *IEEE Control Systems Magazine*, 37(1):70–97, 2017. 8

Bharat Menon Radhakrishnan and Dipti Srinivasan. A multi-agent based distributed energy management scheme for smart grid applications. *Energy*, 103: 192–204, 2016. xxvi, 3

Saša V. Raković, Eric C. Kerrigan, Konstantinos I. Kouramas, and David Q. Mayne. Invariant approximations of the minimal robust positively invariant set. *IEEE Transactions on Automatic Control*, 50(3):406–410, 2005. 36, 104, 107

Margarita Ramírez Ramírez, Felipe Lara Rosano, Ricardo Fernando Rosales Cisneros, Esperanza Manrique Rojas, Hilda Beatriz Ramírez Moreno, and Gonzalo Maldonado Guzmán. Multi-agent complex system for identification of characteristics and personality types and their relationship in the process of motivation of students. In *Agents and Multi-agent Systems: Technologies and Applications 2019*, pages 143–151. Springer, 2020. xxvi, 2

James B. Rawlings and David Q. Mayne. *Model Predictive Control: Theory and Design*. Nob Hill Pub., 2009. xxix, 8

Li-Hong Ren, Yong-Sheng Ding, Yi-Zhen Shen, and Xiang-Feng Zhang. Multi-agent-based bio-network for systems biology: protein–protein interaction network as an example. *Amino Acids*, 35(3):565–572, 2008. xxvi, 2

Wei Ren and Randal W. Beard. *Distributed Consensus in Multi-Vehicle Cooperative Control*, volume of *Communications & Control Engineering*. Springer, 2008. xxviii, 6, 7, 63, 71, 74, 78

Elon Rimon and Daniel E. Koditschek. Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, 8(5):501–518, 1992. 7

Benjamin Roche, Jean-François Guégan, and François Bousquet. Multi-agent systems in epidemiology: a first step for computational biology in the study of vector-borne disease transmission. *BMC Bioinformatics*, 9(1):435, 2008. xxvi, 2

Murray Rosenblatt. *Random Processes*, volume 17 of *Graduate Texts in Mathematics*. Springer, 2nd edition, 1974. 56

Federico Rossi, Saptarshi Bandyopadhyay, Michael Wolf, and Marco Pavone. Review of multi-agent algorithms for collective behavior: a structural taxonomy. *IFAC-PapersOnLine*, 51(12):112–117, 2018. xxviii, 6

Gauthier Rousseau, Cristina Stoica Maniu, Sihem Tebbani, Mathieu Babel, and Nicolas Martin. Quadcopter-performed cinematographic flight plans using minimum jerk trajectories and predictive camera control. In *17th European Control Conference*, pages 2897–2903. IEEE, 2018. 114

Peter Forbes Rowat. *Representing spatial experience and solving spatial problems in a simulated robot environment*. Ph.D. thesis, University of British Columbia, 1979. 44

Francesco Sabatino. *Quadrotor control: modeling, nonlinear control design, and simulation*. Master's thesis, KTH Royal Institute of Technology, 2015. 78

Iman Saboori and Khashayar Khorasani. Actuator fault accommodation strategy for a team of multi-agent systems subject to switching topology. *Automatica*, 62: 200–207, 2015. 149

Marcelo A. Santos, Antonio Ferramosca, and Guilherme V. Raffo. Tube-based MPC with Nonlinear Control for Load Transportation using a UAV. *IFAC-PapersOnLine*, 51(25):459–465, 2018. 98

Riccardo Scattolini. Architectures for distributed and hierarchical model predictive control–a review. *Journal of Process Control*, 19(5):723–731, 2009. xxviii, 6

Carsten Scherer and Siep Weiland. *Linear Matrix Inequalities in Control*. Dutch Institute for Systems and Control, 2015. 20

Alexander Schrijver. *Theory of Linear and Integer Programming*, volume of *Wiley Series in Discrete Mathematics & Optimization*. John Wiley & Sons, 1998. 25

Mac Schwager, Daniela Rus, and Jean-Jacques Slotine. Unifying geometric, probabilistic, and potential field approaches to multi-robot deployment. *The International Journal of Robotics Research*, 30(3):371–383, 2011. xxvi, xxix, xxx, 3, 9, 11, 61, 62, 63

Fred Schweppe. Recursive state estimation: Unknown but bounded errors and system inputs. *IEEE Transactions on Automatic Control*, 13(1):22–28, 1968. 23

Elham Semsar Kazerooni and Khashayar Khorasani. Multi-agent team cooperation: A game theory approach. *Automatica*, 45(10):2205–2213, 2009. 8

Emilio Serrano and Carlos A. Iglesias. Validating viral marketing strategies in Twitter via agent-based social simulation. *Expert Systems with Applications*, 50: 140–150, 2016. xxvi, 2

S.M. Hashemy Shahdany, Saleh Taghvaeian, José M. Maestre, and A.R. Firoozfar. Developing a centralized automatic control system to increase flexibility of water delivery within predictable and unpredictable irrigation water demands. *Computers and Electronics in Agriculture*, 163:104862, 2019. xxvi, 3

Farid Sharifi, Abbas Chamseddine, Hamid Mahboubi, Youmin Zhang, and Amir G. Aghdam. A distributed deployment strategy for a network of cooperative autonomous vehicles. *IEEE Transactions on Control Systems Technology*, 23(2): 737–745, 2014. 11, 39, 62, 63

Daniel Simon, Johan Löfberg, and Torkel Glad. Reference tracking MPC using dynamic terminal set transformation. *IEEE Transactions on Automatic Control*, 59(10):2790–2795, 2014. 93

Vijay Pratap Singh, Nand Kishor, and Paulson Samuel. Distributed multi-agent system-based load frequency control for multi-area power system in smart grid. *IEEE Transactions on Industrial Electronics*, 64(6):5151–5160, 2017. xxvi, 3

Yuan Song, Bing Wang, Zhijie Shi, Krishna R. Pattipati, and Shalabh Gupta. Distributed algorithms for energy-efficient even self-deployment in mobile sensor networks. *IEEE Transactions on Mobile Computing*, 13(5):1035–1047, 2013. xxx, 11

Nathan Sorensen and Wei Ren. Rendezvous problem in multi-vehicle systems: Information relay and local information based strategies. In *IEEE Mountain Workshop on Adaptive and Learning Systems*, pages 183–188. IEEE, 2006. xxviii, 7

Kokichi Sugihara. Approximation of generalized Voronoi diagrams by ordinary Voronoi diagrams. *CVGIP: Graphical Models and Image Processing*, 55(6):522–531, 1993. xxix, 9, 44

A. Sujil, Jatin Verma, and Rajesh Kumar. Multi agent system: concepts, platforms and applications in power systems. *Artificial Intelligence Review*, 49(2): 153–182, 2018. xxvii, 4

Zhongqi Sun, Li Dai, Kun Liu, Yuanqing Xia, and Karl H. Johansson. Robust MPC for tracking constrained unicycle robots with additive disturbances. *Automatica*, 90:172–184, 2018. 98

Andrew S. Tanenbaum and Maarten Van Steen. *Distributed Systems: Principles and Paradigms*. Prentice-Hall, 2007. xxvii, 3

Petter Tøndel, Tor Arne Johansen, and Alberto Bemporad. An algorithm for multi-parametric quadratic programming and explicit MPC solutions. *Automatica*, 39(3):489–497, 2003. xxix, 8

Marina Torres, David A. Pelta, José L. Verdegay, and Juan C. Torres. Coverage path planning with unmanned aerial vehicles for 3D terrain reconstruction. *Expert Systems with Applications*, 55:441–451, 2016. xxix, 9, 61

Paul A. Trodden. A one-step approach to computing a polytopic robust positively invariant set. *IEEE Transactions on Automatic Control*, 61(12):4100–4105, 2016. 36, 37, 38

Paul A. Trodden and Arthur G. Richards. Cooperative tube-based distributed MPC for linear uncertain systems coupled via constraints. In José M. Maestre and Rudy R. Negenborn, editors, *Distributed Model Predictive Control Made Easy*, volume 69 of *Intelligent Systems, Control and Automation: Science and Engineering*, pages 57–72. Springer, 2014. 8, 99

Juan Carlos Trujillo, Rodrigo Munguía, Eduardo Ruiz Velázquez, and Bernardino Castillo Toledo. A cooperative aerial robotic approach for tracking and estimating the 3D position of a moving object by using pseudo-stereo vision. *Journal of Intelligent & Robotic Systems*, 96(2):297–313, 2019. xxix, 9, 61

Mert Turanli and Hakan Temeltas. Multi-robot workspace allocation with Hopfield networks and imprecise localization. *Acta Polytechnica Hungarica*, 17(5): 169–188, 2020. xxx, 11

Valerio Turri, Bart Besselink, and Karl H. Johansson. Cooperative look-ahead control for fuel-efficient and safe heavy-duty vehicle platooning. *IEEE Transactions on Control Systems Technology*, 25(1):12–28, 2016. xxvi, 3, 6

Mariliza Tzes, Sotiris Papatheodorou, and Anthony Tzes. Visual area coverage by heterogeneous aerial agents under imprecise localization. *IEEE Control Systems Letters*, 2(4):623–628, 2018. xxx, 11, 44, 99

Valery A. Ugrinovskii. Robust H_∞ control in the presence of stochastic uncertainty. *International Journal of Control*, 71(2):219–237, 1998. 123

Sebastian Van De Hoef, Karl H. Johansson, and Dimos V. Dimarogonas. Fuel-efficient en route formation of truck platoons. *IEEE Transactions on Intelligent Transportation Systems*, 19(1):102–112, 2017. xxvi, 3

Andreas Varga. *Solving Fault Diagnosis Problems*, volume 84 of *Studies in Systems, Decision and Control*. Springer, 1st edition, 2017. 148

Christos K. Verginis and Dimos V. Dimarogonas. Closed-form barrier functions for multi-agent ellipsoidal systems with uncertain lagrangian dynamics. *IEEE Control Systems Letters*, 3(3):727–732, 2019. xxvii, 5

Christophe Viel. *Control law and state estimators design for multi-agent system with reduction of communications by event-triggered approach*. Ph.D. thesis, Université Paris-Saclay, 2017. 6

Holger Voos. Nonlinear control of a quadrotor micro-UAV using feedback-linearization. In *IEEE International Conference on Mechatronics*, pages 1–6. IEEE, 2009. 83

Georges Voronoï. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Premier Mémoire. Sur quelques propriétés des formes quadratiques positives parfaites. *Journal für die reine und angewandte Mathematik*, 133:97–178, 1908. xxix, 9, 41

Nianfeng Wan, Chen Zhang, and Ardalan Vahidi. Probabilistic anticipation and control in autonomous car following. *IEEE Transactions on Control Systems Technology*, 27(1):30–38, 2017. 124

Ban Wang, Khaled A. Ghamry, and Youmin Zhang. Trajectory tracking and attitude control of an unmanned quadrotor helicopter considering actuator dynamics. In *35th Chinese Control Conference*, pages 10795–10800. IEEE, 2016a. 79

Dong Wang, Zehua Wang, Wei Wang, and Jie Lian. Event-triggered based containment control of multi-agent systems with general linear dynamics. In *IEEE International Conference on Information and Automation*, pages 1064–1069. IEEE, 2015. xxvii, 4

Qiang Wang, Jie Chen, and Hao Fang. Fault-tolerant topology control algorithm for mobile robotic networks. *International Journal of Control, Automation and Systems*, 12(3):582–589, 2014. 149

Shiyong Wang, Jiafu Wan, Daqiang Zhang, Di Li, and Chunhua Zhang. Towards smart factory for industry 4.0: a self-organized multi-agent system with big data based feedback and coordination. *Computer Networks*, 101:158–168, 2016b. 2

Ye Wang, Vicenç Puig, and Gabriela Cembrano. Non-linear economic model predictive control of water distribution networks. *Journal of Process Control*, 56: 23–34, 2017. xxvi, 3

Yuanzhe Wang, Mao Shan, Yufeng Yue, and Danwei Wang. Vision-based flexible leader–follower formation tracking of multiple nonholonomic mobile robots in unknown obstacle environments. *IEEE Transactions on Control Systems Technology*, 28(3):1025–1033, 2019. 5

Gerhard Weiss. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT press, 1999. 2

Guoxing Wen, C.L. Philip Chen, Yan-Jun Liu, and Zhi Liu. Neural network-based adaptive leader-following consensus control for a class of nonlinear multiagent state-delay systems. *IEEE Transactions on Cybernetics*, 47(8):2151–2160, 2016. xxix, 8

Michael J. Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley & Sons, 2nd edition, 2009. xxv, 1, 2

Michael J. Wooldridge and Nicholas R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995. xxv, 2

Peter R. Wurman, Raffaello D'Andrea, and Mick Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI magazine*, 29(1):9–9, 2008. xxvi, 3

Qing Xu, Hao Yang, Bin Jiang, Donghua Zhou, and Youmin Zhang. Fault tolerant formations control of UAVs subject to permanent and intermittent faults. *Journal of Intelligent & Robotic Systems*, 73(1-4):589–602, 2014. 149

Yonggang Xu and João P. Hespanha. Communication logic design and analysis for networked control systems. In Laura Menini, Luca Zaccarian, and Chaouki T. Abdallah, editors, *Current trends in nonlinear systems and control*, volume of *Systems and Control: Foundations & Applications*, pages 495–514. Birkhäuser, 2006. xxvii, 4

Xiang Yu, Zhixiang Liu, and Youmin Zhang. Fault-tolerant formation control of multiple UAVs in the presence of actuator faults. *International Journal of Robust and Nonlinear Control*, 26(12):2668–2685, 2016. 149

Ziquan Yu, Zhixiang Liu, Youmin Zhang, Yaohong Qu, and Chun-Yi Su. Distributed finite-time fault-tolerant containment control for multiple unmanned aerial vehicles. *IEEE Transactions on Neural Networks and Learning Systems*, 31 (6):2077–2091, 2020. xxix, 8

Chi Yuan, Zhixiang Liu, and Youmin Zhang. Learning-based smoke detection for unmanned aerial vehicles applied to forest fire surveillance. *Journal of Intelligent & Robotic Systems*, 93(1-2):337–349, 2019. xxix, 9, 61

Yuan Yuan, Zidong Wang, Peng Zhang, and Hongli Dong. Nonfragile near-optimal control of stochastic time-varying multiagent systems with control-and state-dependent noises. *IEEE Transactions on Cybernetics*, 49(7):2605–2617, 2018. xxix, 8

George Zames. Feedback and optimal sensitivity: Model reference transformations, multiplicative seminorms, and approximate inverses. *IEEE Transactions on Automatic Control*, 26(2):301–320, 1981. 98

Huaguang Zhang, Jilie Zhang, Guang-Hong Yang, and Yanhong Luo. Leader-based optimal coordination control for the consensus problem of multiagent differential games via fuzzy adaptive dynamic programming. *IEEE Transactions on Fuzzy Systems*, 23(1):152–163, 2014. 8

Xiaojing Zhang, Kostas Margellos, Paul Goulart, and John Lygeros. Stochastic model predictive control using a combination of randomized and robust optimization. In *52nd Conference on Decision and Control*, pages 7740–7745. IEEE, 2013. 123

Youmin Zhang and Jin Jiang. Bibliographical review on reconfigurable fault-tolerant control systems. *Annual Reviews in Control*, 32(2):229–252, 2008. 148

Bo Zhou, Wei Wang, and Hao Ye. Cooperative control for consensus of multi-agent systems with actuator faults. *Computers & Electrical Engineering*, 40(7): 2154–2166, 2014. 149

Hai Zhu and Javier Alonso Mora. Chance-constrained collision avoidance for mavs in dynamic environments. *IEEE Robotics and Automation Letters*, 4(2): 776–783, 2019. 124

Shuyi Zhu, Ran Sun, Jiaolong Wang, Jihe Wang, and Xiaowei Shao. Robust model predictive control for multi-step short range spacecraft rendezvous. *Advances in Space Research*, 62(1):111–126, 2018. 124

Günter M. Ziegler. *Lectures on Polytopes*, volume 152 of *Graduate Texts in Mathematics*. Springer, 2007. 27, 32

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Commande prédictive robuste pour le déploiement et la reconfiguration de systèmes multi-agents

Mots clés : Systèmes multi-agents, Commande prédictive, Commande robuste, Déploiement, Partition de Voronoï, Méthodes ensemblistes

Résumé : Cette thèse porte sur le développement de techniques de commande prédictive pour le déploiement et la reconfiguration d'un système multi-agents dans une zone convexe et bornée en deux dimensions. Un nouvel algorithme de commande prédictive décentralisé, fondé sur une partition de Voronoï de l'espace pour le déploiement d'une flotte de drones quadrirotor, est construit. La loi de commande prédictive décentralisée est d'abord rendue plus robuste pour supporter des perturbations déterministes bornées s'appliquant sur les agents, introduisant une nouvelle partition de Voronoï garantie fondée sur des boîtes pour assurer la sécurité du déploiement. Dans ce cas, un nouveau correcteur prédictif fondé sur des tubes avec observateur est conçu en résolvant des problèmes d'optimisation sous contraintes d'inégalités matricielles linéaires/bilinéaires. Ensuite, pour supporter des perturbations stochastiques non bornées, un nouvel algorithme de commande prédictive sous contraintes probabilistes est proposé, rendu résoluble par la transformation des contraintes probabilistes apparaissant dans le problème d'optimisation en contraintes algébriques. Enfin, une stratégie de reconfiguration fondée sur un correcteur prédictif décentralisé est conçue pour permettre à des agents de rejoindre ou de quitter le système multi-agents durant son déploiement. Des résultats de simulation sur une flotte de drones quadrirotor valident l'efficacité des stratégies de commande proposées.

Title: Robust model predictive control for deployment and reconfiguration of multi-agent systems

Keywords: Multi-agent systems, Model predictive control, Robust control, Deployment control, Voronoi tessellation, Set-theoretic methods

Abstract: This thesis presents Model Predictive Control (MPC) techniques for the deployment and the reconfiguration of a dynamical Multi-Agent System (MAS) in a bounded convex two-dimensional area. A novel decentralized predictive control law for the Voronoi-based deployment of a fleet of quadrotor Unmanned Aerial Vehicles (UAVs) is derived. The proposed decentralized MPC is firstly robustified to deal with bounded deterministic perturbations acting on the agents, introducing a new box-based guaranteed Voronoi tessellation to ensure a safe deployment. In this case, a new output-feedback tube-based MPC is designed by solving constrained optimization procedures relying on linear/bilinear matrix inequalities. Secondly, to deal with unbounded stochastic perturbations, a new output-feedback chance-constrained MPC algorithm is proposed, solved by mean of a relaxation of the considered probabilistic constraints into algebraic constraints. Finally, a decentralized MPC-based reconfiguration strategy is designed to deal with the case of agents joining or leaving the multi-agent system during the deployment. Illustrative simulation results on a fleet of quadrotor UAVs validate the effectiveness of the proposed control strategies.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France