Security of Distributed Model Predictive Control under False Data Injection

Rafael Accácio Nogueira

DOCTORAT BRETAGNE LOIRE / MATHSTIC

CentraleSupélec

THÈSE DE DOCTORAT DE

CENTRALESUPÉLEC

ÉCOLE DOCTORALE Nº 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Automatique, Productique, Robotique*

Par

Rafael Accácio NOGUEIRA

Security of distributed Model Predictive Control under False Data Injection

Thèse présentée et soutenue à Cesson Sévigné, le 12 décembre 2022
Unité de recherche : Institut d'Électronique et des Technologies du numÉrique (IETR)
Thèse Nº : 2022CSUP0006

Rapporteurs avant soutenance :

	José María MAESTRE TORREBLANCA	Professeur, Universidad de Sevilla
	Ionela PRODAN	Maîtresse de Conférences, Grenoble INP

Composition du Jury :

Président :	Eric BIDEAUX	Professeur, INSA Lyon
Examinateurs :	Eric BIDEAUX	Professeur, INSA Lyon
	Romain BOURDAIS	Professeur, CentraleSupélec
	José María MAESTRE TORREBLANCA	Professeur, Universidad de Sevilla
	Ionela PRODAN	Maîtresse de Conférences, Grenoble INP
Dir. de thèse :	Hervé GUÉGUEN	Professeur, CentraleSupélec

Acknowledgement

First, I thank God for **all the people** put in my life. They all served as an example of who to be (or not). The good and bad experiences made me the individual I am now.

Thanks to all my teachers and professors, who showed me the beauty of transposing knowledge from one's mind to others. You undoubtedly instigated me to keep learning new things and inspired me to have a teaching experience during my Ph.D. studies.

I thank my parents, Rosemeri and Rogerio, for their unconditional love and selflessness in supporting my choices. Thanks for the first education and the moral compass which guides me wherever I go. Tenho orgulho de ser filho de vocês. Obrigado por tudo, sem vocês não chegaria a lugar algum. Amo muito vocês! [1]

I thank my wife, Evelise. These were difficult years for both of us for diverse reasons, but knowing you were there, made things lighter. Thank you for choosing to be with me despite all randomness imposed on our lives. Also, thanks for your suggestions. Having someone intelligent with critical thinking by my side always comes in handy. I love you!

I thank my supervisors, Hervé et Romain. Thank you for accepting me as your Ph.D. student and helping me with the initial bureaucracy. Thanks for your patience when I was overwhelmed by the possible choices. Taking a step back does give us a clearer overall picture. Thanks for your guidance during these years, in research and in life. Thanks also for the opportunity of taking classes. Your impact on my life will never be forgotten.

I thank everybody in the AUT team of CentraleSupélec IETR for the warm reception and the chats in the break room or during lunch. It was nice to be here once again.

Thanks, Paul, for the tips on academic life. Thanks, Mélissa, for helping me understand the administrative part of teaching. Thank you, Pierre, for the extensive discussions on the most diverse topics and the help with my visual supports used for the conferences. Thank you, Nabil and Marie-Anne, for always being gentle and helpful. Working with you on the other side of the curtain was a pleasure.

Thanks to my fellow Ph.D. students Marina, Alexandre, Joy, Elsy, Sen, and Zhighang for the exchanges that gave me other views on life. I wish you all well at the end of your studies. And I also thank my other fellow Ph.D. students, now Doctors, Amanda, Xiang, and Jesse-James. Thanks for the help in my initial steps as a Ph.D. student and all the other moments we spent talking about work, life, music, and games.

Thanks, Simon, for the collaboration on our article. Your suggestions greatly influenced and contributed to the shape of this work.

1. I'm proud of being your son. Thanks for everything; I wouldn't get anywhere if it weren't for you. I love you very much!

I thank my Monitoring Committee, Malek Ghanes, and Stanislav. The discussions about publishing, academic life, and the incentive to start teaching were invaluable.

I thank all the technical and administrative personnel at CentraleSupélec. I thank you not only for helping me during the first week but also for the enrollment in conferences and courses, solving IT problems, and finally, in this big last step. Thanks, Catherine, Karine, Cecile, Sarah, Jeannine, Véronique, Ophelie, Gaëlle, Yves, and especially Gregory Cadeau and Myriam Andrieux.

I also thank Professors Eric Bideaux, J.M. Maestre, and Ionela Prodan for accepting to be the examiners of my work. The comments on the reports and during the defense about this work were heartwarming. The questions and discussions were inspiring.

And I thank you, reader, for your time. This is the work of 3 years of my life.

Enjoy.

Table of Contents

Acknowledgement **3**

List of Acronyms **9**

List of Figures **11**

List of Tables **15**

Notation **17**

1 Introduction **21**
1.1 When greediness backfires . . . 22
1.2 Motivation and Contributions . . . 25
1.3 Publications . . . 26
1.4 Programming and material . . . 27

I Model Predictive Control - Decomposition and Security **29**

2 Decomposing the Model Predictive Control **31**
2.1 Model Predictive Control . . . 31
2.1.1 General discrete Model-based Predictive Control (MPC) . . . 32
2.1.2 Convex MPC (quadratic case) . . . 33
2.2 Optimization Decomposition Frameworks . . . 35
2.2.1 Decomposable problems . . . 35
2.2.2 Uncoupled problems . . . 36
2.2.3 Coupled problems . . . 37
2.2.4 Generalizing . . . 39
2.2.5 Decomposition techniques . . . 40

3 Topologies, Trust and Responsivity **41**
3.1 How to distribute the computation units? . . . 42
3.2 Representing communication . . . 43
3.3 Do computing units communicate? . . . 45
3.4 How do units influence each other? . . . 45
3.5 How do units communicate? . . . 47

3.6 Some drawbacks and compromises 48

4 Anomalous Behaviors — What they are and how to combat 51

4.1 What are anomalous behaviors? 51
4.2 Security 52
4.3 Vulnerabilities in Cyber-Physical Systems (CPS) 53
4.4 Attacks in CPS 54
4.5 Attacks in distributed Model based Predictive Control (dMPC) 58
4.6 Securing CPS 59
4.6.1 Passive strategies 60
4.6.2 Active strategies 61
4.6.3 Detection 61
4.6.4 Mitigation/Recovery 62

II Towards a safe dMPC algorithm 63

5 Vulnerabilities in Primal Decomposition-based dMPC 69

5.1 Primal decomposition-based dMPC 70
5.2 Example and Interpretations 73
5.3 Anomalous behaviors and their consequences 76
5.3.1 Attack of interest 77
5.4 Conclusion 79

6 Resilient Primal Decomposition-based dMPC for deprived systems 81

6.1 Deprived systems under attack 82
6.1.1 Deprived Systems 82
6.1.2 Why deprived systems? 84
6.1.3 Analysis of local problems 84
6.1.4 Analysis of negotiation 86
6.2 A detection mechanism 90
6.2.1 Considerations about parameter estimation 91
6.2.2 Complete detection technique 94
6.3 Mitigation 94
6.4 Complete safe dMPC for deprived systems 98
6.5 Numerical experiment 98
6.5.1 District Heating Network 99
6.5.2 Applying Resilient Primal Decomposition-based dMPC for deprived systems (RPdMPC-DS) 101
6.6 Conclusion 103

7 Resilient Primal Decomposition-based dMPC using Artificial Scarcity 105
7.1 Relaxing the problem . . . 106
7.1.1 Impact on local problems' solution . . . 107
7.1.2 Impact on negotiation equations . . . 112
7.2 Adapting detection and mitigation methods . . . 118
7.2.1 Mitigation and artificial scarcity . . . 118
7.2.2 Detection . . . 119
7.2.3 Multiple Parameter Estimation . . . 120
7.2.4 Complete safe dMPC under artificial scarcity . . . 125
7.3 Numerical experiment . . . 125
7.3.1 Applying Resilient Primal Decomposition-based dMPC under Artificial Scarcity (RPdMPC-AS) . . . 126
7.4 Conclusion . . . 129

8 Conclusion 131
8.1 Retrospective and Contributions . . . 131
8.1.1 Contributions . . . 132
8.1.2 Shortcomings . . . 132
8.2 Possible Future directions . . . 133

Appendices 135

A Résumé étendu en français 137
A.1 Contexte et Motivation . . . 137
A.1.1 Contributions . . . 140
A.1.2 Publications . . . 140
A.1.3 Code . . . 140
A.2 Commande Prédictive et sa décomposition . . . 141
A.2.1 MPC pour des systèmes linéaires . . . 141
A.2.2 Décomposition de problèmes d'optimisation . . . 143
A.3 Differents topologies . . . 145
A.3.1 Par distribution des unités de computation . . . 145
A.3.2 Par communication . . . 145
A.3.3 Par relations de pouvoir . . . 146
A.3.4 Par type de communication . . . 147
A.4 Comportements anormaux . . . 148
A.4.1 Security . . . 148
A.4.2 Vulnérabilités des CPS . . . 148
A.4.3 Attaques in CPS . . . 149

A.4.4 Attaques à la dMPC . . . 152
A.4.5 Sécurisant le CPS . . . 152
A.4.6 Stratégies Passives . . . 153
A.4.7 Stratégies Actives . . . 154
A.4.8 Détection . . . 154
A.4.9 Mitigation/Récupération . . . 155
A.5 Comparaison avec état de l'art . . . 156
A.6 Vulnérabilités de la decomposition primale . . . 158
A.6.1 La dMPC baséee sur la décomposition primale . . . 158
A.6.2 Attaque d'intérêt . . . 160
A.7 Commande Prédictive résiliente pour systèmes dépourvus . . . 162
A.7.1 Systèmes dépourvus sous attaque . . . 162
A.7.2 Systèmes dépourvus . . . 162
A.7.3 Analysises des problèmes locaux . . . 164
A.7.4 Analyse de la négociation . . . 164
A.7.5 Détection . . . 168
A.7.6 Estimation . . . 168
A.7.7 Technique de détection complète . . . 170
A.7.8 Mitigation . . . 171
A.7.9 La dMPC résiliente pour systèmes dépourvus . . . 174
A.7.10 Expériment Numérique . . . 175
A.8 Commande Prédictive résiliente sur pénurie artificielle . . . 180
A.8.1 Relaxant le problème . . . 180
A.8.2 L'impacte sur la solution des problèmes locaux . . . 180
A.8.3 L'impact sur la négociation . . . 184
A.8.4 Mitigation et pénurie artificielle . . . 189
A.8.5 Détection . . . 190
A.8.6 Estimation de paramètres multiples . . . 190
A.8.7 Adaptation pour le problème étudié . . . 191
A.8.8 La dMPC résiliente utilissant la pénurie artificielle . . . 194
A.8.9 Expériment Numérique . . . 194
A.9 Conclusion . . . 199
A.9.1 Rétrospective et Contributions . . . 199
A.9.2 Contributions . . . 199
A.9.3 Inconvénient . . . 199
A.10 Directions Futures . . . 200

Bibliography **200**

List of Acronyms

ADMM	Alternating Direction Method of Multipliers.
CEM	Classification Expectation Maximization.
CIA	Confidentiality, Integrity and Availability.
CPS	Cyber-Physical Systems.
DDD	Disclosure, Deception and Denial of Service.
DDoS	Distributed Denial of Service.
DHN	District Heating Network.
dMPC	distributed Model based Predictive Control.
DoS	Denial of Service.
DT	Discrete Time.
EFT	Electronic Funds Transfer.
EM	Expectation Maximization.
FDI	False Data Injection.
FTC	Fault Tolerant Control.
GPU	Graphical Processing Unit.
HMI	Human Machine Interface.
HVAC	Heating, ventilation, and air conditioning.
IOT	Internet of Things.
KKT	Karush-Kuhn-Tucker.
LP	Linear Programming.
LS	Least Squares.
LTI	Linear Time-Invariant.
LTICT	Linear Time-Invariant Continuous Time.
LTIDT	Linear Time-Invariant Discrete Time.
MIQP	Mixed Integer Quadratic Programming.
MPC	Model-based Predictive Control.
PLC	Programmable Logic Controller.

PWA Piecewise Affine.

QoS Quality of Service.
QP Quadratic Programming.

RHS Receding Horizon Strategy.
RLS Recursive Least Squares.
RPdMPC-AS Resilient Primal Decomposition-based dMPC under Artificial Scarcity.
RPdMPC-DS Resilient Primal Decomposition-based dMPC for deprived systems.

SCADA Supervisory Control and Data Acquisition.
sEM stochastic Expectation Maximization.
SISO Single Input Single Output.

TCP Transmission Control Protocol.

WDN Water Distribution Network.

List of Figures

1.1 Centralized heat provider with glowing light. . . . 22
1.2 Exchange between local controllers and coordinator. . . . 23
1.3 Plot of local and global loss functions from last 10 weeks. . . . 24

3.1 Decomposition of sub-problems (P_i) and sub-systems (S_i). . . . 42
3.2 A vertex/node. . . . 43
3.3 Arcs and edges. . . . 43
3.4 Directed and undirected graphs with 7 vertices. . . . 43
3.5 Neighbor subgraph of vertex 6 (in red). . . . 44
3.6 A complete graph with 6 vertices. . . . 44
3.7 A path between vertices 1 and 5 (in red). . . . 44
3.8 A disjoint graph. . . . 45
3.9 Hierarchic and anarchic topologies. . . . 46
3.10 Sequential and parallel topologies. . . . 46
3.11 A polytree. . . . 47
3.12 Ring and small-world graphs. . . . 48

4.1 General abstraction of CPS. . . . 54
4.2 General abstraction of networked CPS. . . . 55
4.3 Attacks in networked CPS. . . . 56
4.4 3 dimensional attack space. . . . 56
4.5 Attack space and some attacks. . . . 57
4.6 Covert agent. . . . 58

5.1 Exchange between local controllers and coordinator in Primal decomposition-based dMPC. . . . 73
5.2 Evolution of variables during iterative process and control. . . . 75
5.3 Evolution of $\boldsymbol{\lambda}_i$ for different k when agent 2 is hijacked. . . . 76
5.4 Evolution of state $x_i[k]$ and reference $w_i[k]$ during control when agent 2 is hijacked. . . . 77
5.5 Accumulate objective functions for different values of τ_3. . . . 78
5.6 Evolution of $\boldsymbol{\lambda}_i$ during negotiation for $k = 5$ with different values of τ_3. . . . 79

6.1 Original minimum. . . . 95
6.2 Minimum after attack. . . . 95

6.3 Optimal value after ignoring attacker. 96
6.4 Optimal value after trying to recover original behavior. 97
6.5 Exchange between agents and coordinator in RPdMPC-DS. 98
6.6 District with 4 houses. 99
6.7 Thermic Model 3R-2C of a house. 99
6.8 Temperature in house I and the variable $E_I(k)$ for different scenarios. . . . 102
6.9 Air temperature in all houses for different scenarios. 103

7.1 Two constraints partitioning $\boldsymbol{\theta}_i$ solution space. 109
7.2 Set of constraints with no intersection. 111
7.3 Two constraints with $\langle_{\boldsymbol{\eta}_1}^{\boldsymbol{\eta}_2} = 180^o$. 111
7.4 A 3-sided polyhedron. 111
7.5 Effects of different radii r when generating points close to $\overset{\varnothing}{\boldsymbol{\theta}}_i$. 119
7.6 Gaussian Mixture for a 1D Piecewise Affine (PWA) function with 2 modes. 123
7.7 Air temperature in house I and the decision variable $E_I[k]$ for different scenarios. 127
7.8 Air temperature in all houses for different scenarios. 128
7.9 Control applied in all houses for the 3 scenarios. 129

A.1 District Heating Network (DHN) avec 4 maisons. 138
A.2 Échange entre contrôleurs locaux et coordinateur. 138
A.3 Graphique des fonctions objectifs dans une période de 10 semaines. 139
A.4 Décomposition du problèmes en (P_i) et du système en (S_i). 145
A.5 Topologies hierarchiques et anarchiques. 146
A.6 Topologies séquentiel et parallèle. 147
A.7 Abstraction Générale d'un CPS dans un réseau. 149
A.8 Attaques à un CPS dans un réseau. 150
A.9 Espace d'attaques avec 3 axes. 151
A.10 Espace d'attaques peuplé. 151
A.11 Échange entre agents et coordinateur pendant une dMPC basée sur la décomposition primale. 159
A.12 Évolution des variables pendant procès itératif. 160
A.13 Accumulate objective functions for different values of τ_3. 161
A.14 Evolution of $\boldsymbol{\lambda}_i$ during negotiation for $k = 5$ with different values of τ_3. . . 161
A.15 Minimum Original. 172
A.16 Minimum après attaque. 172
A.17 Nouveau optimal quand négligent l'attaquant. 173
A.18 Valeur optimale après récuperer comportement original. 173
A.19 Échange entre agents et coordinateur dans la RPdMPC-DS. 174
A.20 District avec 4 maisons. 175

A.21 Modèle thermique 3R-2C d'une maison. 175
A.22 Température de la maison I et la variable $E_I(k)$ pour les 3 scénarios. . . . 178
A.23 Temperature de l'air dans toutes les maisons pour les 3 scénarios. 179
A.24 Deux contraintes qui partionnent l'espace de $\boldsymbol{\theta}_i$. 181
A.25 Ensemble de contraintes avec intersection nulle. 183
A.26 Deux contraintes où $\langle^{\boldsymbol{\eta}_2}_{\boldsymbol{\eta}_1} = 180^o$. 183
A.27 Un polyèdre de 3 faces. 184
A.28 Effets de la génération avec différent rayons r. 190
A.29 Mixture gaussianne pour PWA d'une dimension. 192
A.30 Température de la maison I et la variable $E_I[k]$ pour les 3 scénarios. 196
A.31 Température de l'air dans toutes les maisons pour les 3 scénarios. 197
A.32 Commande appliquée en toutes les maisons pour les 3 scénarios. 197

List of Tables

4.1 Comparison between adjacent works 67

6.1 Model Parameters 100
6.2 Parameters for each agent 100
6.3 Objective functions J_i (% error) 102

7.1 Objective functions J_i (% error). 128

A.1 Comparaison entre travaux adjacents 157
A.2 Paramètres du Modèle 176
A.3 Paramètres pour chaque maison 176
A.4 Fonctions objectif J_i (% erreur) 178
A.5 Fonctions objectif J_i (% erreur). 198

NOTATION

— Scalars are denoted by normal-faced minuscule characters (a)

— Column vectors are represented by bold-faced characters ($\boldsymbol{x}$, $\boldsymbol{U}$)

— Matrices are denoted by normal-faced capital characters (A)

— Sets are denoted by calligraphic capital characters ($\mathcal{A}$).
Some special sets are denoted using blackboard bold-faced capital characters ($\mathbb{N}$)

— For matrix calculus, we use the numerator-layout convention such as $\frac{\partial A\boldsymbol{x}}{\partial \boldsymbol{x}} = A$.

— For boolean algebra, we use 1 and 0 to represent true and false, respectively.

Set Theory

$\varnothing$	Empty set
$\forall\ \exists$	Universal quantifier, Existential quantifier
$(x_1, \ldots, x_n)$	An ordered n-tuple
$:\quad (\mid)$	Such that (in set builder)
$\{\ldots\}$	Set builder
$\{\cdot \mid \ldots\}$	Set builder defined by predicate.
$\setminus\ \cup\ \cap$	Set difference, union and intersection, respectively
$\in\ \subset\ \subseteq$	Set membership, proper subset, subset, respectively
$\ni\ \supset\ \supseteq$	And their respective converses
$\times$	Cartesian product, $\mathcal{S}_1 \times \cdots \times \mathcal{S}_n = \{(s_1, \ldots, s_n) \mid s_i \in \mathcal{S}_i, \forall i \in \{1 : n\}\}$
$\mathcal{S}^n$	Cartesian power, $\mathcal{S}^n = \underbrace{\mathcal{S} \times \cdots \times \mathcal{S}}_{n} = \{(s_1, \ldots, s_n) \mid s_i \in \mathcal{S}, \forall i \in \{1 : n\}\}$
$\#\,\mathcal{A}$	Cardinality of the set $\mathcal{A}$
$\mathbb{Z}$	Set of integer numbers
$\mathbb{R}$ ($\mathbb{R}_+$)	Set of (nonnegative) real numbers
$\mathbb{N}$ ($\mathbb{N}_+$)	Set of (nonnegative) natural numbers
$\mathbb{N}_\varnothing$	Set of natural numbers without zero

Linear Algebra

$\boldsymbol{v}_{(i)}$	The i-th element of vector $\boldsymbol{v}$
$M_{(i,j)}$	The element in the i-th row and j-th column of matrix M
$M_{(i,*)}$ $M_{(*,j)}$	Row i and column j of matrix M
end	The last element of row or column of the entity.
$0_{m\times n}$ $1_{m\times n}$	0 and 1 filled matrices of order $m \times n$ (if indices are suppressed, size implicit by other elements)
$\mathbf{0}_m$ $\mathbf{1}_m$	0 and 1 filled vectors of size m (if index is suppressed size implicit by other elements)
I_m	Identity of order $m \times m$ (if index suppressed, size implicit by other elements)
$\mathbb{R}^{m\times n}$	Set of real matrices of order $m \times n$
$\mathbb{S}$ ($\mathbb{S}^n$)	Set of symmetric real matrices (of order n)
$\mathbb{S}^n_+$ $\mathbb{S}^n_{++}$	Set of positive semidefinite and positive definite matrices of order n
$\prec \preceq$	Component-wise less and less or equal for vectors and matrix inequality for matrices in $\mathbb{S}$
$\succ \succeq$	And their converses
$\#\,\boldsymbol{x}$	Number of elements in vector, i.e., if $\boldsymbol{x} : \mathbb{R}^n$, then $\#\,\boldsymbol{x} = n$
$\#\,A$	Size of matrix A, i.e., if $A : \mathbb{R}^{m\times n}$, then $\#\,A = m \times n$
$[\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]$	Horizontal concatenation
$[\boldsymbol{x}_1; \ldots; \boldsymbol{x}_n]$	Vertical concatenation
$\boldsymbol{x}'$ A'	Transposes of vector $\boldsymbol{x}$ and matrix A
$\oslash$	Block diagonal concatenation, i.e., $A \oslash B = \left(\begin{smallmatrix} A & 0 \\ 0 & B \end{smallmatrix}\right)$, and $\bigoslash_{i=1}^{M} A_i = \begin{pmatrix} A_1 & 0 & 0 & 0 \\ 0 & A_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \cdots & A_M \end{pmatrix}$
$\langle \boldsymbol{x}, \boldsymbol{y} \rangle$	Inner product between vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, same as $\boldsymbol{x}'\boldsymbol{y}$
$\circ$	Hadamard product, i.e., for $A, B \in \mathbb{R}^{m\times n}$, $(A \circ B)_{(i,j)} = A_{(i,j)}B_{(i,j)}$
$\otimes$	Kronecker product, i.e., $A \otimes B = \begin{pmatrix} A_{(1,1)}B & \cdots & A_{(1,n)}B \\ \vdots & \ddots & \vdots \\ A_{(m,1)}B & \cdots & A_{(m,n)}B \end{pmatrix}$
$\dagger$	Pseude-inverse such $A^\dagger = (A^TA)^{-1}A^T$
$\mathrm{vec}(A)$	Vectorizes matrix $A \in \mathbb{R}^{m\times n}$, such $\mathrm{vec}(A) = [A_{(1,*)}, \ldots, A_{(n,1)}]^{\mathrm{T}}$
$\lvert\cdot\rvert$ $\lVert\cdot\rVert_n$ $\lVert\cdot\rVert_F$	Absolute value, and ℓ_n (ℓ_2 if index suppresed) and Frobenius norms
$\lVert\boldsymbol{v}\rVert_Y$	Weighted norm $\left\lVert Y^{\frac{1}{2}}\boldsymbol{v}\right\rVert$, where $Y \in \mathbb{S}$
$\mathrm{Proj}^{\mathcal{T}}(\cdot)$	Euclidean projection onto set $\mathcal{T}$
$\langle^{\boldsymbol{x}}_{\boldsymbol{y}}$	Angle between vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, i.e., $\langle^{\boldsymbol{x}}_{\boldsymbol{y}} = \arccos \frac{\langle \boldsymbol{x},\boldsymbol{y}\rangle}{\lVert x\rVert\lVert y\rVert}$

General Operators, functions, and relations

$A \Rightarrow B$	A implies B, i.e., if A is true than B must also be true
$A \Leftrightarrow B$	A implies B and B implies A, i.e., A is true if and only if (iff) B is true
$A \vee B$	Logical inclusive or operation, i.e., true if A or B are true.
$A \wedge B$	Logical and operation, i.e., true if A and B are true.
$\neg A$	Logical not, i.e., true if A is false.
$f : \mathcal{A} \to \mathcal{B}$	A $\mathcal{B}$-valued function on some subset of $\mathcal{A}$, its domain
$\operatorname{dom} f$	Domain of f
$f(x)$	The value of function f at x, $x \in \operatorname{dom} f$
$f : x \mapsto f(x)$	The mapping of element $x \in \operatorname{dom} f$ in function f
$\mathbb{1}_{\{x\}}$	Indicator function returning 1 if x is true and 0 otherwise.
$\nabla_x f(x)$	Gradient of $f(x)$ with respect to x.
$\lceil x \rceil$ $\lfloor x \rfloor$	Ceiling and floor functions
$\operatorname{fix} : \mathbb{R} \to \mathbb{Z}$	$\operatorname{fix} : x \mapsto \begin{cases} \lfloor x \rfloor & \text{if } x \geqslant 0 \\ \lceil x \rceil & \text{if } x < 0 \end{cases}$
$n : i : j$	Row vector builder with elements $\{n, n+i, \ldots, n+mi\}$ where $m = \operatorname{fix}(\frac{j-n}{i})$
$n : j$	Similar to $n : 1 : j$

Control Systems

$\boldsymbol{v}_i$ M_i	Vector $\boldsymbol{v}$ and matrix M of i-th agent
$\boldsymbol{v}[i\|k]$	Prediction of vector $\boldsymbol{v}$ for time $k+i$ predicted at time k
$\boldsymbol{v}[0 : i\|k]$	Sequence of predictions of vector $\boldsymbol{v}$ for times $k : k+i$ predicted at time k
$\boldsymbol{v}^{(p)}$	Vector $\boldsymbol{v}$ at the p-th iteration in an algorithm

Chapter 1

Introduction

Welcome back my friends
to the show that never ends
We're so glad you could attend
Come inside! Come inside!

Karn Evil 9: 1st Impression, Pt. 2
Emerson, Lake & Palmer

The increase in computation power and dissemination of computing devices in recent years ($\sim$ 15 years) is undeniable. Devices with the same (or greater) computation power that took man to the moon are within a hand's reach today.

With this increase in computation power, we can now solve some problems that were not solvable in a reasonable time in the past. Thus the *renaissance* of some optimization-based methods, such as Neural networks and Artificial Intelligence.

For other more common methods, such Model-based Predictive Control (MPC) [GPM89], this development meant solving more significant problems in less time. Some of those problems are solved even in real-time [BLM08], using computation units smaller than coins [BM14].

Consequently, the list of possible applications of MPC increased. Instead of using it on a single piece of equipment in petrochemical plants with large time scales, we can use it to control systems with the size of districts and cities such as Water Distribution Networks (WDNs) [Zha+21b] and DHNs [Tay+21]. This change presents the gradual insertion of MPC into CPS, in which computers and physical machines are tightly coupled.

Nevertheless, the calculation with multiple decision variables may still be expensive for some large-scale systems. To ease the computation, we need some artifices, such as distributing it into different computation units and making them communicate. As we will see, this strategy is called dMPC, and it comes in many flavors.

However, only a few studies have been made on the security of the dMPC strategies when units do not communicate honestly. This work, however, studies what happens when computation units do not work cooperatively in a dMPC framework.

To give a perspective to a general reader on how it can affect daily life, we present a simple qualitative example in the form of a story. The conclusion will serve as the main takeaway of the possible effects of non-cooperative behaviors.

1.1 When greediness backfires

Six o'clock on a Christmas morning. You wake up. Shivering. The tips of your fingers are blue, and you ask yourself why you accepted to make part of this housing project.

The housing project was set in an undisclosed place to test a new sustainable DHN. Engineers from around the world were invited to take part in the project for one year to test drive the DHN and file bugs.

Nourishment and all other amenities would be paid for. So you willingly accepted. What could go wrong?

It seemed okay-ish until 6 weeks ago, when the days became shorter and shorter and the air colder and colder.

You climb out of bed and look at your house's central temperature control indicator. The outside temperature is -5^oC, and the mean temperature inside is 7^oC. The thermostat is still on, indicating that the comfortable 23^oC you set yesterday was still saved. However, you see a flashing dim orange light in the corner of your eye. A small LED panel indicates:

ERR706

Something is wrong. This is the first time you have seen this message, and it is the last week of the project. You turn around, looking for the manual given on your first day. Nada!

You take your coat, put on some gloves, and look outside your window. You observe the centralized heat provider unit in the middle of the *cul-de-sac* you live.
A similar orange light is flashing.

Figure 1.1 – Centralized heat provider with glowing light.

You go outside to see if there is anything you can do. Apparently, engineers think alike, and all of your neighbors are gathered around the roundabout. One of your neighbors has in her hands an open manual and reads it:

"ERR706": Consensus not reached.
Restart the system and verify communication.

The DHN in question has a controller that solves an optimization problem to allocate the energy for each house. However, some privacy questions were raised during its design, and the designers decided:

"The controller must not know the exact needs of each resident!"

After some discussion, they opted for a distributed method. Each house was given one local controller to manage its temperature, and another controller with a coordinator role was added to iteratively propose allocations for each building.

Just above the error code in the manual, you can take a glimpse of a diagram followed by a small text indicating the communication and the flux of information:

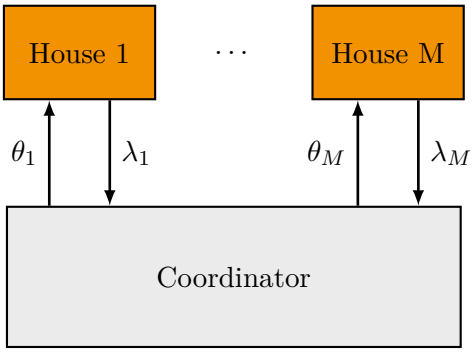


Figure 1.2 – Exchange between local controllers and coordinator.

> Allocations θ_i are proposed for each house by the coordinator. The local controllers respond with indices λ_i to indicate satisfaction. Allocations are updated based on these indices until a consensus is reached.

Today was not the case.

You and your friends verify the communication cables. Everything is intact. You restart the system. Lights turn off. Some seconds pass, and the orange LED turns on again. The same error.

You try to debug the problem. Fortunately, the program has an open data policy for the period of tests. You look into the historical data but have too much data to grasp something valuable. A friend gives an idea:

– What if we compare the same day of the week of the last ten weeks? We can accumulate a loss function that depends on the deviation from the setpoints. Small values would mean good performance.

"What a nice idea! Why didn't I think of this before?", You think.

– I don't know if it is really a good idea. Maybe it is some internal error in the coordinator program.

says your neighbor of house 1.

You open a terminal in the central computer, anyway. You can hardly feel the tips of your fingers touching the keyboard. You filter the logs. Calculate the deviations and compute the loss functions for each house. You use the parameters of each house given by the manual, kindly lent by your colleague. You also sum the functions for all houses giving a global loss function. You plot the points and smooth the curves.

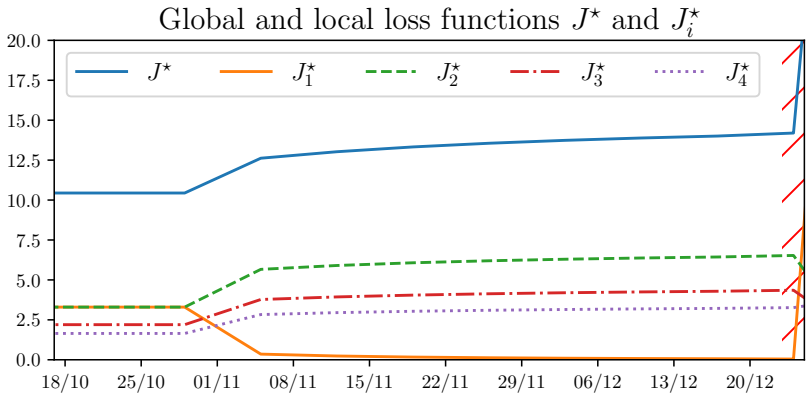


Figure 1.3 – Plot of local and global loss functions from last 10 weeks.

A-ha! Against facts, there are no arguments!

You see the global function increasing after a given day. The increase rate steadily decreased until yesterday, when everything went awry. You highlight it by hatching the area in red. Your house, the house number 4, has increased its function. And all the other houses too. Wait! House 1's function decreases. Well, well, well ... You show the plots to your neighbors, who afterward confront the resident of house number 1. After the indubitable proof, he finally breaks down and tells the truth.

> Okay, okay. It was me. Some time ago, after studying the exchanges between my house and the coordinator, I realized something. When my house sent increasing values of λ_i, the following allocations θ_i were greater. So I changed my circuit by adding an amplifier to increase my resources. You guys know I'm not used to these cold temperatures. I began with a small amplification. And as no one noticed and the days were getting colder, I increased the amplification every day a bit. Everything went smoothly until today. And I would have gotten away with it, too, if it weren't for you meddling fellow engineers!

The greediness of the neighbor shifted the negotiation so he could receive more resources. Still, he was so greedy that the system could not find a consensus at some point.

You and your friends remove the said amplifier, file a complaint to the program organizers against the unethical colleague, and file a bug, showing the system's vulnerability.

After this day, you decide to make the system more resilient to such attacks, preventing their effects. But first, you must catch up on the lost sleep hours and study the security of CPS.

1.2 Motivation and Contributions

Although the example given is ludic, it presents possible effects of attacks and malfunction in a CPS: Loss of optimality and eventually breakdown of the control strategy.

While one of them can sometimes be acceptable (depending on the suboptimality), the other is absolutely inadmissible, principally when we are talking about CPS that are essential for the population. The consequences of non-nominal behaviors may impact important parts of the life of the population and can even endanger it.

A bibliometric study [Zac+19] shows how awareness of the security of CPS has been raised. This awareness was sadly motivated by many examples of attacks and faults all around the world in the last two decades. Some examples are the Brazilian blackouts in 2010 [Con10], the Stuxnet attack on an Iranian uranium enrichment plant in 2011 [Lan11], attacks in Ukrainian Power System in 2016 [Bin17] and other [Din+18; Dib+19].

The main motivation of this work is the security of CPS controlled by dMPC. We focus on the cases where the systems are attacked by ill-intentioned agents who change some variables for their own benefit. As we will see, these attacks are called **False Data Injection (FDI) attacks**.

And using the knowledge gathered throughout the study, we want to be able to reduce the effects of the attacks as much as possible. And to guide us on our quest, we raise some questions:

— **Can we detect the attack?**
— **Can we identify the ill-intentioned agent(s)?**
— **Can we mitigate the effects of the attack?**

This work has as its objective to answer these questions, at least for specific cases which will be formally presented.

To understand and answer those questions, we divide this work into two parts.

Part I It serves as a gentle introduction to dMPC design, security, and attacks in CPS and the choices in this work for each one of those steps. [1]

For an unfamiliar reader, Chapter 2 explains what is a MPC to begin with and shows some of the challenges of decomposing it. Chapter 3 discusses possible topologies used for those decompositions. In Chapter 4, we define anomalous behaviors, categorize them and present some methods used in the literature to prevent and combat them.

1. A reader accustomed with the theme is referred to Part II.

Part II It contains the contributions of this work.

We begin with a brief recapitulation of the choices made in each of the chapters in Part I and a comparison between the methods in this work and the most adjacent works in the literature.
Chapter 5 presents the decomposition studied in this work (primal decomposition). We present its vulnerabilities and how they can affect the performance of the system. Here we formalize the example given in the story, giving it a more quantitative approach. Once we know the vulnerabilities and possible effects of attacks, we divide the mitigation problem into more manageable parts.
First, in Chapter 6, we analyze an unsophisticated problem, so we can assess the difficulties that can be found when solving the mitigation problem. From the analysis of the problem, we propose a detection and mitigation mechanism, followed by an academic example to illustrate its functioning.
Then, in Chapter 7, we analyze a similar problem but with a twist. As we will see, just one small modification to the initial problem can increase its complexity exponentially. From the analysis of the newly-found problem, we propose a similar strategy but with adequate modifications to contain the exponential nature of the problem.
We conclude this work in Chapter 8 with a discussion about the results found during this study, benefits as well as shortcomings. Some of this discussion leads to open questions that can incite new works.

Contributions

We can list as a minor contribution of this work the study of vulnerabilities of Primal decomposition-based dMPC, which wasn't studied to the best of our knowledge, and as main contributions, two mitigation strategies for different kinds of systems with increasing complexities.

The first strategy, for systems where the total resources available are not enough to satisfy local demands, we call this systems **Deprived Systems**.

The second one is for an exponentially more complex class of systems, where we assume the local demands respect at least some constraints, which we can use to get scarcity information by means we call **artificial scarcity**.

1.3 Publications

The work and discussion presented in this thesis yielded the following publications

[NBG21] Conference article for the SysTol'21

[Nog+22] Conference article for the NecSys'22

1.4 Programming and material

All the simulations were done using a PC with

Processor Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz

Memory 15GiB System memory

under a GNU/Linux system based on architecture x86_64 with

Linux kernel version 5.4.0-73-generic

Ubuntu distribution version #82 18.04.1-Ubuntu SMP Fri Apr 16 15:10:02 UTC 2021

They were executed on

MATLAB version R2019b Update 3 (9.7.0.1261785) 64-bit (glnxa64).

The graphs were then generated using

python version 3.6.9

with packages:

matplotlib version 3.2.2

numpy version 1.19.0

scipy version 1.3.2

Images and diagrams were created using

TikZ version 3.1.9a

Inkscape version 0.92.3

Blender version 3.3.1

All source code of this work is available at `https://github.com/Accacio/thesis/`

Part I

Model Predictive Control

Decomposition and Security

Chapter 2

Decomposing the Model Predictive Control

The mystery of the universe
is not time
but size.

The Gunslinger
Stephen King

This chapter presents the Model Predictive Control strategy and some considerations needed to decompose it.

Contents

2.1 Model Predictive Control 31
2.1.1 General discrete MPC 32
2.1.2 Convex MPC (quadratic case) 33
2.2 Optimization Decomposition Frameworks 35
2.2.1 Decomposable problems 35
2.2.2 Uncoupled problems 36
2.2.3 Coupled problems 37
2.2.4 Generalizing 39
2.2.5 Decomposition techniques 40

2.1 Model Predictive Control

Model-based Predictive Control is a closed-loop control strategy based on the solution of optimization problems. Once given a system model and an objective function, this strategy uses the system model to predict future states and compute a control sequence that optimizes the given objective function. Since the solution is found through optimization problems, it is natural to add some restrictions on the system, usually defined as (in)equality constraints.

The fact that we may effortlessly add constraints to the specifications gave it a special place in the industry, where it is used in a plethora of applications (energy

management [Ana+18], water distribution [Zha+21b], control of quadrotors [BM14] and others).

2.1.1 General discrete MPC

As it is generally implemented through a digital computer and the transmission of continuous signals can potentially demand infinite bandwidth [HML22], it is natural to assume the system to be modeled with Discrete Time (DT) dynamics

$$\boldsymbol{x}[k+1] = f(\boldsymbol{x}[k], \boldsymbol{u}[k]), \tag{2.1}$$

where $\boldsymbol{x}[k] \in \mathbb{R}^{n_x}$ is the state of the system and $\boldsymbol{u}[k] \in \mathbb{R}^{n_u}$ is the input of the system.

The system can be under constraints

$$h(\boldsymbol{x}[k], \boldsymbol{u}[k]) \preceq \mathbf{0}, \tag{2.2}$$

with $h : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_c}$. Observe that mathematically equality constraints can be represented by a couple of inequality constraints, although for computer implementation, it can lead to poor numerical conditioning, resulting in unexpected behavior [BBM17].

Given an objective function $J : \mathbb{R}^{n_x} \times \mathbb{R}^{N \cdot n_u} \to \mathbb{R}$, we can optimize this objective for a given horizon $\mathcal{N} = \{1 : N\}$ to obtain a control sequence $\boldsymbol{U}^\star[k] = [\boldsymbol{u}^\star[0|k]; \ldots; \boldsymbol{u}^\star[N-1|k]]$. To calculate $\boldsymbol{U}^\star[k]$, we use states and input predictions of time $k+i$ calculated in a time k, represented by $\boldsymbol{x}[i|k]$ and $\boldsymbol{u}[i|k]$. The problem solved to calculate the control sequence is

$$\begin{aligned} \underset{\boldsymbol{U}[k]}{\text{minimize}} \quad & J(\boldsymbol{x}[0|k], \boldsymbol{U}[k]) \\ \text{subject to} \quad & \left.\begin{aligned} \boldsymbol{x}[i|k] &= f(\boldsymbol{x}[i-1|k], \boldsymbol{u}[i-1|k]) \\ h(\boldsymbol{x}[i-1|k], \boldsymbol{u}[i-1]) &\preceq \mathbf{0} \end{aligned}\right\} \forall i \in \mathcal{N} \end{aligned} \tag{2.3}$$

After solving this problem, we have an optimal control sequence $\boldsymbol{U}^\star[k]$ and only the first element $\boldsymbol{u}^\star[0|k]$ is applied in the system. The process is repeated for $k+1$ following a Receding Horizon Strategy (RHS).

This formulation is as general as possible. However, depending on the problem's convexity, it can be hard to solve, especially online. Convex problems are the most widely studied, and strategies to solve them are extensively diffused. Some of those problems even have explicit solutions [BV04].

2.1.2 Convex MPC (quadratic case)

The MPC community favors the use of convex problems when possible. Two families of convex problems broadly used are Quadratic Programming (QP) and Linear Programming (LP). In this work, we concentrate on QP problems, which have quadratic objective functions and affine or linear constraints. Numerous mathematical solvers are apt to solve this kind of problem directly or through equivalent problems. We can cite non-extensively some solvers such as MATLAB internal QP solvers [1], OSQP [2], MOSEK [3] and ECOS [4].

A commonly used quadratic objective function is

$$J(\boldsymbol{x}[0|k], \boldsymbol{U}[k]) = \sum_{i \in \mathcal{N}} \left[\|\boldsymbol{v}[i|k]\|_Q^2 + \|\boldsymbol{u}[i-1|k]\|_R^2 \right] \tag{2.4}$$

where $\boldsymbol{v}$ is a control objective, and $Q \in \mathbb{S}_+$ and $R \in \mathbb{S}_{++}$ are weighting matrices, which can represent the costs of each term of the equation. The ratio between these matrices describes the compromise between control signal energy and control objective.

The control objectives can be for example, *disturbance rejection*, where $\boldsymbol{v}[k] = \boldsymbol{x}[k]$, or *reference tracking*, where $\boldsymbol{v}[k] = \boldsymbol{w}[k] - \boldsymbol{x}[k]$, being $\boldsymbol{w}[k]$ a setpoint. Observe that this last example is for state reference tracking. For output reference tracking, the system output $\boldsymbol{y}[k]$ should be used instead of $\boldsymbol{x}[k]$, depending on the system an adequate relation between $\boldsymbol{y}[k]$ and $\boldsymbol{x}[k]$ can be found.

The predictions of $\boldsymbol{v}[k]$ and $\boldsymbol{w}[k]$ can be stacked as $\boldsymbol{V}[k] = [\boldsymbol{v}[1|k]; \ldots; \boldsymbol{v}[N|k]]$ and $\boldsymbol{W}[k] = [\boldsymbol{w}[1|k]; \ldots; \boldsymbol{w}[N|k]]$ and then (2.4) can be rewritten as

$$J(\boldsymbol{x}[0|k], \boldsymbol{U}[k]) = \|\boldsymbol{V}[k]\|_{\bar{Q}}^2 + \|\boldsymbol{U}[k]\|_{\bar{R}}^2, \tag{2.5}$$

where $\bar{Q} = I_N \otimes Q$ and $\bar{R} = I_N \otimes R$.

For the constraints in (2.3) to be affine or linear, first, we suppose the system is linear, and we use a Linear Time-Invariant (LTI) model

$$\begin{aligned} \boldsymbol{x}[k+1] &= A\boldsymbol{x}[k] + B\boldsymbol{u}[k] \\ \boldsymbol{y}[k] &= C\boldsymbol{x}[k] \end{aligned} . \tag{2.6}$$

In this work, we concentrate in **input constraint systems** whose constraints do not depend on the system state, so we drop the $\boldsymbol{x}[k]$ terms in h, and since it is affine or linear, we rewrite it as

$$\Gamma \boldsymbol{u}[k] \preceq \boldsymbol{u}_{\max}, \tag{2.7}$$

with Γ of adequate size $\mathbb{R}^{\#\boldsymbol{u}_{\max} \times n_u} = \mathbb{R}^{n_c \times n_u}$.

1. https://fr.mathworks.com/help/optim/ug/quadprog.html
2. https://osqp.org
3. https://www.mosek.com
4. https://github.com/embotech/ecos

Equation (2.7) can be vectorized for a horizon in $\mathcal{N}$

$$\bar{\Gamma}\boldsymbol{U}[k] \preceq \boldsymbol{U}_{\max}, \tag{2.8}$$

with

$$\bar{\Gamma} = I_N \otimes \Gamma \tag{2.9}$$

$$\boldsymbol{U}_{\max} = \mathbf{1}_N \otimes \boldsymbol{u_{max}}. \tag{2.10}$$

For the control objectives, henceforth we only consider *reference tracking*, since *disturbance rejection* can be described as a *reference tracking* when $C = I_{n_x}$ and $\boldsymbol{w}[k] = \mathbf{0}_{n_x}$.

Putting it all together, we have

$$\begin{aligned} \underset{\boldsymbol{U}[k]}{\text{minimize}} \quad & \|\boldsymbol{V}[k]\|_{\bar{Q}}^2 + \|\boldsymbol{U}[k]\|_{\bar{R}}^2 \\ \text{subject to} \quad & \boldsymbol{x}[i|k] = A\boldsymbol{x}[i-1|k] + B\boldsymbol{u}[i-1|k] \quad \forall i \in \mathcal{N} \quad . \\ & \bar{\Gamma}\boldsymbol{U}[k] \preceq \boldsymbol{U}_{\max} \end{aligned} \tag{2.11}$$

If we opt for a Batch Approach [BBM17, Chapter 8.2] to solve the problem, we can rewrite the equalities in (A.4) compactly as

$$\underbrace{\begin{bmatrix} \boldsymbol{y}[1|k] \\ \boldsymbol{y}[2|k] \\ \vdots \\ \boldsymbol{y}[N|k] \end{bmatrix}}_{\boldsymbol{Y}[k]} = \underbrace{\begin{bmatrix} CA^1 \\ CA^2 \\ \vdots \\ CA^N \end{bmatrix}}_{\mathcal{Y}^x} \boldsymbol{x}[0|k] + \underbrace{\begin{bmatrix} CA^0B & 0 & \dots & 0 \\ CA^1B & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ CA^{N-1}B & \dots & \dots & CA^0B \end{bmatrix}}_{\mathcal{Y}^u} \boldsymbol{U}[k] \tag{2.12}$$

and substitute them in the objective function, yielding the QP problem, which implicitly respects these constraints

$$\begin{aligned} \underset{\boldsymbol{U}[k]}{\text{minimize}} \quad & \|\boldsymbol{U}[k]\|_H^2 + 2\boldsymbol{f}[k]^T\boldsymbol{U}[k] + c[k] \\ \text{subject to} \quad & \bar{\Gamma}\boldsymbol{U}[k] \preceq \boldsymbol{U}_{\max} \end{aligned} \tag{2.13}$$

where

$$H = \|\mathcal{Y}^u\|_{\bar{Q}}^2 + \bar{R} \tag{2.14}$$

$$\boldsymbol{f}[k] = \mathcal{Y}^{uT}\bar{Q}(\mathcal{Y}^x\boldsymbol{x}[0|k] - \boldsymbol{W}[k]) \tag{2.15}$$

$$c[k] = \|\mathcal{Y}^x\boldsymbol{x}[0|k]\|_{\bar{Q}}^2 - 2\boldsymbol{W}[k]^T\bar{Q}\mathcal{Y}^x\boldsymbol{x}[0|k] + \|\boldsymbol{W}[k]\|_{\bar{Q}}^2 . \tag{2.16}$$

By dividing the objective function by 2 and ignoring the constant term $c[k]$, we have a problem structured in the standard QP form, which can easily be used in the supra-cited

solvers

$$\begin{array}{ll} \underset{\boldsymbol{U}[k]}{\text{minimize}} & \frac{1}{2}\left\|\boldsymbol{U}[k]\right\|_H^2 + \boldsymbol{f}[k]^T\boldsymbol{U}[k] \\ \text{subject to} & \bar{\Gamma}\boldsymbol{U}[k] \preceq \boldsymbol{U}_{\max} \end{array} . \tag{2.17}$$

Remark 2.1

Observe that even though the problems (2.17) and (2.13) are equivalent (same solution), they are not the same problem. So, to obtain the correct objective function value, we still need to recalculate the function in (2.13) using the optimal $\boldsymbol{U}^\star[k]$ found.

Although the solution to such problems can be straightforward (solvers are widely known), and the problems are well structured, in this **monolithic** form, these problems can be computationally intensive depending on n_x, n_u, n_c and N. For large-scale systems such as WDNs [Zha+21b], and DHNs [Tay+21], the computation time and memory needed to solve their associated problems may increase drastically. We can use some decomposition methods to make the solution to such problems more tractable. For instance, we can decompose the parts of different time-scales [Che+11], use game theory [MMC11] or even genetic algorithms paired with state observers [Xie+16]. In this work, we will focus on decomposition techniques based on optimization decomposition, such as in [Gis+13].

2.2 Optimization Decomposition Frameworks

In this section, we briefly describe the optimization decomposition frameworks finding what they have in common so we can decompose the MPC problem proposed. As seen in [BV04], maximization problems can be rewritten as minimization problems, so we use only minimization problems henceforth.

2.2.1 Decomposable problems

As shown in [Con+06] and [Boy+15], a **decomposable optimization problem** has more than 1 decision variable and can be decomposed into at least 2 sub-problems. For simplicity, the examples in this section divide problems into two sub-problems. If the sub-problems variables can be partitioned, we can partition the original problem further.

The examples in this section are decomposed using the primal problem, which is the most direct method. Other methods exist and will be shortly commented on during this work.

Be the following problem a general decomposable problem, i.e., the decision variable $\boldsymbol{x}$

can be partitioned into two groups of variables $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$:

$$\begin{array}{ll} \underset{\boldsymbol{x}_1,\boldsymbol{x}_2}{\text{minimize}} & J(\boldsymbol{x}_1,\boldsymbol{x}_2) \\ \text{subject to} & f_i(\boldsymbol{x}_1,\boldsymbol{x}_2) \leqslant 0,\ i \in \mathcal{M} = \{1:m\} \\ & g_i(\boldsymbol{x}_1,\boldsymbol{x}_2) = 0,\ i \in \mathcal{P} = \{1:p\} \end{array} \tag{2.18}$$

Such a family of problems can be divided into two categories:

1. Uncoupled
2. Coupled

2.2.2 Uncoupled problems

The decomposition of uncoupled problems is straightforward and trivial, being so rare in the real world that it is frequently ignored.

For an uncoupled problem, the groups of variables $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are disjoint, the function $J(\boldsymbol{x})$ can be rewritten as $J(\boldsymbol{x}) = J_1(\boldsymbol{x}_1) + J_2(\boldsymbol{x}_2)$, and the constraints can be divided into 2 groups, one that depends only on $\boldsymbol{x}_1$ and other only on $\boldsymbol{x}_2$, i.e. $\mathcal{M}_1$ and $\mathcal{M}_2$ for the inequality constraints and $\mathcal{P}_1$ and $\mathcal{P}_2$ for the equality ones:

$$\begin{array}{lccc} \underset{\boldsymbol{x}_1,\boldsymbol{x}_2}{\text{minimize}} & J_1(\boldsymbol{x}_1) & + & J_2(\boldsymbol{x}_2) \\ \text{subject to} & f_i(\boldsymbol{x}_1) \leqslant 0,\ i \in \mathcal{M}_1 & & f_i(\boldsymbol{x}_2) \leqslant 0,\ i \in \mathcal{M}_2 \\ & g_i(\boldsymbol{x}_1) = 0,\ i \in \mathcal{P}_1 & & g_i(\boldsymbol{x}_2) = 0,\ i \in \mathcal{P}_2 \end{array} . \tag{2.19}$$

We can trivially rewrite (2.19) as

$$\underset{\boldsymbol{x}_1,\boldsymbol{x}_2}{\text{minimize}} \quad J_1(\boldsymbol{x}_1^\star) + J_2(\boldsymbol{x}_2^\star), \tag{2.20}$$

where $J_1(\boldsymbol{x}_1^\star)$ and $J_2(\boldsymbol{x}_2^\star)$ can be found by solving

$$J_1(\boldsymbol{x}_1^\star) = \begin{array}{ll} \underset{\boldsymbol{x}_1}{\text{minimize}} & J_1(\boldsymbol{x}_1) \\ \text{subject to} & f_i(\boldsymbol{x}_1) \leqslant 0,\ i \in \mathcal{M}_1 \\ & g_i(\boldsymbol{x}_1) = 0,\ i \in \mathcal{P}_1 \end{array} \tag{2.21a}$$

$$J_2(\boldsymbol{x}_2^\star) = \begin{array}{ll} \underset{\boldsymbol{x}_2}{\text{minimize}} & J_2(\boldsymbol{x}_2) \\ \text{subject to} & f_i(\boldsymbol{x}_2) \leqslant 0,\ i \in \mathcal{M}_2 \\ & g_i(\boldsymbol{x}_2) = 0,\ i \in \mathcal{P}_2 \end{array} \tag{2.21b}$$

Each problem in (2.21) can be solved in parallel and when solutions $\boldsymbol{x}_i^*$ are found for each sub-problem, the problem in (2.20) is automatically solved ($J_i(\boldsymbol{x}_i^\star)$ are constants).

2.2.3 Coupled problems

On the other hand, coupled problems are more frequent in the real world. In this category, we cannot solve the sub-problems entirely in parallel since the coupling implies exchanging information between sub-problems (each sub-problem needs information from others to solve itself). The coupling nature of the problem can be traced to two main sources

1. Complicating variables
2. Complicating constraints

We called them *complicating* because, without them, the solution would be trivial (uncoupled category shown in the last section) [Con+06].

Complicating variables

For this sub-category, we suppose the variables $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are not disjoint, i.e., there are some shared variables. To force disjointedness, we redivided $\boldsymbol{x}$ into 3 disjoint groups $\boldsymbol{x}_a$ (present only in sub-problem 1), $\boldsymbol{x}_b$ (present only in sub-problem 2), and $\boldsymbol{x}_{ab}$ (present in both sub-problems). We suppose $J(\boldsymbol{x})$ can be rewritten as $J(\boldsymbol{x}) = J_1(\boldsymbol{x}_a, \boldsymbol{x}_{ab}) + J_2(\boldsymbol{x}_b, \boldsymbol{x}_{ab})$, and the constraints can be divided into groups that depend on $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$, i.e., $\mathcal{M}_a$ and $\mathcal{M}_b$ for the inequality constraints and $\mathcal{P}_a$ and $\mathcal{P}_b$ for the equality ones:

$$\begin{array}{lccc} \underset{\boldsymbol{x}_a, \boldsymbol{x}_{ab}, \boldsymbol{x}_b}{\text{minimize}} & J_1(\boldsymbol{x}_a, \boldsymbol{x}_{ab}) & + & J_2(\boldsymbol{x}_b, \boldsymbol{x}_{ab}) \\ \text{subject to} & f_i(\boldsymbol{x}_a) \leqslant 0,\ i \in \mathcal{M}_a & & f_i(\boldsymbol{x}_b) \leqslant 0,\ i \in \mathcal{M}_b \\ & g_i(\boldsymbol{x}_a) = 0,\ i \in \mathcal{P}_a & & g_i(\boldsymbol{x}_b) = 0,\ i \in \mathcal{P}_b \end{array} . \tag{2.22}$$

Similarly, we can rewrite (2.22) as

$$\underset{\boldsymbol{x}_a, \boldsymbol{x}_{ab}, \boldsymbol{x}_b}{\text{minimize}} \quad J_1(\boldsymbol{x}_a^\star, \boldsymbol{x}_{ab}) + J_2(\boldsymbol{x}_b^\star, \boldsymbol{x}_{ab}), \tag{2.23}$$

where $J_1(\boldsymbol{x}_a^\star, \boldsymbol{x}_{ab})$ and $J_2(\boldsymbol{x}_b^\star, \boldsymbol{x}_{ab})$ can be found by solving

$$J_1(\boldsymbol{x}_a^\star, \boldsymbol{x}_{ab}) = \begin{array}{lll} \underset{\boldsymbol{x}_a}{\text{minimize}} & J_1(\boldsymbol{x}_a, \boldsymbol{x}_{ab}) & \\ \text{subject to} & f_i(\boldsymbol{x}_a) \leqslant 0, & i \in \mathcal{M}_a \\ & g_i(\boldsymbol{x}_a) = 0, & i \in \mathcal{P}_a \end{array} \tag{2.24a}$$

$$J_2(\boldsymbol{x}_b^\star, \boldsymbol{x}_{ab}) = \begin{array}{lll} \underset{\boldsymbol{x}_b}{\text{minimize}} & J_2(\boldsymbol{x}_b, \boldsymbol{x}_{ab}) & \\ \text{subject to} & f_i(\boldsymbol{x}_b) \leqslant 0, & i \in \mathcal{M}_b \\ & g_j(\boldsymbol{x}_b) = 0, & j \in \mathcal{P}_b \end{array} \tag{2.24b}$$

Observe that the solutions of (2.24) depend on a given value of $\boldsymbol{x}_{ab}$, therefore a coordination between sub-problems is needed to find an optimal $\boldsymbol{x}_{ab}^\star$ to solve problem (2.23) and thus the original problem (2.22).

Complicating Constraints

For complicating constraints, as the name suggests, instead of variables, we have constraints that prevent complete separation into sub-problems.

The variables $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are disjoint, $J(\boldsymbol{x})$ can be rewritten as $J(\boldsymbol{x}) = J_1(\boldsymbol{x}_1) + J_2(\boldsymbol{x}_2)$ and besides the constraints in sets $\mathcal{M}_1$ $\mathcal{M}_2$ and $\mathcal{P}_1$ $\mathcal{P}_2$ we also have equality and inequality constraints which couple $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, denoted $h_i(\boldsymbol{x}_1, \boldsymbol{x}_2) = h_{i_1}(\boldsymbol{x}_1) + h_{i_2}(\boldsymbol{x}_2) \leqslant 0$, $i \in \mathcal{O}$ and $w_i(\boldsymbol{x}_1, \boldsymbol{x}_2) = w_{i_1}(\boldsymbol{x}_1) + w_{i_2}(\boldsymbol{x}_2) = 0$, $i \in \mathcal{N}$:

$$\begin{array}{lccc} \underset{\boldsymbol{x}_1,\boldsymbol{x}_2}{\text{minimize}} & J_1(\boldsymbol{x}_1) & + & J_2(\boldsymbol{x}_2) \\ \text{subject to} & f_i(\boldsymbol{x}_1) \leqslant 0,\ i \in \mathcal{M}_1 & & f_i(\boldsymbol{x}_2) \leqslant 0,\ i \in \mathcal{M}_2 \\ & g_i(\boldsymbol{x}_1) = 0,\ i \in \mathcal{P}_1 & & g_i(\boldsymbol{x}_2) = 0,\ i \in \mathcal{P}_2 \\ & h_{i_1}(\boldsymbol{x}_1) + h_{i_2}(\boldsymbol{x}_2) \leqslant 0,\ i \in \mathcal{O} & & \\ & w_{i_1}(\boldsymbol{x}_1) + w_{i_2}(\boldsymbol{x}_2) = 0,\ i \in \mathcal{N} & & \end{array} \tag{2.25}$$

Adding auxiliary variables $\boldsymbol{\theta}_o$ and $\boldsymbol{\theta}_n$ we can rewrite the problem as

$$\underset{\boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{\theta_o},\boldsymbol{\theta}_n}{\text{minimize}} \quad J_1(\boldsymbol{x}_1^\star, \boldsymbol{\theta}_o, \boldsymbol{\theta}_n) + J_2(\boldsymbol{x}_2^\star, \boldsymbol{\theta}_o, \boldsymbol{\theta}_n), \tag{2.26}$$

where $J_1(\boldsymbol{x}_1^\star, \boldsymbol{\theta}_o, \boldsymbol{\theta}_n)$ and $J_2(\boldsymbol{x}_2^\star, \boldsymbol{\theta}_o, \boldsymbol{\theta}_n)$ can be found by solving

$$J_1(\boldsymbol{x}_1^\star, \boldsymbol{\theta}_o, \boldsymbol{\theta}_n) = \begin{array}{lll} \underset{\boldsymbol{x}_a}{\text{minimize}} & & J_1(\boldsymbol{x}_1) \\ \text{subject to} & f_i(\boldsymbol{x}_a) \leqslant 0,\ i \in \mathcal{M}_1 & h_{m_1}(\boldsymbol{x}_1) \leqslant \boldsymbol{\theta}_o,\ m \in \mathcal{O} \\ & g_j(\boldsymbol{x}_a) = 0,\ j \in \mathcal{P}_1 & w_{n_1}(\boldsymbol{x}_1) = \boldsymbol{\theta}_n,\ i \in \mathcal{N} \end{array} \tag{2.27a}$$

$$J_2(\boldsymbol{x}_2^\star, \boldsymbol{\theta}_o, \boldsymbol{\theta}_n) = \begin{array}{lll} \underset{\boldsymbol{x}_b}{\text{minimize}} & & J_2(\boldsymbol{x}_2) \\ \text{subject to} & f_i(\boldsymbol{x}_b) \leqslant 0,\ i \in \mathcal{M}_2 & h_{m_2}(\boldsymbol{x}_2) \leqslant -\boldsymbol{\theta}_o,\ m \in \mathcal{O} \\ & g_j(\boldsymbol{x}_b) = 0,\ j \in \mathcal{P}_2 & w_{n_2}(\boldsymbol{x}_2) = -\boldsymbol{\theta}_n,\ i \in \mathcal{N} \end{array} \tag{2.27b}$$

Again, observe that the solutions of (2.27) depend on the values of $\boldsymbol{\theta}_o$ and $\boldsymbol{\theta}_n$, thus a coordination between sub-problems is needed to find their optimal values in order to solve problem (2.26) and thus the original problem (2.25).

Remark 2.2

As seen in [Boy+15], a complicating constraint problem can be transformed into a complicating variable problem and vice versa by using auxiliary variables. For instance, we can substitute a complicating variable with two new local variables (one for each sub-problem) and add equality constraints to each sub-problem to force them to be equal. We will use henceforth only complicating constraints.

2.2.4 Generalizing

If we observe all the examples given, we can see a pattern emerge:

A decomposable problem

$$\begin{array}{lllll} \underset{\text{decision variables}}{\text{minimize}} & \text{objective}_1 & + & \text{objective}_2 & \\ \text{subject to} & \text{constraints}_1, & & \text{constraints}_2 & \\ & & \text{coupling constraints}_{12} & & \end{array} \tag{2.28}$$

is rewritten as an equivalent problem

$$\underset{\substack{\text{decision variables} \\ + \\ \text{auxiliary variables}}}{\text{minimize}} \quad \text{objective}_1^*(\text{auxiliary variables}) + \text{objective}_2^*(\text{auxiliary variables}) \tag{2.29}$$

where $\text{objective}_1^*(\text{auxiliary variables})$ and $\text{objective}_2^*(\text{auxiliary variables})$ are calculated by solving sub-problems

$$\begin{array}{ll} \underset{\text{decision variables}_1}{\text{minimize}} & \text{objective}_1 \\ \text{subject to} & \text{constraints}_1 \\ & \text{coupling constraints}_{12_1}(\text{auxiliary variables}_1) \end{array} \tag{2.30a}$$

$$\begin{array}{ll} \underset{\text{decision variables}_2}{\text{minimize}} & \text{objective}_2 \\ \text{subject to} & \text{constraints}_2 \\ & \text{coupling constraints}_{12_2}(\text{auxiliary variables}_2) \end{array} \tag{2.30b}$$

whose solutions depend on auxiliary variables and need some coordination to find optimal values for those auxiliary variables.

The equivalent problem, we call it *main problem*, while the sub-problems we call them *local problems*, which have some *local constraints*, and part of the *global (coupling) constraints* which depend on auxiliary variables that we call *interface variables*.

The coordination is usually an iterative method chosen to solve the *main problem*. At an iteration, the coordination uses some information from the solution of the sub-problems (given a value of *interface variables* as input) in order to update the *interface variables* to be used in the next step.

If we structure the coordination as an algorithm, we have algorithm 2.1.

Algorithm 2.1: General coordination for distributed optimization

Initialize interface variables for all sub-problems

repeat

  Solve sub-problems and calculate information about the solution

  Use information of solution to update interface variables

until *interface variables converge*

2.2.5 Decomposition techniques

If we analyze algorithm 2.1, to have a decomposition method, we need

— *Local problems*

— *interface variables*

— An update method to solve the *main problem*

The *local problems* are obtained through equivalent problems. For instance, we can use the original (primal) problem itself [Pau+16; CNN22] (examples shown in this section), the dual problem [Mor+11; BGB12; Vel+18], use some operator such as the proximal operator [Iid19; OV14], or other strategies.

The *interface variables* depend on the equivalent problem chosen. For instance, for the primal problem, usually, the dual variables are used [Coh78]; for the dual problem, the constraint residuals [Boy+15]; and for Alternating Direction Method of Multipliers (ADMM), primal variables and other multipliers (the alternating update of such multipliers gives its name) [Boy+11].

The update method is based on one optimization algorithm, such as bisection, cutting-plane, or methods that use the (sub)gradient or its approximates. The last class is the most used, and some examples can be the Arrow-Hurwicz-Uzawa iteration [BGB12], projected sub-gradient [Bie+12], and gradient ascent [Boy+11]. The exact form will depend on the topology used to solve the *main problem* (to be yet discussed).

This work will concentrate on a decomposition based on the **primal problem** and the **projected sub-gradient**. This decomposition will be explained in a future section (§5). For other examples of decomposition, the reader is referred to [MN+14] or [Con+06] and other articles cited throughout this work.

Now, after choosing a decomposition method, we need to know the topology of the problem. This topology can be chosen or imposed by some constraints (geographic or technological, for example). A brief discussion about different topologies follows.

Chapter 3

Topologies, Trust and Responsivity

Trust is an illusion,
M'Lady.
I believe only in
mutual self-interest.

Jupiter Ascending
The Wachowskis

When dividing the computation into different computing units, some questions are raised: How to distribute the computation units? Do they communicate? How do they influence each other? How do they communicate? In this chapter, we try to answer these questions.

Contents

3.1 How to distribute the computation units? 42
3.2 Representing communication 43
3.3 Do computing units communicate? 45
3.4 How do units influence each other? 45
3.5 How do units communicate? 47
3.6 Some drawbacks and compromises 48

3.1 How to distribute the computation units?

The distribution of the computation units depends on the system we want to control and the processing power/number of computation units needed to solve.

For example, suppose the system is large but geographically in the same place, depending on the number of computation units or the processing needed. In that case, the units can be in the same hardware (here, Graphical Processing Units (GPUs), multiple cores, threads, and other strategies are used). However, if the technical constraints (processing power) do not allow, multiple hardware must be used, and a physical communication channel is needed to link the processing units.

Assume the system is geographically diffused, and we want to use computing units to control groups of sub-systems. In such circumstances, it is straightforward to distribute the computing units in multiple hardware, placed strategically to correspond to their respective groups geographically.

For this work, we suppose that the large-scale system we want to control (2.6) can be decomposed into geographically diffused sub-systems. Each sub-system will have states $\boldsymbol{x}_i$ and inputs $\boldsymbol{u}_i$ that are a partition of the states and inputs of the original $\boldsymbol{x}$ and $\boldsymbol{u}$.

We also suppose the optimization problem we want to use to control the system (2.17) is decomposable. The computing units used to solve the sub-problems will not necessarily correspond to each sub-system (Fig. A.4a). For instance, a sub-problem may solve a problem with a subset of states of two different sub-systems to find the subset of inputs of a third and fourth sub-system.

(a) Non-corresponding.

(b) Corresponding.

Figure 3.1 – Decomposition of sub-problems (P_i) and sub-systems (S_i).

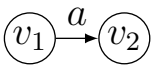
In the literature, it is not unusual to assume that we can choose the computing units in order to correspond to the sub-systems of the system as in Fig. A.4b [ACM21], i.e., a computing unit computes the solution of a sub-problem to find the input of the same corresponding sub-system. So, in **this work** we also assume the sub-problems and sub-systems are **corresponding**. Then, the terms sub-problems, sub-systems, agents and units will be used interchangeably.

3.2 Representing communication

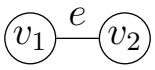
To represent and better visualize the agents and how they communicate, we will use graphs. In this section, we present a brief introduction to graphs. The fundamental unit in graph theory is a vertex/node (Fig. 3.2), which for us, will represent an agent.

Figure 3.2 – A vertex/node.

Arcs and edges represent the connection between nodes. While arcs are an ordered pair of vertices, such as $a = (v_1, v_2)$, indicating a direction, edges are an unordered pair, not implying direction, as $e = \{v_1, v_2\}$. We can see edges and arcs in Fig. 3.3.



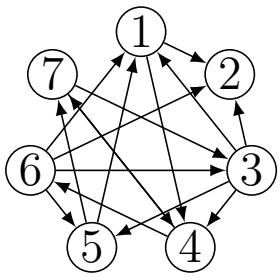
(a) An arc.



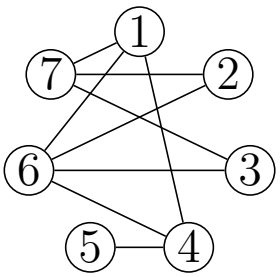
(b) An edge.

Figure 3.3 – Arcs and edges.

A directed graph $\mathscr{G}$ is composed of a set of vertices and a set of arcs $\mathcal{A} \subseteq \mathcal{V} \times \mathcal{V}$, written $\mathscr{G} = (\mathcal{V}, \mathcal{A})$. Similarly, an undirected graph $\mathscr{G}$ is a tuple composed of a set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, written $\mathscr{G} = (\mathcal{V}, \mathcal{E})$. An undirected graph can be interpreted as a directed graph where each edge $\{v_i, v_j\}$ can be split into two arcs (v_i, v_j) and (v_j, v_i). In Fig. 3.4, we can see examples of directed and undirected graphs.



(a) Directed graph.



(b) Undirected graph.

Figure 3.4 – Directed and undirected graphs with 7 vertices.

In an undirected graph, two vertices are called adjacent or neighbors if an edge links them. The neighborhood $\mathcal{N}$ of a vertex is the set of all the vertices to which it is linked. We can define a neighbor subgraph $\mathscr{N}(v)$ of $\mathscr{G}$ with the neighbors of v and the edges connecting them. For example, taking the graph in Fig. 3.4b, we can overlay the neighbor subgraph of vertex 6 in red (Fig. 3.5).

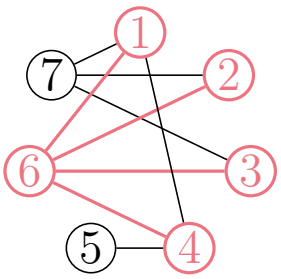


Figure 3.5 – Graph in Fig. 3.4b overlayed by neighbor subgraph of vertex 6 (in red).

For directed graphs, the notion of neighborhood can be split into in and out-neighborhoods of a vertex ($\mathcal{N}_{\text{in}}$ and $\mathcal{N}_{\text{out}}$), which are the sets of vertices whose arcs are inbound and outbound, respectively. A graph that all vertices are neighbors is called a *complete graph* (as in Fig. 3.6).

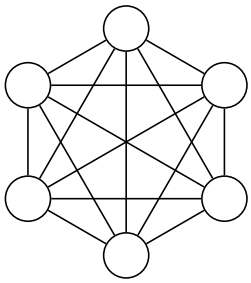


Figure 3.6 – A complete graph with 6 vertices.

A path of length r is a sequence of r edges/arcs connecting two vertices. In Fig. 3.7 we can see a path of length 2 between vertices 1 and 5.

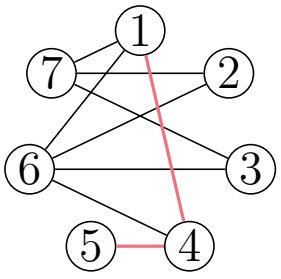


Figure 3.7 – A path between vertices 1 and 5 (in red).

A graph where for all pairs of vertices there is a path is called *connected*, otherwise *disconnected* or *disjoint* (Fig. 3.8).

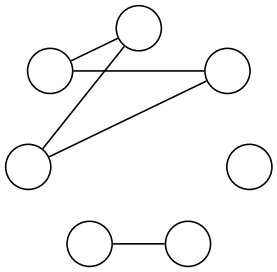


Figure 3.8 – A disjoint graph.

3.3 Do computing units communicate?

As shown in Section 2.2.1, the only case where communication is not needed to solve the optimization problem is in the uncoupled case (§2.2.2). However, there are some decomposition methods for the dMPC that even with coupled problems, they exploit the robustness properties of the MPC to compute the solution without communication [VSK14]. For the decomposition of some optimization problems, it is shown that in certain circumstances, the communication is not necessary [VE22].

Remark 3.1

Usually in dMPC literature the term "decentralized" refers to frameworks where the agents do not communicate [Chr+13, §4],[NM14]. However, the term can be confusing, and even in these works it is sometimes "misused". So, in this work, we opt for a different nomenclature, calling those kinds of frameworks "uncoordinated control" since there is no coordination between agents nor a coordinator agent to referee. We use the term decentralized as the opposite of centralized (monolithic), i.e., to describe structure instead of communication.

Most of the literature uses *coordinated control* [NM14; ACM21], in **this work**, analogously, we suppose the agents communicate, and the communication **graph** is **connected**.

3.4 How do units influence each other?

There exist multiple communication schemes. They depend on the decomposition, topology, and also trust/power. When looking at the trust/power axis, we can divide these schemes into two big groups (Fig. 3.9), one where groups of agents are more important than others (*hierarchy*) and another where all nodes are equally important (*anarchy*).

Sometimes the hierarchical structure may be implied by the decomposition.
For instance, some decompositions need the agents to solve their problems one after the other until the values converge (Fig. 3.10a) [LMC09]. These schemes are called *sequential*. We can see a hierarchy among the agents. Although convergence is reached, one group

(a) Hierarchy. (b) Anarchy

Figure 3.9 – Hierarchic and anarchic topologies.
Bigger nodes represent more power/influence over others.

of agents solves their problems disregarding the others, giving them an initial position of power, even if temporary.

In other decompositions, all agents solve their problems independently of others and then share results until convergence is reached, iteratively or not (Fig. 3.10b) [Liu+10]. These structures are called *parallel*. In such structures, there is no hierarchy.

(a) Sequential topology. (b) Parallel topology.

Figure 3.10 – Sequential and parallel topologies.

Moreover, in other decompositions, the hierarchical structure can be chosen. For example, when negotiation between agents is needed, questions about trust and security can shape the presence or absence of hierarchy. One of these questions is: Can an agent trust all other agents?

This question is common in multiple areas where communication, exchanges, or consensus between many agents are needed (politics, economy, and others). If an agent distrusts others, it can treat itself the security issues, or the agent can outsource the treatment to another agent. Additional agents can be included to serve as referees, coordinators, regulators, or certifiers. It is done because it is easier to trust in/verify a single agent or small group of agents than to trust in all agents. This way, the work is divided into more manageable parts.

One real-life example can be the Electronic Funds Transfer (EFT) between a seller and client [Sta88]. A seller, instead of trusting the credit (capacity of paying in future date) of each of its clients, she or he trusts in a small group of credit card brands to fulfill the EFT. Each credit card brand, in turn, trusts in a select group of credit-granting institutions with which potentially these clients are associated. This kind of arrangement creates a hierarchical tree-like structure called *polytree* in graph theory terms (Fig. 3.11). Seller and clients are leaves (terminal nodes), credit card brands are branch vertex (intermediary), and credit-granting institutions are roots (no parent nodes). While transactions today

happen in a distributed way, we still have these hierarchies. Some other political/societal parallel views are established in [McN+18] and [Ola+18].

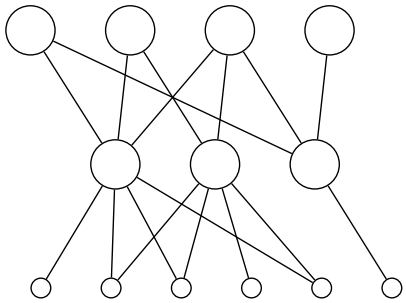


Figure 3.11 – A polytree.

In **this work**, we choose to use a **hierarchical** structure, described later (§5.1).

3.5 How do units communicate?

Depending on the problem and the structure, the connection of the nodes can be unidirectional or bidirectional. As said, these connections are represented by arcs and edges as shown in Fig. 3.3. While edges are usually simpler to deal with, arcs may need more convoluted ways to prove that the two nodes can somehow communicate (if there is a path between them) or that the algorithm converges [GS10].

Another classification of communication is the synchronization of the exchanges. Agents can exchange simultaneously (synchronously) or not. In general, synchronous communication is the most straightforward, while asynchronous depends on the communication scheme/policy. Broadcast and gossiping [GS10] are examples of these communication schemes.
In the broadcast scheme (a unidirectional scheme), an agent transmits its information to all its out-neighbors ($\mathcal{N}_{\text{out}}$), and each recipient agent updates its local variables.
Gossiping can be divided into two, asymmetric and symmetric. In the asymmetric (unidirectional), an agent chooses one of its out-neighbors, sends its information, and the recipient updates its local variables on the message's arrival. On the other hand, in a symmetric case (bidirectional), after updating its variables, the recipient becomes a transmitter, sending its information to the original sender, who updates its local variables.

In **this work**, we suppose nodes communicate **bidirectionally** and **synchronously**.

3.6 Some drawbacks and compromises

Failure/Robustness Although the hierarchical structure can help reduce the sources of distrust, it may also reduce robustness. Take, for example, the graph in Fig. 3.11. The hierarchical tree becomes disconnected if we remove the 3 intermediary nodes. There is a possibility that the complete system will collapse since the communications will not be accomplished. For decentralized systems with a central unit, this problem is commonly referenced as the *single point of failure* since removing the central unit collapses the communication and moves the system to a failure state. This problem is usually deviated by the use of redundancy. On the other hand, non-hierarchical structures with more complex topologies than tree-like ones can be more robust. In graph theory, there is what is called f-robustness [DI15], which defines how many connections we have to remove to make the graph disconnected. As we will see, this definition can be used in favor of some decomposition techniques to ensure robustness.

Propagation of information It is known that the propagation of information depends on the connectivity of the graphs, that means, to the neighborhood of each node. For instance, for a network with ring topology (Fig. 3.12a), the more agents there are, the more time the information takes to be diffused to all nodes. On the other hand, in a small-world topology (Fig. 3.12b), where there are some hubs (in red), propagation is increased.

(a) Ring graph. (b) Small-world graph.

Figure 3.12 – Ring and small-world graphs.

Energy consumption and Environment To overcome the problem of trust in anarchic (peer-to-peer) structures, a solution based on cryptography recently in vogue called *blockchain* was proposed [Nak08]. All historical data (log) is shared between agents in this solution. Before being sent to all other agents (or only neighbors), new data need to be processed to generate a *proof of work* (proof of a difficult computation which is easy to test) to create a block. This block needs to be validated (data and *proof of work* are valid) by at least more than half of the agents before being chained into the historical data, forming a chain of blocks, thus the name. In the case of multiple chains, the longest

chain is always chosen since it is where most *proofs of work* resides. This policy is called the *Longest Chain Rule*.

As each block has a different *proof of work*, to create a new block with unreasonable data, an ill-intentioned agent needs to alter all the past blocks in the chain, recreating all *proofs of work* one by one, until the false data could be accepted. This way, the attacker would need to have more than half of the total processing power of the network, creating the longest chain. This kind of attack is called a 51% attack or majority attack [Cil+20].

The *proof of work* strategy trades trust for energy. A recent work [CC18] estimates the energy consumption of one of the most diffused applications of the *blockchain*, a crypto-currency called Bitcoin. It estimates that each transaction consumes at least 200kWh, which is greater than the monthly electrical consumption of a small household of two people in Brazil [EPE22]. A more recent work [RD22] makes a Life Cycle Assessment of Bitcoin mining in a power plant and concludes that its annual emissions of metric tons of CO_2-eq is comparable to the annual emissions of 140,000 passenger vehicles.

All the points above must be considered before choosing a structure.

Chapter 4

Anomalous Behaviors: What they are? How to combat?

"What the hell is going on with our equipment?"
"It wasn't meant to do this in the first place."

Half-Life
Valve

The decompositions shown in the last chapters work in normal conditions, but we can analyze more interesting cases, such as when systems do not behave nominally.

This chapter briefly discusses the causes of anomalous behaviors, how they can happen, and the primary means used in the literature to mitigate their effects.

Contents

4.1 What are anomalous behaviors? 51
4.2 Security 52
4.3 Vulnerabilities in CPS 53
4.4 Attacks in CPS 54
4.5 Attacks in dMPC 58
4.6 Securing CPS 59
4.6.1 Passive strategies 60
4.6.2 Active strategies 61
4.6.3 Detection 61
4.6.4 Mitigation/Recovery 62

4.1 What are anomalous behaviors?

We define *anomalous behaviors* (or non-compliant behaviors) as any non-expected (change of) behavior of a system.

There are two primary causes for a system not to perform as nominally expected: *faults* and *attacks*. The main difference between these two is *intention*; while faults happen

unintentionally, uncoordinated and with no objective, attacks happen intentionally, usually coordinated having a malicious objective.

Remark 4.1

A non-expected behavior can also be observed when there are modeling errors, i.e., the theoretical model used to describe the system does not correspond to the actual system. In this case, the system did not change its behavior, but the expectation was false. So, for the time being, we will ignore modeling errors and assume they do not cause non-nominal behaviors.

Both faults and attacks can deviate the overall system from its nominal optimal behavior, but the most severe effect they can produce is a complete breakdown. In CPS, as the systems are connected to physical components, a malfunction can provoke a great impact on the population since some of the systems can control important aspects of modern life, such as water and energy supply.

That is why we are interested in assuring the normal operation of such systems.

4.2 Security

Security alludes to a system's safe and reliable operation even under unexpected circumstances, such as faults and attacks. To understand the security of CPS, we can first borrow some definitions from cyber-security and then extend them to CPS.

Computer systems are usually described in terms of data and resources, and *cyber-security* is ensured by three pillars: Confidentiality, Integrity and Availability (CIA) of such components [Bis05].

Confidentiality refers to how undisclosed the data and resources are. Sometimes parts of the system must be secret and only accessible to a selected group. Be it for privacy reasons, to avoid disclosing personal info, or even to prevent the exposition of possible system vulnerabilities (a pop culture example can be the Death Star Plans on the Star Wars Saga).

Integrity means how trustworthy the data and resources are. For example, new equipment is more trustworthy than old ones. For data traffic, the integrity of the communication may be divided into the integrity of the information sent/received (data integrity) and the integrity of identifying the source/recipient (called authentication).

Availability relates to how accessible to use the data and resources are when desired. For example, if one control is given to a system, it should be applied to it, and equally, if

we need data from a sensor, it must be able to send it. Furthermore, it is always good to call attention to, as exposed in [Bis05], that "an unavailable system is at least as bad as no system at all".

The vulnerabilities of a system can compromise any of the three pillars, sometimes multiple at the same time. So, let us explore the sources of the vulnerabilities in CPS.

4.3 Vulnerabilities in CPS

As it is known, CPS have components in both Cyber and Physical domains. So, as in computer systems, we can divide them into data (the immaterial part) and resources (the material part). We can also divide them into more specific parts such as: **Transducers**, which are sensors and actuators (as motors, valves, cameras, antennae, thermocouples, etc.); **Channels**, used for communication (wires, air, tubes, etc.); **Connectors**, which connect components by transmitting some physical quantity (wires, circuit traces, cables, water pipes, etc.); the **Software**, which is the logic used to operate the physical components (code, Programmable Logic Controller (PLC) logic, etc.); and the **Controller**, which is the hardware running the software (PLCs, micro-controllers, computers etc.).

Each one of the aforementioned components represents a source of vulnerability.
Transducers can be targets of sabotage. For instance, an attacker can use a cold object to disrupt thermostat readings to increase the temperature of a room. Nevertheless, the components may also deteriorate with time, which can eventually cause a fault. **Channels** and **connectors** can as well be targets of sabotage, as someone can interfere with the transmission to observe the traffic to gather information or intentionally interrupt it. Regardless, interruptions and corruptions might be caused by natural accidents such as solar flares (causing radio blackouts), trees knocking down electricity cables, or sharks eating underwater cables. **Software** can also be a source of vulnerability due to *bugs*, which can interrupt normal functioning or even let a *hacker* gain total control of the system. The **controller**, if not well dimensioned (computing capacity), can also be a vulnerability. If it receives more requisitions than expected, the system will be overloaded and will cease to respond.

It is worth emphasizing that in a Internet of Things (IOT) context, almost all sensors and actuators have embedded controllers and software, which can also be sources of vulnerabilities themselves.

Attackers can discover some of those vulnerabilities and use them in their favor. Due to this adversary behavior, in this work we will concentrate on attacks, eventually mentioning faults.

4.4 Attacks in CPS

From the definition of cyber-attacks [Bis05] and other definitions used in this chapter, we can define an **attack** as an ill-intentioned action that uses vulnerabilities of a cyber-physical system to violate its security. The perpetrator of the action is called **the attacker**.

In a computer context [Bis05], Bishop divides cyber-attacks into "four broad classes": disclosure, deception, disruption and usurpation. **Disclosure** is the unauthorized access to information (break of confidentiality). **Deception** tries to deceit, making false data pass as true (break of data integrity). **Disruption** interrupts or prevents the system from working correctly (break of availability). Furthermore, **usurpation** is the unauthorized control of the system (break of authenticity).

In a CPS/control context [CAS08], normally only the first 3 categories are used, calling them Deception, Disclosure, and Denial of Service (DoS) instead of disruption. Sometimes, for instance [ACS09], authors use only deception and DoS to divided attacks, probably because disclosure attacks do not affect performance. However, as we will see, depending on the attacker, disclosure attacks combined with other kinds of attacks can create some sneaky attacks.

This model of categorization is usually called the Disclosure, Deception and Denial of Service (DDD) model, and authors in CPS also add physical attacks [Dib+19]. To demonstrate the DDD model, the components of CPS are abstracted into 4 different parts: the physical system to be controlled, the controller, the communication channels, and the information trafficking by them (as seen in Fig. 4.1).

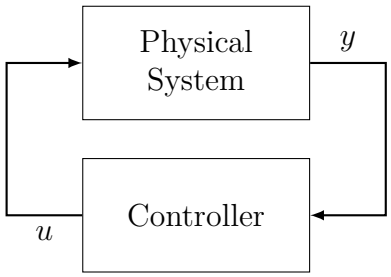


Figure 4.1 – General abstraction of CPS. (Adpated from [CAS08; ACS09])

In the diagram, u is the control input, which the controller calculates. The input is then applied in the physical system by the actuators. Moreover, y corresponds to the systems' outputs transmitted by the sensors.

Due to the networked nature of the systems, we modify the scheme also to model the communication between systems. In Fig. 4.2, we represent all other systems as a single entity called "network" with which the system may communicate. We also add two other signals, which represent the information shared with the network, I_{in} is information received from all other systems, and I_{out} is the information sent to all other systems.

While using this general networked control system scheme, the DDD model can be

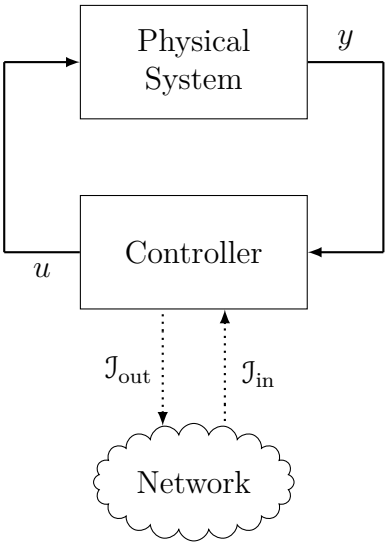


Figure 4.2 – General abstraction of networked CPS.

characterized by where and how the attack happens. In Fig. 4.3, we can see multiple kinds of attacks enumerated from A1 to A13. The attacks from A1 to A4 are disclosure attacks, which happen at the information level. The attacker observes the traffic, maybe to a posterior use. This attack is also known as a **snooping** or **eavesdropping** attack. A5 to A8 represent deception attacks. The attacks also happen at the information level, but the attacker changes the signals to a modified version (here represented by a ˜ above the variable name). The attacks from A10 to A12 are disruption (or **DoS**) attacks, where the attacker attacks the communication channel, momentarily or not preventing communication between two entities. Furthermore, A13 are physical attacks, where the physical system is attacked. In this attack, the attacker physically changes the behavior of the system.

As argued in [CAS08], to accomplish attacks A13, attackers need physical access to the plant. Alternatively, the attackers need to be close to the plant, or they use a teleoperated machine near the plant. The risks involved in trespassing and interacting with the machine or planting a machine can discourage the attacker since they can injure themselves or be identified. Attacks A1-A12, on the other hand, may present less risk since the attacker need only to have access to the communication channel.

For instance, nowadays, as communication may happen through the internet, attackers could snoop on some information from the comfort of their houses. They can use an open-source "packet sniffer" such as wireshark [1], which can read different protocols, including the Fieldbus family. Many of the distributed control systems use protocols from this family, for example, PROFIBUS and PROFINET (used for Supervisory Control and Data Acquisition (SCADA) architectures connecting PLCs, Human Machine Interfaces (HMIs) and Supervisors Interfaces), CAN (used for automobiles), and others.

1. https://www.wireshark.org

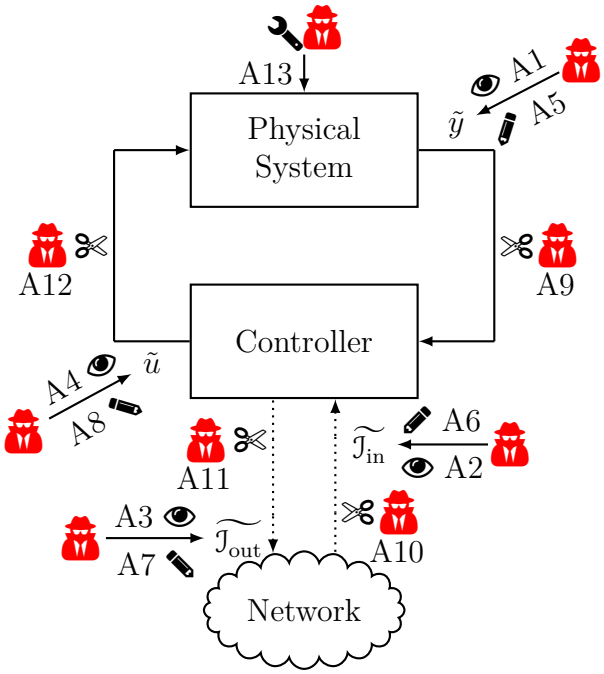


Figure 4.3 – Attacks in networked CPS. Adpated from [ACS09; Dib+19].

In [Tei+15], the authors stress that the attacks do not necessarily happen purely. For instance, an attacker could record the information trafficking in the communication channel (disclosure attack) to replay it in a future time, deceiving controllers and supervisors (deception attack) and generating some kind of input in the system which can be harmful for the physical system. This attack is commonly known as **Replay attack** [ZM14]. One of the classic examples of this attack in pop culture is heist movies, such as Ocean's Eleven, where the protagonists record the video feed of surveillance cameras to deceive security guards. This kind of attack exemplifies how deception attacks can be subtler and more ingenious than DoS attacks, as stressed in [MS09].

In [Tei+15], the authors also emphasize how the knowledge of control and physical aspects of CPS (sensors, actuators, and plant dynamics) can be advantageous for an attacker. So, instead of using the DDD model purely, Teixeira et al. [Tei+15] proposed an attack space with 3 orthogonal dimensions: Model Knowledge, Disruption resources, and Disclosure resources as seen in Fig. 4.4.

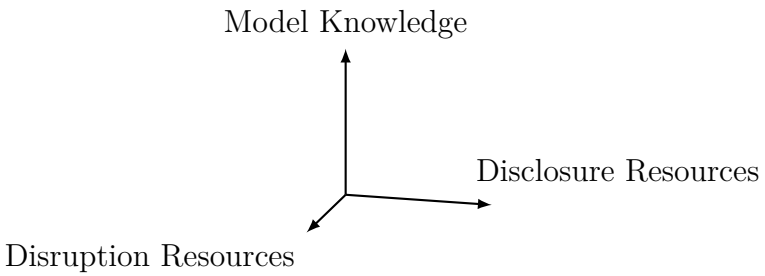


Figure 4.4 – 3 dimensional attack space.

To illustrate, we can populate the attack space with some attacks, some already cited and others which will be explained subsequently (Fig. 4.5).

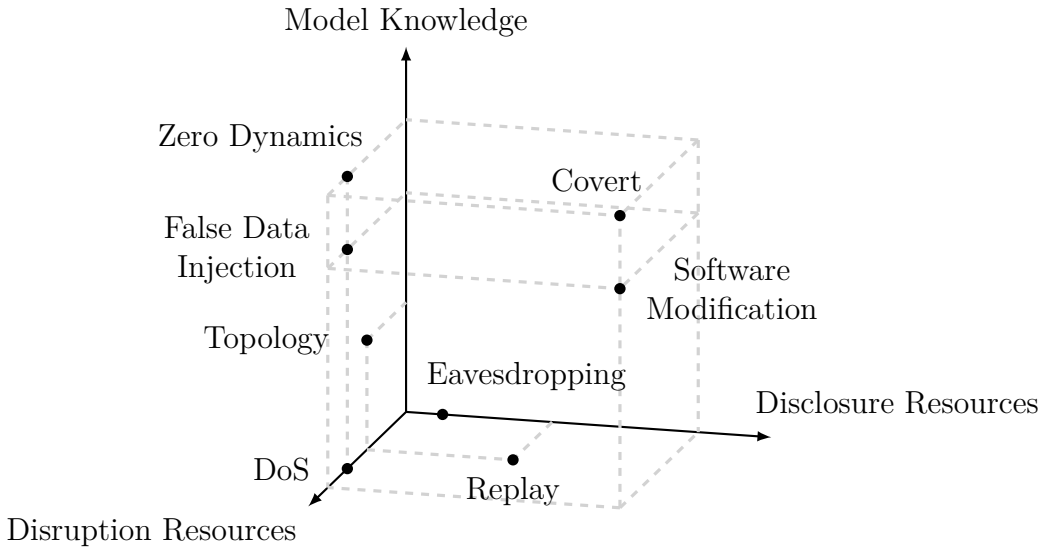


Figure 4.5 – Attack space and some attacks. (Adpated from [Tei+15])

In **DoS attacks**, the attacker interrupts the communication between entities directly by attacking the communication channel [SY19; Zha+20; Yan+19] or by making more requisitions to the controller than it can expect, this is also called **spamming** or **jamming**. Such attacks cause what we call **packet drops** [Che+21]. Depending on the scale of the system (computational power), attackers may coordinate multiple computational units (usually geographically distributed) to make the requests. This is called a Distributed Denial of Service (DDoS) [WL13; Hus+21].

Remark 4.2

Observe that packet drops may also occur in some unreliable networks, sometimes called **lossy networks**. The study of communication in such networks is important since wireless communication (profusely used nowadays) usually presents packet loss and acknowledgment to the reception of packets with protocols such Transmission Control Protocol (TCP) needs synchronization, which for some use cases may increase drastically delays [Bof+19].

In **Topology attacks**, the attackers manipulate the system's topology or mislead the control to believe the system is under a different topology (with false data). Such attacks are done by changing the state of devices like switches and breakers [KT13; WDL16; ZLW21].

In a **Software Modification attack**, the attacker modifies the software used in the controller, as done by the attackers in the Stuxnet attack [Lan11].

In **FDI attacks**, the attacker sends false data to the controller or the plant, usually

with some insight about the functioning of the system [PDB13]. In an example shown in [Tei+15], the attacker's objective is to inject a constant bias so it can affect the steady state value of the variables in the system.

In **Covert attacks**, the attacker fabricates new control and output signals to be fed to the plant and the controller. For this, the attacker uses a double of the plant (model) and all available signals. This way, the covert controller can choose a different reference, and since the dynamics are close enough to the plant, the attack may go unnoticed (stealth attack) [Smi15; HZ16; Bar+20] (See Fig. 4.6)

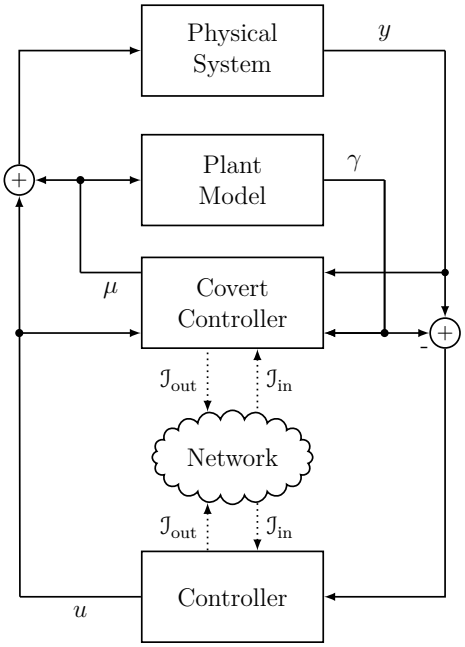


Figure 4.6 – Covert agent. Adpated from [Smi15; Bar+20].

As we will soon discuss, to counter these attacks, sometimes we use detectors of anomalous behaviors which try to detect a change in the system's dynamics (plant plus controller) [Tei+12; Tei+15; HZ16]. More generically, an attack that can go unnoticed under these detectors is called a **Zero Dynamics Attack**.

The cited attacks are not extensive but give a background of the more usual attacks presented in the literature. Other attacks can be seen in [Tei+15; ZLW21] and all other cited works in this chapter.

4.5 Attacks in dMPC

As pointed out in [ACM21], although cyber-physical security has been studied in MPC frameworks [SY19; FLT22], the dMPC community has not yet studied it sufficiently. While many decompositions methods are discussed, the agents usually are considered cooperative, and no vulnerabilities are explored. However, some effort has been made in recent years to explore the vulnerabilities of some decompositions methods, noticeably

dual decomposition [Vel+17b; Vel+17a; Vel+18; Ana+18; Ana+19; Ana+20] (which is the most widely used decomposition) and Jacobi-Gauß decomposition [CMI18].

In all these works, the attacks are not originated from an external agent. One (or a group) of the agents behave non-cooperatively and tries to steer de decomposition based on an objective, be it to destabilize the system or to benefit a group of agents. These kinds of attacks are called **inter-agent attacks** and are sometimes considered as **insider attacks** (as if someone with sufficient access rights has taken control). Usually, the objectives of the attacker are accomplished by deception attacks.

The mentioned works present FDI attacks, which they categorize based on where in the optimization problem the deception occurs.

Selfish attack is when the attacker modifies its local objective function (usually multiplying it by a positive constant), so it can be more penalized. This change causes all other agents to increase their efforts to balance this disadvantage and decrease the local cost of the attacker.

In **Fake weights attacks**, the non-cooperative agent uses different weights in its local objective function. A selfish attack can be viewed as a specific case of this kind of attack.

Supposing the agents follow a reference, in a **false reference attack** the malicious agent uses a different value of reference to calculate its local objective function.

For a **fake constraints attack**, the attacker uses different constraints when solving its local optimization problem.

The authors call a **"liar" agent attack** when, after the negotiation is finished, the attacker uses a control different from the agreed one.

In **this work**, we also choose to study **FDI attacks** but use another approach for security.

4.6 Securing CPS

The first thing to remember is that nothing is safe, and everything can be potentially attacked. The initial step is to assess the risks and see which parts of your system you trust. Some parts of the system can be more *vulnerable* to *threats*, i.e. some components have more probability of being attacked or suffering from faults. As a second measure, we usually protect these components with more probability of presenting an anomalous behavior [Bis05] to prevent such behaviors. In [WL13], some risk assessment methods for smart grids are presented.

Prevention To prevent anomalous behaviors, we secure the vulnerable components. Some examples follow. Tampering of *physical components* is prevented by *enclosing* them by walls and doors with *access control* and adding *monitoring devices* (cameras and alarms) [CAS08; Din+18], persuading the attacker not to approach such components.

Faults due to *deterioration* are usually prevented by a *periodical preventive maintenance* [Che+21], and substitution of deteriorated components by new ones. Attacks and faults caused by *software vulnerabilities* may be prevented by *corrective patches* that are sent to all users of the software to correct the bugs. Another method is *software rejuvenation* [Alo+12; Gri+20], where the system is refreshed with a trusted copy of the original software periodically, useful when there is a chance the software was corrupted. For communication/transmission, we increase the robustness of the mean by using better cables with insulation and braided shields, for example. To secure exchanges, usually, *cryptography* is used to ensure data integrity and authenticity of agents [Din+18]. Another option to secure data is the *blockchain* (mentioned in 3.6), which the proper conception can protect against deception attacks or data corruption. Another method to prevent deception attacks is to use a game-theoretic approach [MMC11; ZM11] which may also disincentivize the attacker from being greedy since it knows that potentially all other agents could also be greedy. This approach can result in suboptimality.

If prevention fails, it needs to have another strategy to maintain security. In the literature, those strategies are usually accompanied by one of two words: robustness and resilience. Although both terms are generally used interchangeably in the literature, here we make a distinction. **Robustness** can be defined as the ability to withstand perturbations without change in function [Jen03]. **Resilience** is the ability to maintain an acceptable level of operational normalcy in response to disturbances, including threats of an unexpected and malicious nature [Rie10].

Here we interpret robust methods as passive strategies, while resilient methods are active and adaptive to reestablish a more optimal behavior. So we can divide the security methods into two categories: active and passive.

4.6.1 Passive strategies

Passive methods are usually based on robust control, where the control law implemented does not change in the presence of faults or attacks. It can maintain satisfactory performance (with loss of optimality) if values of some project variables are inside a given safety range or if a certain maximum number of components present anomalous behaviors.

The majority of the robust strategies is used for Fault Tolerant Control (FTC). However, depending on the similarities between faults and the attacks, some robust control approaches can be used, as indicated in [Tei+15; Din+18; ACM21].

For example, a FDI attack could change the value of a variable just as a fault or malfunction could add a disturbance on the signal, so some passive FTC techniques such as the one shown in [VSK14] could also be used provided that the data injected respects the bounds set by the method,

Another example is in distribution grids where there is a N-1 static security index, which indicates how stable the grid is even if 1 component presents a failure (or is disconnected) [Qia+22]. So, for a system said N-1 safe, if a component is attacked (a DoS for example) the system still can present satisfactory performance. Velarde et al. in [Vel+17b; Vel+18] used a similar solution based on f-robustness of graphs [2] [DI15; WI19] for robust dMPC based on dual decomposition. During each negotiation step, extreme values (the f biggest and smallest) are ignored.

Yet another strategy is the one in [Vel+17a; Mae+21], where trustworthy historical scenarios (MPC trajectories) with known occurrence probabilities are stored and used to solve the problem together with the current calculation, which may not be so reliable due to attacks.

4.6.2 Active strategies

For other specific kinds of anomalous behaviors, e.g., more ingenious attacks such as replay attacks or covert attacks, some resilient methods need to be created. The active methods are usually bi-modal. When there is no attack, it behave in some way, and when an attack happens, it changes its behavior.

In order to identify an anomalous behavior, first, we need some prior information about the normal behavior of the system, for example, the dynamics of certain variables, bounds, or any other useful information. With this knowledge, we can supervise the threatened components and create monitors. This part is called the **Detection** phase, where the attack can be detected through the monitors/detectors. Sometimes there is an isolation step in which the misbehaving component/agent is identified. In some specific cases, the isolation can occur during the detection phase.

Once an attack is detected, the control law changes to mitigate the anomalous behavior's effects. This stage is called the **Mitigation** or **Recovery** phase.

4.6.3 Detection

The detection phase techniques can be categorized in some ways. For example, depending on if an additional signal to the monitored data is added (**active detection**) or not (**passive detection**).

The techniques can also be divided by whether they use event-triggered approaches. We give the works in [SY19; Hu+21; Sun+22] as examples of event-triggered methods, but this work does not focus on this kind of method.

Another categorization is if the monitoring devices use analytical knowledge of the system or use learning methods.

2. Graph maintain connectivity even if a number f of agents is disconnected

For active detection, we can mention the *watermarking* where a (pseudo) random authentication signal is superimposed to the monitored signal we want to secure [MS09; MWS15; SK17; KV17; LGG21], it can protect against replay attacks since with the knowledge of the random seed it is possible to have a unique timestamp.

Some passive and analytical examples can be the observators used in [HZ16] or residues of state dynamics [Tei+15; Boe+20], which can be associated with hypothesis testing [MS09] (χ^2 detectors), or other set membership detection as in [For+16] or [MTI18] for instance. Some more examples are seen in [PBB12; PDB13; Zha+21a; ACM21].

For learning methods we can reference [Ana+18; Ana+19; Ana+20] which uses a bayesian approach to learning bounds and detecting suspect agents. Another example is training neural networks/deep learning to create detectors. In [Hus+21], the authors train a convolutional neural network to detect 91% of DDoS attacks of a certain kind.

In [BAL20b; BAL20a], the authors use an optimization problem in order to identify which part of the system has been attacked.

Another method could be using coalitional control [CMC21], which can potentially cluster healthy agents and detect changes in topologies.

Remark 4.3

As stated in [ACM21] is important to observe that when choosing a bound to detect anomalous behaviors, there are false positives associated.

4.6.4 Mitigation/Recovery

The recovery phase consists of damage control, changing the system accordingly to reestablish normal behavior.

The main recovery option is the immediate substitution of the malfunctioning (or attacked) components by their redundancy (if there is) to prevent further consequences. Then we fix the component for later reuse (if possible).

If an immediate substitution is not possible, the control law is changed. The component is disconnected, and we use some robust strategies to mitigate anomalous behavior's effects. These strategies make the system behave not precisely as initially intended but in a satisfactory suboptimal way until we can restore normalcy. Examples can be [MTI18] in which tube-based MPC to robustify against disturbances inside a safety set. And similarly to [Vel+18], in [Ana+18; Ana+19; Ana+20] the authors base their solution on f-local graph robustness, where suspect agents are ignored during the dMPC negotiation.

Another option is to use adaptive strategies, which try to estimate variables or sets and compensate for the disturbances caused by the anomalous behaviors [YZG22; LCK20].

This last strategy is used in **this work**, which will be detailed in the following chapters.

Part II

Towards a safe dMPC algorithm

Choices & Stance

> There are few people, however, who, if you told them a result, would be able to evolve from their own inner consciousness what the steps were which led up to that result.
>
> *A Study in Scarlet*
> Sir Arthur Conan Doyle

As we have seen in the last chapters, there are many steps a designer has to take to design a CPS and decompose the MPC. Moreover, if we are interested in security, some more thinking must be put into it, and choices to be made.

In this brief chapter, we retrace the steps, recapitulating the choices made for this work mentioned in the course of the chapters. The idea of this chapter is to concentrate them all in one place so a reader can have a better global vision and know more or less what to expect (or not) from the next chapters.

We divide it into two sections. The first is dedicated to the choices regarding the design of the system's control. The second one is for the choices for security. In the latter, we compare this thesis with the most adjacent works, already presented in Chapter 4.

Control design

Control We model the system as a **Linear Time-Invariant Discrete Time (LTIDT) system** with **linear input constraints**, and we choose to control it using **MPC**. The objective function used is **quadratic**, which by consequence produces a MPC that solves a **QP problem**. However, we suppose the problem may impact the calculation if we try to solve it in a monolithic form. We need to **decompose** it.

Decomposition We choose to use the **optimization decomposition framework**, shown in [Con+06; Boy+15]. We suppose the system is **decomposable** into **sub-systems** and our original problem is decomposable into **sub-problems**.

Using the framework, we see the input constraints **couple** the sub-systems, and as a consequence, the sub-problems. The system is decomposed using the configuration for **complicating constraints**. Finally, we choose to use a decomposition called **primal decomposition**, detailed in the next chapter, which needs **coordination** to find the optimal solution.

Topology The first choice for the system's topology was to assume **correspondence** between sub-systems and sub-problems.

The agents were supposed to form a **connected** graph, in which the agents can communicate to coordinate the decomposition (**coordinated control**). For some security/trust reasons we choose to use an **hierarchical** topology, where we *entrust* an agent the role of **referee**, coordinating the exchanges between other agents.

We suppose the communication is **bidirectional**, where the agents can communicate with the **referee** and vice-versa. We also suppose the agents communicate **synchronously**, so we do not have delays in the communication.

Security

For security, we use [Vel+17b; CMI18] as inspiration to demonstrate the **vulnerabilities of the primal decomposition** method, which were never presented.

As the most adjacent works, we are only interested in **inter-agent attacks** of the **FDI** type, whose exact model is presented in the next chapter.

In [Vel+17b; Vel+18], the authors present robust methods based on graph theory, where the attacker is not detected per se, but suspect agents are disconnected or ignored somehow. In [Vel+17a; Mae+21], another robust method is used, this time employing a scenario-based approach, where suspect sequences receive different weights.

We opt, however, as in [Ana+18; Ana+19; Ana+20], for an **active resilient** approach. The strategy is precisely detailed in the next chapters, but we use an **active** method for the detection, where we add some data to detect the attacker. We use a **hybrid** of **analytical knowledge** of the decomposition and **learning methods** to create the detector. For the recovery, we use the same **analytical knowledge** used for the detection to reconstruct some variables and compensate for disturbances caused by the attack.

Due to the reduced number of adjacent works, we can compile and compare them in a tabular manner. In Tab. 4.1, we compare them with the methods presented in this thesis.

	Decomposition	Present vulnerabilities?	Resilient/Robust	Detection	Mitigation
[Vel+17a] [Mae+21]	Dual	Yes	Robust (Scenario)	NA	NA
[Vel+17b] [Vel+18]	Dual	Yes	Robust (f-robust)	NA	NA
[CMI18]	Jacobi-Gauß	Yes	–	–	–
[Ana+18] [Ana+19] [Ana+20]	Dual	Yes	Resilient	Analyt./Learn.	Disconnect (Robustness)
Our	Primal	Yes	Resilient	Active Analyt./Learn.	Data reconstruction

Table 4.1 – Comparison between adjacent works. NA meaning not applicable.

Chapter 5

Vulnerabilities in Primal Decomposition-based distributed Model Predictive Control

All the world
is birthday cake,
so take a piece,
but not too much.

It's All Too Much
The Beatles

In this chapter, we describe in more details the decomposition framework studied throughout this work. We apply the decomposition to the MPC problem we are interested and we give observations on the possible interpretations of the steps needed to solve the problem.

Then we present our first contribution, which is the study of the vulnerabilities of this framework and the impact on the MPC control.

We illustrate the effects with a simple academic example, so we can easily grasp and observe the possible consequences.

Contents

5.1 Primal decomposition-based dMPC 70
5.2 Example and Interpretations 73
5.3 Anomalous behaviors and their consequences 76
5.3.1 Attack of interest 77
5.4 Conclusion 79

5.1 Primal decomposition-based dMPC

We start from the monolithic MPC equivalent problem presented in (2.17), which is reproduced here for the reader's convenience:

$$\begin{array}{ll} \underset{\boldsymbol{U}[k]}{\text{minimize}} & \frac{1}{2}\left\|\boldsymbol{U}[k]\right\|_H^2 + \boldsymbol{f}[k]^T\boldsymbol{U}[k] \\ \text{subject to} & \bar{\Gamma}\boldsymbol{U}[k] \preceq \boldsymbol{U}_{\max} \end{array} . \tag{2.17}$$

As stated in §2.1.2, this problem is an input-constrained QP problem, which is convex.

To decompose the problem with the optimization decomposition frameworks shown in §2.2, we assume the objective function of problem (2.17) to be decomposable into M equally sized parts, which correspond (corresponding sub-systems and sub-problems §3.1) to subdivisions of the initial system (2.6). The subsystems are ruled by dynamics (A_i, B_i, C_i), and have states $\boldsymbol{x}_i[k]$, references $\boldsymbol{w}_i[k]$ and inputs $\boldsymbol{u}_i[k]$.

Each sub-problem is given an index in set $\mathcal{M} = \{1 : M\}$ and we rewrite (2.17) as

$$\begin{array}{ll} \underset{\boldsymbol{U}_1[k],\ldots,\boldsymbol{U}_M[k]}{\text{minimize}} & \sum\limits_{i\in\mathcal{M}} \left[\frac{1}{2}\left\|\boldsymbol{U}_i[k]\right\|_{H_i}^2 + \boldsymbol{f}_i[k]^T\boldsymbol{U}_i[k]\right] \\ \text{subject to} & \sum\limits_{i\in\mathcal{M}} \left[\bar{\Gamma}_i\boldsymbol{U}_i[k]\right] \preceq \boldsymbol{U}_{\max} \end{array} , \tag{5.1}$$

with local versions $\boldsymbol{U}_i[k] = [\boldsymbol{u}_i[0|k]; \ldots; \boldsymbol{u}_i[N-1|k]]$, $\bar{\Gamma}_i$, H_i and $\boldsymbol{f}_i[k]$ of variables presented in §2.2.

Remark 5.1

To pass from the global version of the variables in (2.17) to the local version in 5.1, we apply a permutation transformation $Z^TZ = I$ to $\boldsymbol{U}[k]$ to reorder its components, such

$$\bar{\boldsymbol{U}}[k] = Z\boldsymbol{U}[k] = [\boldsymbol{U}_1[k]; \ldots; \boldsymbol{U}_M[k]] . \tag{5.2}$$

Then we reorder other variables' elements consistently using the transformation.

$$\bar{\bar{\Gamma}} = \bar{\Gamma}Z^T = \left[\bar{\Gamma}_1, \ldots, \bar{\Gamma}_M\right] \tag{5.3}$$

$$\bar{\boldsymbol{f}}[k] = Z\boldsymbol{f}[k] = [\boldsymbol{f}_1[k]; \ldots; \boldsymbol{f}_M[k]] \tag{5.4}$$

$$\bar{H} = ZHZ^T = \bigotimes_{i\in\mathcal{M}} H_i \tag{5.5}$$

Using the methodology in §2.2.4 for coupling constraints, the primal decomposition divides the problem (5.1) into a main problem (5.6b) and local problems (5.6a) based on

the original primal problem (5.1) and using interface variables $\boldsymbol{\theta}_i[k], \forall i \in \mathcal{M}$:

$$\bar{J}_i^\star(\boldsymbol{\theta}_i[k])[k] = \begin{array}{ll} \underset{\boldsymbol{U}_i[k]}{\text{minimize}} & \frac{1}{2}\left\|\boldsymbol{U}_i[k]\right\|_{H_i}^2 + \boldsymbol{f}_i[k]^T \boldsymbol{U}_i[k] \\ \text{subject to} & \bar{\Gamma}_i \boldsymbol{U}_i[k] \preceq \boldsymbol{\theta}_i[k] : \boldsymbol{\lambda}_i[k] \end{array} \tag{5.6a}$$

$$\bar{J}^\star[k] = \begin{array}{ll} \underset{\boldsymbol{\theta}_i[k],\ldots,\boldsymbol{\theta}_M[k]}{\text{minimize}} & \sum_{i\in\mathcal{M}} J_i^\star(\boldsymbol{\theta}_i[k]) \\ \text{subject to} & \sum_{i\in\mathcal{M}} \boldsymbol{\theta}_i[k] \preceq \boldsymbol{U}_{\max} \end{array}, \tag{5.6b}$$

where $\boldsymbol{\lambda}_i[k], \forall i \in \mathcal{M}$ are the dual variables of the problems, i.e. the Lagrange multipliers associated with the constraints [BV04].

Problem (5.6b) is solved by updating the θ_i until convergence. The method chosen is the projected sub-gradient, which update is given by

$$\boldsymbol{\theta}[k]^{(p+1)} = \text{Proj}^{\mathcal{S}}(\boldsymbol{\theta}[k]^{(p)} - \rho^{(p)} \boldsymbol{g}[k]^{(p)}), \tag{5.7}$$

where p is a given step in the iterative process, $\boldsymbol{\theta}[k] = [\boldsymbol{\theta}_1[k]; \ldots; \boldsymbol{\theta}_M[k]]$, $\rho^{(p)}$ is a given step length, $\mathcal{S} = \{\boldsymbol{\theta}[k] \mid I_c^M \boldsymbol{\theta}[k] \preceq \boldsymbol{U}_{\max}\}$, $c = \#\,\boldsymbol{\lambda}_i[k] = \#\,\boldsymbol{\theta}_i[k] = \#\,\boldsymbol{U}_{\max} = n_c N$, $I_c^M = \mathbf{1}_M^{\mathrm{T}} \otimes I_c$, and $\boldsymbol{g}^{(p)}[k]$ is a sub-gradient of the objective function of problem in (5.6b) in step p. The choice of step length is given such

$$\lim_{p\to\infty} \rho^{(p)} \to 0$$

and

$$\sum_{p=1}^{\infty} \rho^{(p)} \to \infty.$$

A usual law is

$$\rho^{(p)} = \frac{1}{a + bp}, \tag{5.8}$$

with a and b arbitrarily chosen [Con+06].

From [BV04] and [Boy+15], we can derive that the opposite of the dual variables $\boldsymbol{\lambda}_i[k]$ of the local problems are sub-gradients of the local problems, and since the objective function of the main problem is a sum of the local objectives, then the $\boldsymbol{\lambda}_i$ can form a sub-gradient of the main problem. If we stack the $\boldsymbol{\lambda}_i[k]$ in $\boldsymbol{\lambda}[k] = [\boldsymbol{\lambda}_1[k]; \ldots; \boldsymbol{\lambda}_M[k]]$, it can be used in (5.7), yielding

$$\boldsymbol{\theta}[k]^{(p+1)} = \text{Proj}^{\mathcal{S}}(\boldsymbol{\theta}[k]^{(p)} + \rho^{(p)} \boldsymbol{\lambda}[k]^{(p)}). \tag{5.9}$$

Using (5.6a) and (5.9), we can alter Algorithm 2.1 to correspond to a decomposition of the original MPC problem (5.1) using the primal decomposition (Algorithm 5.1).

Since each problem (5.6a) can potentially be solved in parallel, a different computation

Algorithm 5.1: Decomposition of MPC problem using primal decomposition.

Input: Maximum number of iteration $p_{\max}$, Bound ϵ,Constants a and b
Output: Optimal allocations $\boldsymbol{\theta}[k]$
$p \leftarrow 0$
Initialize $\boldsymbol{\theta}_i[k]^{(p)}$, $\forall i \in \mathcal{M}$, such that $\boldsymbol{\theta}[k]^{(p)} \in \mathcal{S}$
repeat
  Solve local sub-problems (5.6a) and calculate $\boldsymbol{\lambda}_i[k]^{(p)}$ (potentially in parallel)
  Update $\boldsymbol{\theta}[k]$: $\boldsymbol{\theta}[k]^{(p+1)} \leftarrow \text{Proj}^{\mathcal{S}}(\boldsymbol{\theta}[k]^{(p)} + \rho^{(p)}\boldsymbol{\lambda}[k]^{(p)})$
  $p \leftarrow p + 1$
until $\left[\left\|\boldsymbol{\theta}[k]^{(p)} - \boldsymbol{\theta}[k]^{(p-1)}\right\| \leqslant \epsilon\right] \vee [p \geqslant p_{\max}]$

unit is allocated to solve each problem. For the update (5.9) the agents need to have the $\boldsymbol{\lambda}_i[k]$ of all other agents to update the $\boldsymbol{\theta}_i[k]$. We could create an anarchic solution analog to the one in [Vel+18] where the $\boldsymbol{\lambda}_i[k]$ information circulates among agents, and each agent updates its own $\boldsymbol{\theta}_i[k]$. However, we decide on a more private structure. We choose to use an **hierarchic** solution, allocating another agent to aggregate the $\boldsymbol{\lambda}_i[k]$ and coordinate the updates of the $\boldsymbol{\theta}_i[k]$, we called this agent the *coordinator*. This way, each agent has only access to its own $\boldsymbol{\lambda}_i[k]$ and $\boldsymbol{\theta}_i[k]$.

We can illustrate the communication structure using Fig. 5.1, which is the same in our introductory example. One can observe that the structure is the same as the hierarchical structures shown in §3, where the agents trust the coordinator to referee their interactions. We summarize the complete primal decomposition-based dMPC algorithm in Algorithm 5.2.

Algorithm 5.2: Primal decomposition-based dMPC.

Coordinator initialization:
  Initialize k, p, a, b, $p_{\max}$ and ϵ
while $k \geqslant 0$ **do**
  Initialize $\boldsymbol{\theta}_i[k]^{(p)}$, $\forall i \in \mathcal{M}$, such that $\boldsymbol{\theta}[k]^{(p)} \in \mathcal{S}$
  Coordinator sends $\boldsymbol{\theta}_i[k]^{(p)}$ for all agents
  Exchange between Coordinator and agents:
    repeat
      Agents solve local sub-problems (5.6a) and send $\boldsymbol{\lambda}_i[k]^{(p)}$ to coordinator
      Coordinator updates $\boldsymbol{\theta}[k]$ using (5.9) and sends to local agents
      $p \leftarrow p + 1$
    until $\left[\left\|\boldsymbol{\theta}[k]^{(p)} - \boldsymbol{\theta}[k]^{(p-1)}\right\| \leqslant \epsilon\right] \vee [p \geqslant p_{\max}]$
  Agents apply on respective sub-systems the last calculated $\boldsymbol{u}_i^{\star}[0|k]$
  $k \leftarrow k + 1$

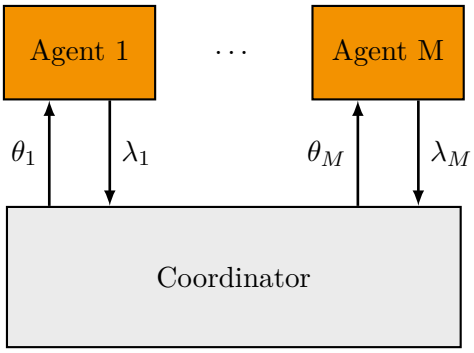


Figure 5.1 – Exchange between local controllers and coordinator in Primal decomposition-based dMPC.

Remark 5.2

The same decomposition can be used if the coupling constraints are equalities or if there are local constraints for the subsystems. We can change problems (5.6a) and (5.9) accordingly to respect such modifications, i.e., adding the local constraints to the local problems and changing the inequality constraints to equality ones, including in the set $\mathcal{S}$. Examples of each are given in the next section and chapters.

5.2 Example and Interpretations

As a simple numerical example to illustrate the functioning of the algorithm, we take $M = 3$ 1-dimensional Single Input Single Output (SISO) ($n_x = 1$,$n_c = 1$) systems coupled by the inputs. The systems have no physical meaning, but their small scale makes it easier to visualize.

The systems are described by the following LTIDT dynamics

$$x_i[k+1] = a_i x_i[k] + b_i u_i[k] \tag{5.10}$$

where $a_1 = 0.8$, $a_2 = 0.6$, $a_3 = 0.4$, $b_1 = 0.3$, $b_2 = 0.5$, and $b_3 = 0.6$. The system is constraint by an equality constraint

$$u_1 + u_2 + u_3 = \boldsymbol{u}_{\max} \tag{5.11}$$

with $\boldsymbol{u}_{\max} = 4$.

We use a dMPC with horizon $N = 2$, initial conditions $x_1[0] = 3$, $x_2[0] = 2$, and $x_3[0] = 1$ and state references $w_1[0] = 1.07x_1[0]$, $w_2[0] = 1.10x_2[0]$, and $w_3[0] = 1.05x_3[0]$

and gains $Q_i = 10I_2$ and $R_i = I_2$ that results in the problems

$$\bar{J}_i^\star(\boldsymbol{\theta}_i[k])[k] = \begin{array}{ll} \underset{\boldsymbol{U}_i[k]}{\text{minimize}} & \frac{1}{2}\left\|\boldsymbol{U}_i[k]\right\|_{H_i}^2 + \boldsymbol{f}_i[k]^T\boldsymbol{U}_i[k] \\ \text{subject to} & \bar{\Gamma}_i\boldsymbol{U}_i[k] = \boldsymbol{\theta}_i[k] : \boldsymbol{\lambda}_i[k] \end{array} \tag{5.12}$$

with

$$\begin{aligned} H_1 &= \begin{pmatrix} \frac{6077}{3125} & \frac{288}{625} \\ \frac{288}{625} & \frac{197}{125} \end{pmatrix}, \quad \boldsymbol{f}_1[k] = \begin{pmatrix} \frac{1968}{625} \\ \frac{192}{125} \end{pmatrix} x_i[k] - \begin{pmatrix} \frac{12}{5} & \frac{48}{25} \\ 0 & \frac{12}{5} \end{pmatrix} \boldsymbol{W}_i[k], \quad \bar{\Gamma}_1 = I_2, \\ H_2 &= \begin{pmatrix} \frac{278}{125} & \frac{27}{50} \\ \frac{27}{50} & \frac{19}{10} \end{pmatrix}, \quad \boldsymbol{f}_2[k] = \begin{pmatrix} \frac{306}{125} \\ \frac{27}{25} \end{pmatrix} x_i[k] - \begin{pmatrix} 3 & \frac{9}{5} \\ 0 & 3 \end{pmatrix} \boldsymbol{W}_i[k], \quad \bar{\Gamma}_2 = I_2, \\ H_3 &= \begin{pmatrix} \frac{5213}{3125} & \frac{144}{625} \\ \frac{144}{625} & \frac{197}{125} \end{pmatrix}, \quad \boldsymbol{f}_3[k] = \begin{pmatrix} \frac{696}{625} \\ \frac{48}{125} \end{pmatrix} x_i[k] - \begin{pmatrix} \frac{12}{5} & \frac{24}{25} \\ 0 & \frac{12}{5} \end{pmatrix} \boldsymbol{W}_i[k], \quad \bar{\Gamma}_3 = I_2. \end{aligned} \tag{5.13}$$

The coordinator update function is the same as (5.9) but with $\mathcal{S} = \{\boldsymbol{\theta}[k] \mid I_c^M\boldsymbol{\theta}[k] = \boldsymbol{U}_{\max}\}$, where $c = 2$.

From now on, we remove the $[k]$ indices for convenience since all calculation is for the same step k unless expressly stated otherwise.

This Euclidean projection can be calculated explicitly, yielding

$$\boldsymbol{\theta}^{(p+1)} = \boldsymbol{\theta}^{(p)} + \rho^{(p)}\boldsymbol{\lambda}^{(p)} + I_c^{M^{\mathrm{T}}}/(I_c^M I_c^{M^{\mathrm{T}}})\left(I_c^M(\boldsymbol{\theta}^{(p)} - \rho^{(p)}\boldsymbol{\lambda}^{(p)}) - \boldsymbol{U}_{\max}\right), \tag{5.14}$$

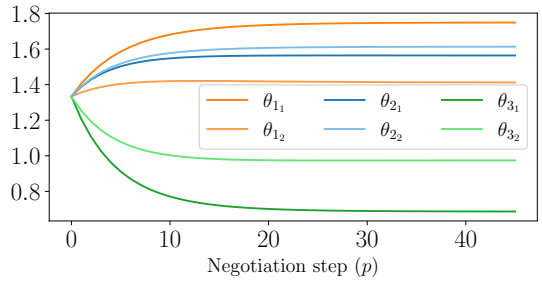
If the initial $\boldsymbol{\theta}^{(0)}$ belongs to $\mathcal{S}$, we can rewrite for each individual $\boldsymbol{\theta}_i$:

$$\boldsymbol{\theta}_i^{(p+1)} = \boldsymbol{\theta}_i^{(p)} + \rho^{(p)}\left(\boldsymbol{\lambda}_i^{(p)} - \frac{1}{M}\sum_{j=1}^{M}\boldsymbol{\lambda}_j^{(p)}\right), \forall i \in \mathcal{M} \tag{5.15}$$

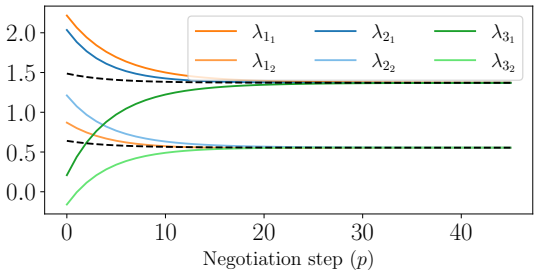
Using (5.12) and (5.15) we can program Algorithm 5.2 and simulate the system for 10 time steps ($k \in \mathcal{K} = \{0:9\}$). The dynamics of variables $\boldsymbol{\theta}_i[0]$ and $\boldsymbol{\lambda}_i[0]$ during the iterative process can be seen in Figs. 5.2a and 5.2b.

We can see in Fig. 5.2c the evolution of states $x_i[k]$ during the simulation. As we can perceive, the system cannot achieve the references due to the global coupling constraint, so all agents find a compromise and get as close to the reference as possible.

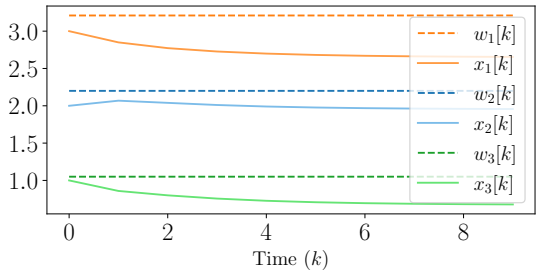
In Figs. 5.2a and 5.2b the sub-systems were color-coded in nuances of orange for agent 1, of blue for agent 2, and of green for agent 3. Observe in Fig. 5.2a that the agent who has bigger values of $\boldsymbol{\lambda}_i$ has a corresponding $\boldsymbol{\theta}_i$ which increases with the iterations and vice versa.



(a) Evolution of $\boldsymbol{\theta}_i$ during iterative process.



(b) Evolution of $\boldsymbol{\lambda}_i$ during iterative process.



(c) Evolution of states $x_i[k]$ and references $w_i[k]$ during control $k \in \mathcal{K}$.

Figure 5.2 – Evolution of variables during iterative process and control.

For $\boldsymbol{\lambda}_i$, we see it stabilizes when values for corresponding elements of the vector meet. It is true due to the global equality constraint, whose simple projection onto the intersection of hyperplanes result in (5.15). Moreover, as one can see, the iterative process converges if $\boldsymbol{\theta}_i^{(p+1)} = \boldsymbol{\theta}_i^{(p)}$, which happens when $\boldsymbol{\lambda}_i^{(p)} = \frac{1}{M}\sum_{j=1}^{M} \boldsymbol{\lambda}_j^{(p)}$. Although it may not necessarily be true for the inequality cases, it gives us an insight into the functioning of the algorithm. The values of $\boldsymbol{\theta}_i$ represent an allocation of the maximum input given to the agents by the coordinator, and $\boldsymbol{\lambda}_i$ is a measure of dissatisfaction of each agent to the value allocated. So the coordinator has as role to negotiate the values of the allocations so all agents can be equally dissatisfied. Due to this process of iterative allocations, other names given for this decomposition are **resource allocation** and **quantity decomposition**.

We can create a parallel with the cake-cutting problem [BT95] where two (or more) agents want to share a cake, dividing it equally so no agent envies the piece of cake of other agents. Usually, in these kinds of problems, a protocol is created where agents cut successively the cake (or parts of it), and the other agents take parts that they think are the greater. But in those cases, usually, all agents have the same hunger.

In our case, agents may have different needs, represented by the dynamics of the sub-systems (tuples (A_i, B_i)), the matrices $\bar{\Gamma}_i$, state $x_i[k]$ and references $w_i[k]$. So, the coordinator works as a mediator, positioning the cake cutter, showing where it would cut the cake (sending $\boldsymbol{\theta}_i$ during the iterative process), and listening to see if the agents would be satisfied with the partition (receiving $\boldsymbol{\lambda}_i$) and adjusting the cake cutter until a consensus is reached. Due to this interpretation, we call the iterative process the **negotiation phase**.

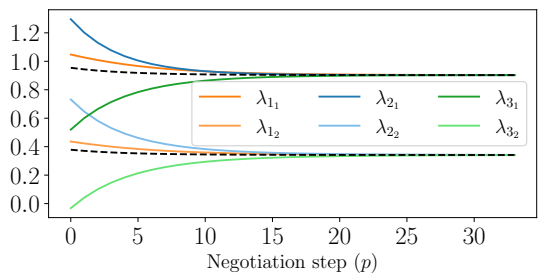
5.3 Anomalous behaviors and their consequences

Since the negotiation (5.9) on the primal decomposition depends on $\boldsymbol{\theta}_i[k]$ and $\boldsymbol{\lambda}_i[k]$, the main source of vulnerabilities is the communication between local agents and coordinator.

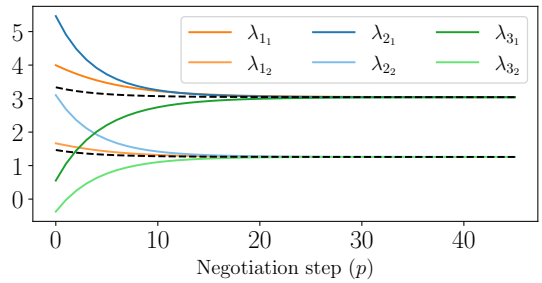
For the equality case shown, it is clear to see that fluctuations in $\boldsymbol{\theta}_i[k]$ alter the results of local problems (5.12), and analogously, fluctuations in $\boldsymbol{\lambda}_i[k]$ alter the negotiation. An ill-intentioned local agent can exploit those facts to drive the negotiation by means of a FDI attack. However, the system could be implemented in such way that the final allocations agreed upon are distributed physically.

With the physical distribution of allocation, **"Liar" agent attacks** as in [Vel+17b] are discouraged since the attacker can not consume more than allocated (which would be beneficial for the attacker). On the other hand, an attack in which an agent is forced to use less than the allocated would still be possible, but no agent would benefit, given that their allocation would not increase (allocations are fixed at the end of negotiation). We will call this kind of attack where an agent is forced to deteriorate its own performance as a **Hijack agent** attack. Moreover, during the following negotiations, the attacked agent would need more and more resources, increasing as a consequence their dissatisfaction $\boldsymbol{\lambda}_i[k]$, forcing other agents to share more resources with the attacked agent.

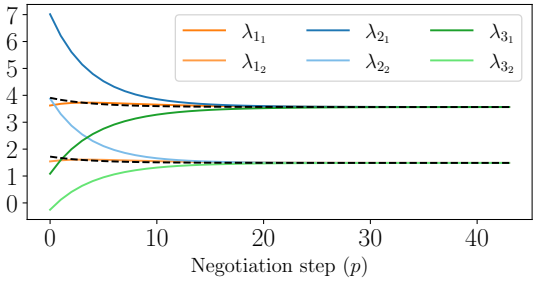
We use the same example in §5.2, but we modify agent 2, so it does not use the resource allocated (the agent does not apply $u_2^\star[0|k]$). In Figs. 5.3a, 5.3b and 5.3c we can compare the values of $\boldsymbol{\lambda}_2$ for different time steps k. As we see, the values of $\boldsymbol{\lambda}_2$ increase as the time step k increases, driving as a consequence all other values of $\boldsymbol{\lambda}_i$ and $\boldsymbol{\theta}_i$. If we compare Fig. 5.4 with Fig. 5.2c we see how the performance is degraded for all agents.



(a) $k = 0$.



(b) $k = 4$.



(c) $k = 9$.

Figure 5.3 – Evolution of $\boldsymbol{\lambda}_i$ for different k when agent 2 is hijacked.

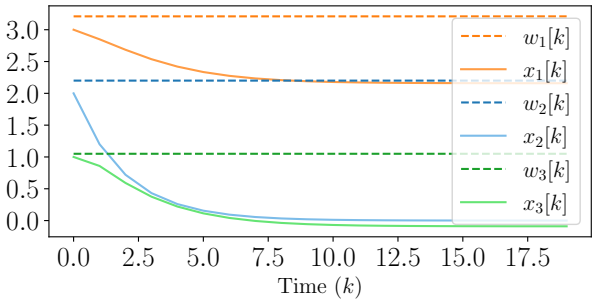


Figure 5.4 – Evolution of state $x_i[k]$ and reference $w_i[k]$ during control when agent 2 is hijacked.

5.3.1 Attack of interest

As illustrated, the change in $\boldsymbol{\lambda}_i$ may cause a modification in the consensus and drive the negotiation. Seeing that $\boldsymbol{\lambda}_i$ form a sub-gradient of the local problems (5.6a), it is perceivable that they carry information about the local problem, i.e. it has somehow embedded in itself information about the 5-tuple $(H_i, \boldsymbol{f}_i[k], \bar{\Gamma}_i, \boldsymbol{\theta}_i[k])$ which we will call the complete local information $\mathcal{I}_i$. So, every attack shown in [Vel+18] (explained in §4.5) will somehow reflect into a modification in $\boldsymbol{\lambda}_i$. And in the coordinator's perspective, what matters is what it receives.

For this reason, we are interested in inter-agent FDI attacks, in which one (or a group) of the agents changes its own $\boldsymbol{\lambda}_i$, so it can benefit at the expense of others. We suppose that before a negotiation, the attacker chooses a function $\gamma : \mathbb{R}^c \times \mathbb{R} \to \mathbb{R}^c$, which can possibly vary with respect to time k, to modify $\boldsymbol{\lambda}_i$ before sending it to the coordinator. This function aggregates any modification on the local information $\mathcal{I}_i$. Here we denote this modified version as

$$\tilde{\boldsymbol{\lambda}}_i = \gamma(\boldsymbol{\lambda}_i[k], k). \tag{5.16}$$

To facilitate future analysis, we suppose the attacker chooses a linear function such

$$\tilde{\boldsymbol{\lambda}}_i[k] = \gamma(\boldsymbol{\lambda}_i[k], k) = T_i[k]\boldsymbol{\lambda}_i[k]. \tag{5.17}$$

To ease communication, we will call the functions γ_i and the matrices $T_i[k]$ as the **cheating** functions and matrices.

This attack can be seen as a general case of the selfish attack [Vel+18], where a positive constant multiplies the local objective function of the attacker. This attack can be translated into a matrix

$$T_i[k] = \tau_i[k] I_c. \tag{5.18}$$

As we will see, even being a simple attack, this attack can have some catastrophic consequences when associated with the negotiation dynamics.

To illustrate, we take again the same example system in §5.2. We modify it so agent 3 attacks using (A.22). We simulate the system for different scenarios. At each one of the scenarios, agent 3 uses a different cheating coefficient $\tau_3[k]$.

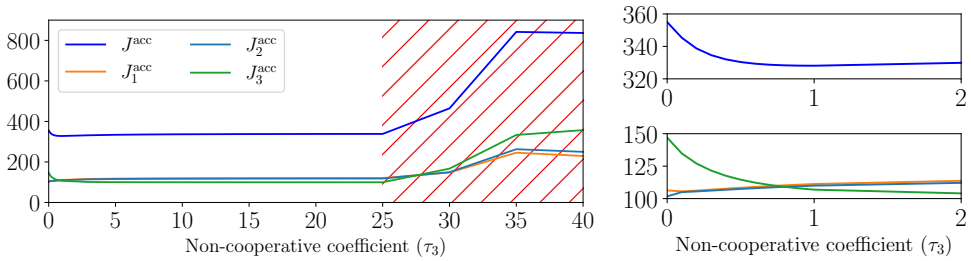
In Fig. 5.5a and 5.5b, we accumulate the local objective functions $J_i^{\text{acc}} = \sum_{k\in\mathcal{K}} J_i^\star(\boldsymbol{\theta}_i^\star[k])[k]$ and also the total $J^{\text{acc}} = \sum_{i\in\mathcal{M}} J_i^{\text{acc}}$ for different values of τ_3.

Remark 5.3

Here, as observed in Remark 2.1, the objective functions $J_i^\star[k]$ and $J^\star[k]$ were calculated by reconstructing the original objective function (2.13), i.e., calculating equivalent $c_i[k]$ for each subsystem and using

$$J_i^\star[k] = 2\bar{J}_i^\star[k] + c_i[k]. \tag{5.19}$$

Observe (Fig. 5.5b) that J^{acc} has its least value when $\tau_3 = 1$, i.e., when there is no attack. When τ_3 increases, the objective function of the attacker (J_3^{acc}) decreases while all other increase. We see the opposite happening for values of τ_3 between 0 and 1. In this case, the attacker deteriorates its own performance, which would not justify such an attack unless in a hijack scenario.

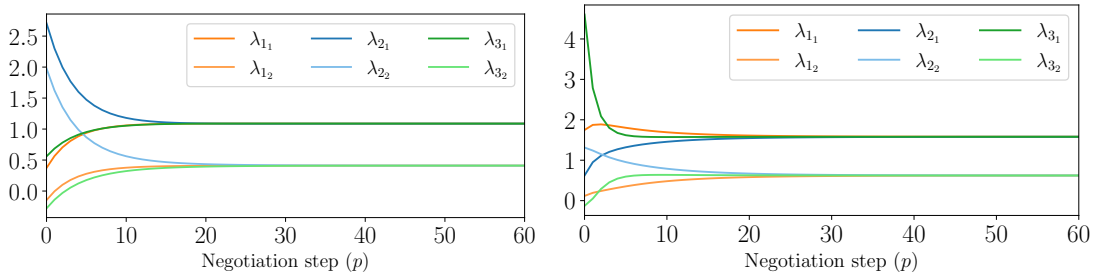


(a) (b) (detail)

Figure 5.5 – Accumulate objective functions for different values of τ_3.

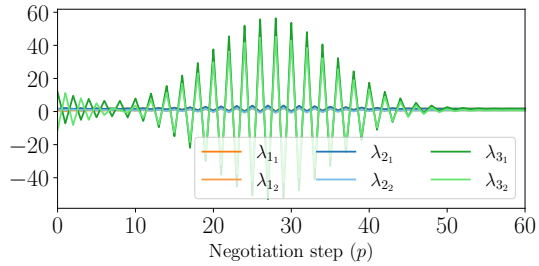
Another fact we can see in Fig. 5.5a is on its right side (hatched in red). The objectives seem to find an equilibrium, but after a given value of τ_3, it increases more and more. This is caused by a collapse in the negotiation. The agents cannot find a consensus anymore. To illustrate, we present the negotiation of $\boldsymbol{\lambda}_i$ for 4 different values of τ_3 (Figs. A.14a to A.14d).

As we can see, as τ_3 increases, the negotiation presents a dampened oscillatory behavior. The dampening decreases if τ_3 continues to rise until the negotiation oscillates and no consensus is reached, breaking down the dMPC and maybe causing damage to the plant.

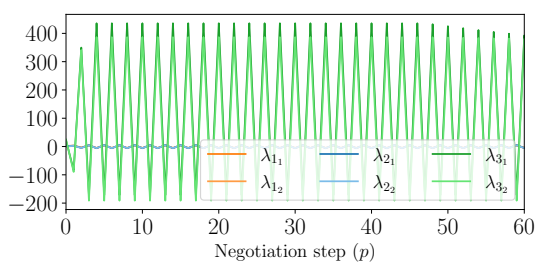


(a) $\tau_3 = 1$.

(b) $\tau_3 = 5$.



(c) $\tau_3 = 20$.



(d) $\tau_3 = 40$.

Figure 5.6 – Evolution of $\boldsymbol{\lambda}_i$ during negotiation for $k = 5$ with different values of τ_3.

5.4 Conclusion

In this chapter, we presented the primal decomposition-based dMPC and its vulnerabilities. We also provided means by which an ill-intentioned agent could exploit said vulnerabilities and their main effects. As we have seen, the main consequences are a loss in global performance and a complete malfunction of the system.

Degradation of global performance, when working with CPS, in the best of scenarios, can cause little to no discomfort for a group of users, but an eventual breakdown may endanger the users. So, we desire to avoid both phenomena whenever possible. In the next chapters, we will analyze the problem and discuss what we can do to avoid such effects.

Chapter 6

Resilient Primal Decomposition-based distributed Model Predictive Control for deprived systems

So you tell me 'trust me'
I can trust you
Just let me show you
But I gotta work it out in a shadow of doubt
'Cause I don't know if I know you

Lie
Dream Theater

In this chapter, we focus on the analysis of the primal decomposition-based dMPC problem for deprived systems when in the presence of the attack chosen in §4.

With the knowledge acquired after the analysis, we propose a method to detect the attacks and mitigate their effects.

Contents

6.1 Deprived systems under attack 82
6.1.1 Deprived Systems 82
6.1.2 Why deprived systems? 84
6.1.3 Analysis of local problems 84
6.1.4 Analysis of negotiation 86
6.2 A detection mechanism 90
6.2.1 Considerations about parameter estimation 91
6.2.2 Complete detection technique 94
6.3 Mitigation 94
6.4 Complete safe dMPC for deprived systems 98

6.5 Numerical experiment 98
6.5.1 District Heating Network 99
6.5.2 Applying RPdMPC-DS 101
6.6 Conclusion 103

6.1 Deprived systems under attack

Different from [Vel+18; Mae+21] which propose robust solutions, we propose a resilient dMPC based on a hybrid of analytical/learning active detection method. To this end, we begin with an analysis of the system under attack.

6.1.1 Deprived Systems

First we recall the monolithic MPC equivalent problem (2.17), once more reproduced for the reader's convenience,

$$\begin{array}{ll} \underset{\boldsymbol{U}[k]}{\text{minimize}} & \frac{1}{2}\left\|\boldsymbol{U}[k]\right\|_H^2 + \boldsymbol{f}[k]^T\boldsymbol{U}[k] \\ \text{subject to} & \bar{\Gamma}\boldsymbol{U}[k] \preceq \boldsymbol{U}_{\max} \end{array}, \tag{2.17}$$

and the local problems (5.6a) solved in the primal decomposition, also reproduced,

$$\bar{J}_i^\star(\boldsymbol{\theta}_i[k])[k] = \begin{array}{ll} \underset{\boldsymbol{U}_i[k]}{\text{minimize}} & \frac{1}{2}\left\|\boldsymbol{U}_i[k]\right\|_{H_i}^2 + \boldsymbol{f}_i[k]^T\boldsymbol{U}_i[k] \\ \text{subject to} & \bar{\Gamma}_i\boldsymbol{U}_i[k] \preceq \boldsymbol{\theta}_i[k] : \boldsymbol{\lambda}_i[k] \end{array}. \tag{5.6a}$$

The unconstrained version of problems (2.17) and (5.6a) are

$$\underset{\boldsymbol{U}[k]}{\text{minimize}} \quad \frac{1}{2}\left\|\boldsymbol{U}[k]\right\|_H^2 + \boldsymbol{f}[k]^T\boldsymbol{U}[k] \quad , \tag{6.1}$$

$$\underset{\boldsymbol{U}_i[k]}{\text{minimize}} \quad \frac{1}{2}\left\|\boldsymbol{U}_i[k]\right\|_{H_i}^2 + \boldsymbol{f}_i[k]^T\boldsymbol{U}_i[k], \tag{6.2}$$

and have analytical solutions [BV04]

$$\mathring{\boldsymbol{U}}^\star = -H^{-1}\boldsymbol{f}[k], \tag{6.3}$$

$$\mathring{\boldsymbol{U}}_i^\star[k] = -H_i^{-1}\boldsymbol{f}_i[k]. \tag{6.4}$$

We call a system deprived when its unconstrained solution $\mathring{\boldsymbol{U}}^\star$ does not respect any of the constraints for all k, i.e.,

$$\bar{\Gamma}\mathring{\boldsymbol{U}}^\star \succ \boldsymbol{U}_{\max}, \forall k. \tag{6.5}$$

Furthermore, in this chapter, we assume that for all sub-systems, their unconstrained

solutions $\mathring{\boldsymbol{U}}_i^\star[k]$ neither respect the constraints

$$\bar{\Gamma}_i \mathring{\boldsymbol{U}}_i^\star[k] > \boldsymbol{\theta}_i[k], \forall i \in \mathcal{M}, \forall k, \tag{6.6}$$

i.e., even if all resources were allocated for a single sub-system, it still could not fulfill its needs.

These assumptions mean that the optimal solution for each problem would need more resources (more than $\boldsymbol{U}_{\max}$). So, the solution to those QP problems is the weighted [1] projection of the unconstrained solutions onto the polytope. Thus, projecting onto the perimeter of the polytope.

We assume that the given projection solves the inequality-constrained problem equivalent to an equality-constraint problem

$$\begin{array}{ll} \underset{\boldsymbol{U}[k]}{\text{minimize}} & \frac{1}{2}\left\|\boldsymbol{U}[k]\right\|_H^2 + \boldsymbol{f}[k]^T\boldsymbol{U}[k] \\ \text{subject to} & \bar{\Gamma}_{\text{eq}}\boldsymbol{U}[k] = \boldsymbol{U}_{\max} \end{array} . \tag{6.7}$$

And as one could perceive, it leads to a decomposition exactly as in §5.2 with local problems (5.12) and negotiation equation (5.15), reproduced here,

$$\bar{J}_i^\star(\boldsymbol{\theta}_i[k])[k] = \begin{array}{ll} \underset{\boldsymbol{U}_i[k]}{\text{minimize}} & \frac{1}{2}\left\|\boldsymbol{U}_i[k]\right\|_{H_i}^2 + \boldsymbol{f}_i[k]^T\boldsymbol{U}_i[k] \\ \text{subject to} & \bar{\Gamma}_i\boldsymbol{U}_i[k] = \boldsymbol{\theta}_i[k] : \boldsymbol{\lambda}_i[k] \end{array} \tag{5.12}$$

$$\boldsymbol{\theta}_i^{(p+1)} = \boldsymbol{\theta}_i^{(p)} + \rho^{(p)}\left(\boldsymbol{\lambda}_i^{(p)} - \frac{1}{M}\sum_{j=1}^{M}\boldsymbol{\lambda}_j^{(p)}\right), \forall i \in \mathcal{M}. \tag{5.15}$$

These are the equations we will use for our analysis.

Remark 6.1

Observe that not necessarily $\bar{\Gamma}_{\text{eq}}$ (or $\bar{\Gamma}_{\text{eq}}$) is the same as $\bar{\Gamma}$ (or $\bar{\Gamma}_i$). The constraints in (2.17) and (5.6a), can be interpreted as the intersection of the halfspaces described by the rows of $\bar{\Gamma}$ (or $\bar{\Gamma}_i$) and $\boldsymbol{U}_{\max}$ (or $\boldsymbol{\theta}_i[k]$), i.e., the halfspaces $\mathcal{S}_i = \{\boldsymbol{x} \mid \bar{\Gamma}_{(i,\star)}\boldsymbol{x} \leqslant \boldsymbol{U}_{\max(i,\star)}\}$, and the intersection $\mathcal{S} = \{\boldsymbol{x} \mid \bar{\Gamma}\boldsymbol{x} \preceq \boldsymbol{U}_{\max}\} = \bigcap_{i=1}^{c}\mathcal{S}_i$ (and conversely for $\bar{\Gamma}_i$ and $\boldsymbol{\theta}_i[k]$).

So, to find the equivalent $\bar{\Gamma}_{\text{eq}}$ (or $\bar{\Gamma}_{i_{\text{eq}}}$) for the equality constraint problem, only the active halfspaces are kept and transformed into the intersection of hyperplanes (equality constraint). This computation of active/inactive sets may be costly depending on the number of halfspaces used and the dimensions of the variables. Some binary-search-tree

1. A QP problem can be seen as weighted projection, in which the set measure is weighted by H and the original pointed is translated using $\boldsymbol{f}_i[k]$

and other techniques may be used to increase performance [SBR22; AB09].

If we assume the feasible regions of $\bar{\Gamma}$ (and $\bar{\Gamma}_i$) to form a cone, if the systems are deprived, $\bar{\Gamma}_{\text{eq}} = \bar{\Gamma}$ (and $\bar{\Gamma}_{i_{\text{eq}}} = \bar{\Gamma}_i$) [a].

One way to ensure this is to make the original constraint (2.7) to have at most as many rows as columns, i.e., $\# \boldsymbol{u}_{\max} \leqslant n_u$, although it may be a little restrictive.

So, from now on, we make this assumption and use $\bar{\Gamma}$ and $\bar{\Gamma}_i$.

a. This can be easily proven geometrically when a point do not respect any of the inequality constraints the projection onto the cone will be its apex (which coincides with the intersection of the equivalent hyperplanes).

6.1.2 Why deprived systems?

When a system does not suffer from scarcity, the solution of the unconstrained problem (6.3) respects the inequalities, i.e.,

$$\bar{\Gamma}\mathring{\boldsymbol{U}}^{\star} \preceq \boldsymbol{U}_{\max}, \forall k. \tag{6.8}$$

This way, the elements of $\mathring{\boldsymbol{U}}^{\star}$ lie inside the polytope, and all the constraints are inactive. As a consequence, no projection onto the polytope is needed since it would result in the same point. That means the solution to the constrained problem is the same as the unconstrained, so the problem is, in fact, unconstrained.

Instead of needing to solve the local problems (5.6a) and use a negotiation to decompose the system, the problem does not need coordination. The problem is solved as shown in §2.2.2, i.e., each sub-system solves (6.3) and the solution is already given.

This case is uninteresting because since the agents do not need to compromise to find a consensus, there is no incentive to attack the system to gain more resources since it would not be used, and it would not affect the others, who would be nevertheless satisfied. Furthermore, since no negotiation is needed, there is no communication, and the agent would need to find other ways to attack the system.

When the system suffers from scarcity, the agents are forced to compete and compromise, which may result in attacks as shown in §5 and their dire consequences.

6.1.3 Analysis of local problems

Since the optimization problems (5.12) are QP problems with equality constraints, it is known that they have analytical solution [BV04]. To find this solution, we usually use the Lagrange duality, so $\boldsymbol{\lambda}_i[k]$ will also have an analytical solution in this case. First, to simplify the notation we define a function $h_i : \mathbb{R}^{\#\boldsymbol{U}_i[k]} \times \mathbb{R}^c$, defined as

$$h_i(\boldsymbol{U}_i[k], \boldsymbol{\theta}_i[k]) = \bar{\Gamma}_i \boldsymbol{U}_i[k] - \boldsymbol{\theta}_i[k], \tag{6.9}$$

to put the problem in standard form for equality constraint QP programs:

$$\begin{aligned} \underset{\boldsymbol{U}_i[k]}{\text{minimize}} \quad & J_i(\boldsymbol{U}_i[k], \boldsymbol{\theta}_i[k]) \\ \text{subject to} \quad & h_i(\boldsymbol{U}_i[k], \boldsymbol{\theta}_i[k]) = 0 : \boldsymbol{\lambda}_i[k] \end{aligned} \tag{6.10}$$

With this form, we define the *Lagrangian* function $\mathscr{L} : \mathbb{R}^{\#\boldsymbol{U}_i[k]} \times \mathbb{R}^c \times \mathbb{R}^c \to \mathbb{R}$ as

$$\mathscr{L}(\boldsymbol{U}_i[k], \boldsymbol{\lambda}_i[k], \boldsymbol{\theta}_i[k]) = J_i(\boldsymbol{U}_i[k], \boldsymbol{\theta}_i[k]) + \boldsymbol{\lambda}_i[k]^T h_i(\boldsymbol{U}_i[k], \boldsymbol{\theta}_i[k]), \tag{6.11}$$

which embeds the equality constraint into the objective function using $\boldsymbol{\lambda}_i[k]$, called the lagrangian multiplier for equality. In this chapter, since there is only one type of lagrangian multiplier, we will call it the dual variable.

Remark 6.2

Observe that by complementary slackness, $\boldsymbol{\lambda}_i^\star[k]^T h_i(\boldsymbol{U}_i^\star[k], \boldsymbol{\theta}_i[k])$ will always be zero. If any element of $h_i(\boldsymbol{U}_i^\star[k], \boldsymbol{\theta}_i[k])$ is different from zero, the corresponding element on the dual variable $\boldsymbol{\lambda}_i^\star[k]$ will be zero, indicating the constraint is inactive.

Then we define the dual function $\mathscr{D} : \mathbb{R}^c \times \mathbb{R}^c \to \mathbb{R}$ which is calculated by solving

$$\mathscr{D}(\boldsymbol{\lambda}_i[k], \boldsymbol{\theta}_i[k]) = \inf_{\boldsymbol{U}_i[k] \in \mathcal{F}} \mathscr{L}(\boldsymbol{U}_i[k], \boldsymbol{\lambda}_i[k], \boldsymbol{\theta}_i[k]), \tag{6.12}$$

where $\mathcal{F} = \operatorname{dom} h_i = \bigcap_{i=1}^{c} \mathcal{P}_i$, with $\mathcal{P}_i$ being the hyperplanes $\mathcal{P}_i = \{\boldsymbol{x} \mid \bar{\Gamma}_{(i,\star)}\boldsymbol{x} = \boldsymbol{U}_{\max(i,\star)}\}$.

The solution to $\mathscr{D}(\boldsymbol{\lambda}_i[k], \boldsymbol{\theta}_i[k])$ is given by the first order Karush-Kuhn-Tucker (KKT) optimality condition

$$\nabla_{\boldsymbol{U}_i[k]} \mathscr{L}(\boldsymbol{U}_i[k], \boldsymbol{\lambda}_i[k], \boldsymbol{\theta}_i[k]) = 0 \tag{6.13}$$

resulting in

$$\boldsymbol{U}_i[k] = H_i^{-1}(-\bar{\Gamma}_i^T \boldsymbol{\lambda}_i[k] - \boldsymbol{f}_i[k]). \tag{6.14}$$

which is

$$\boldsymbol{U}_i[k] = \mathring{\boldsymbol{U}}_i^\star[k] - H_i^{-1}\bar{\Gamma}_i^T \boldsymbol{\lambda}_i[k]. \tag{6.15}$$

Substituting it in $\mathscr{L}(\boldsymbol{U}_i[k], \boldsymbol{\lambda}_i[k], \boldsymbol{\theta}_i[k])$ yields

$$\mathscr{D}(\boldsymbol{\lambda}_i[k], \boldsymbol{\theta}_i[k]) = -\frac{1}{2}\boldsymbol{\lambda}_i[k]^T \bar{\Gamma}_i H_i^{-1} \bar{\Gamma}_i{}^T \boldsymbol{\lambda}_i[k] - \boldsymbol{\lambda}_i[k]^T \bar{\Gamma}_i H_i^{-1} \boldsymbol{f}[k] - \boldsymbol{\lambda}_i[k]^T \boldsymbol{\theta}_i[k]. \tag{6.16}$$

As presented in [BV04] and as we can see from inspecting (6.16), $\mathscr{D}(\boldsymbol{\lambda}_i[k], \boldsymbol{\theta}_i[k])$ is concave. Thus, we can find $\boldsymbol{\lambda}_i^\star[k]$ by solving

$$\boldsymbol{\lambda}_i^\star[k] = \underset{\boldsymbol{\lambda}_i[k]}{\operatorname{argmax}} \quad \mathscr{D}(\boldsymbol{\lambda}_i[k], \boldsymbol{\theta}_i[k]). \tag{6.17}$$

Similarly, using first-order KKT condition we make the gradient of $\mathscr{D}(\boldsymbol{\lambda}_i[k], \boldsymbol{\theta}_i[k])$ vanish

$$\nabla_{\boldsymbol{\lambda}_i[k]} \mathscr{D}(\boldsymbol{\lambda}_i[k], \boldsymbol{\theta}_i[k]) = 0, \tag{6.18}$$

which yields

$$\boldsymbol{\lambda}_i[k] = -P_i \boldsymbol{\theta}_i[k] - \boldsymbol{s}_i[k], \tag{6.19}$$

where $P_i = (\bar{\Gamma}_i H_i^{-1} \bar{\Gamma}_i^T)^{-1}$ and $\boldsymbol{s}_i[k] = P_i \bar{\Gamma}_i H_i^{-1} \boldsymbol{f}_i[k]$.

As we expected, the value of $\boldsymbol{\lambda}_i[k]$ depends on $\boldsymbol{\theta}_i[k]$. Furthermore, because of the quadratic form, the solution is an affine function.

Remark 6.3

One can notice, as foreshadowed in §5.3.1, that the solution embeds information not only of the objective function and the reference and state (presence of H_i and $\boldsymbol{f}_i[k]$), but also of the constraints (presence of $\bar{\Gamma}_i$ and $\boldsymbol{\theta}_i[k]$). So, changes in these parameters affect the resulting value of $\boldsymbol{\lambda}_i[k]$.

6.1.4 Analysis of negotiation

Using the results of equation (5.15), we can observe the dynamics of $\boldsymbol{\theta}_i$ and $\boldsymbol{\lambda}_i$ during a negotiation and, from this, get some insight for future detection and mitigation methods.

Dynamics of $\boldsymbol{\lambda}_i$

We can analyze the dynamics of $\boldsymbol{\lambda}_i$ by substituting (5.15) into (6.19), which yields

$$\boldsymbol{\lambda}_i^{(p+1)} = -P_i \left(\boldsymbol{\theta}_i^{(p)} + \rho^{(p)} \left(\boldsymbol{\lambda}_i^{(p)} - \frac{1}{M} \sum_{j=1}^{M} \boldsymbol{\lambda}_j^{(p)} \right) \right) - \boldsymbol{s}_i[k]. \tag{6.20}$$

We can distribute the terms, resulting in

$$\boldsymbol{\lambda}_i^{(p+1)} = -P_i \rho^{(p)} \boldsymbol{\lambda}_i^{(p)} + P_i \rho^{(p)} \frac{1}{M} \sum_{j=1}^{M} \boldsymbol{\lambda}_j^{(p)} - P_i \boldsymbol{\theta}_i^{(p)} - \boldsymbol{s}_i[k]. \tag{6.21}$$

From inspection of (6.19), the last term $(-P_i \theta_i^{(p)} - \boldsymbol{s}_i[k])$ is equals to $\boldsymbol{\lambda}_i^{(p)}$, substituting it into (6.21) and using a matrix form, we have

$$\boldsymbol{\lambda}^{(p+1)} = \mathcal{A}_\lambda \boldsymbol{\lambda}^{(p)}, \tag{6.22}$$

with

$$\mathcal{A}_\lambda = \begin{bmatrix} I - \frac{M-1}{M}\rho^{(p)}P_1 & \frac{1}{M}\rho^{(p)}P_1 & \dots & \frac{1}{M}\rho^{(p)}P_1 \\ \frac{1}{M}\rho^{(p)}P_2 & I - \frac{M-1}{M}\rho^{(p)}P_2 & \dots & \frac{1}{M}\rho^{(p)}P_2 \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{M}\rho^{(p)}P_M & \frac{1}{M}\rho^{(p)}P_M & \dots & I - \frac{M-1}{M}\rho^{(p)}P_M \end{bmatrix}, \tag{6.23}$$

which is a DT homogeneous system. The system varies with time since ρ changes with p, but usually, it vanishes as it approaches infinity, as seen in (5.8). So, we can use Lyapunov criteria [Hes09, §8.6], i.e., the eigenvalues of $\mathcal{A}_\lambda$, must be inside the unit circle for the system to be stable. For smaller systems, we can use Jury's criteria to test the system [Jur62].

Analyzing $\mathcal{A}_\lambda$ a little further, we see that the sum of its columns is equal to one [2], which means that 1 is a right eigenvalue of $\mathcal{A}_\lambda$:

$$\mathcal{A}_\lambda \mathbf{1} = \mathbf{1}$$

So, supposing the other eigenvalues are inside the unit circle, the system converges to a state $\mathbf{1}c$, i.e., where all $\boldsymbol{\lambda}_i$ are equal [GS10; XBK07]. On the other hand, the sum of rows is not equal, which means the sum of all states is not constant, or as said in the consensus community, the mass is not conserved. This way, we could not use average consensus strategies, unless all subsystems had the same P_i, which would render the problem extremely uninteresting (solution is divide the resources equally).

If we suppose the system is under attack, we can use the attack model (5.17), reproduced here,

$$\tilde{\boldsymbol{\lambda}}_i[k] = T_i[k]\boldsymbol{\lambda}_i[k]. \tag{5.17}$$

Any agent could be the perpetrator of the attack (potentially multiple agents may attack simultaneously), so we use the model for all agents. Remaking the last steps with this attack model, we can finally have the dynamics of the attacked system:

$$\tilde{\boldsymbol{\lambda}}^{(p+1)} = \tilde{\mathcal{A}}_\lambda \tilde{\boldsymbol{\lambda}}^{(p)}, \tag{6.24}$$

with

$$\tilde{\mathcal{A}}_\lambda = \begin{bmatrix} I - \frac{M-1}{M}\rho^{(p)}T_1[k]P_1 & \frac{1}{M}\rho^{(p)}T_1[k]P_1 & \dots & \frac{1}{M}\rho^{(p)}T_1[k]P_1 \\ \frac{1}{M}\rho^{(p)}T_2[k]P_2 & I - \frac{M-1}{M}\rho^{(p)}T_2[k]P_2 & \dots & \frac{1}{M}\rho^{(p)}T_2[k]P_2 \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{M}\rho^{(p)}T_M[k]P_M & \frac{1}{M}\rho^{(p)}T_M[k]P_M & \dots & I - \frac{M-1}{M}\rho^{(p)}T_M[k]P_M \end{bmatrix}, \tag{6.25}$$

2. $I + \frac{M-1}{M}\rho^{(p)}P_i + \sum_{j=1}^{M-1}\frac{1}{M}\rho^{(p)}P_i = I$

As we have attested before through the simple example in §5.3.1, the cheating matrices $T_i[k]$ can change the eigenvalues of the matrix, making the system oscillate and not converge to the optimal value.

Remark 6.4

Observe that as p increases, $\rho^{(p)}$ vanishes, and as consequence the matrix $\mathcal{A}_\lambda$ presents less and less influence of P_i (and $T_i[k]$) and its eigenvalues approach 1. This measure ensures convergence in finite time but with a loss in accuracy.

Dynamics of $\boldsymbol{\theta}_i$

Conversely, we can analyze the dynamics of $\boldsymbol{\theta}_i$ by taking the negotiation equation (5.15), and substituting $\boldsymbol{\lambda}_i[k]$ by (6.19), which yields

$$\boldsymbol{\theta}_i^{(p+1)} = \boldsymbol{\theta}_i^{(p)} + \rho^{(p)} \left((-P_i \boldsymbol{\theta}_i^{(p)} - \boldsymbol{s}_i[k]) - \frac{1}{M} \sum_{j \in \mathcal{M}} (-P_j \boldsymbol{\theta}_j^{(p)} - \boldsymbol{s}_j[k]) \right), \forall i \in \mathcal{M}. \tag{6.26}$$

As we can see, we can separate the function into two parts, one which depends on $\boldsymbol{\theta}_i$ and the other which does not:

$$\boldsymbol{\theta}_i^{(p+1)} = \boldsymbol{\theta}_i^{(p)} + \rho^{(p)} \left(-P_i \boldsymbol{\theta}_i^{(p)} + \frac{1}{M} \sum_{j \in \mathcal{M}} P_j \boldsymbol{\theta}_j^{(p)} \right) + \rho^{(p)} \left(-\boldsymbol{s}_i[k] + \frac{1}{M} \sum_{j \in \mathcal{M}} \boldsymbol{s}_j[k] \right), \forall i \in \mathcal{M}. \tag{6.27}$$

We can finally write it in matrix form:

$$\boldsymbol{\theta}^{(p+1)} = \mathcal{A}_\theta \boldsymbol{\theta}^{(p)} + \boldsymbol{B}_\theta[k] \tag{6.28}$$

where

$$\mathcal{A}_\theta = \begin{bmatrix} I - \frac{M-1}{M}\rho^{(p)} P_1 & \frac{1}{M}\rho^{(p)} P_2 & \dots & \frac{1}{M}\rho^{(p)} P_M \\ \frac{1}{M}\rho^{(p)} P_1 & I - \frac{M-1}{M}\rho^{(p)} P_2 & \dots & \frac{1}{M}\rho^{(p)} P_M \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{M}\rho^{(p)} P_1 & \frac{1}{M}\rho^{(p)} P_2 & \dots & I - \frac{M-1}{M}\rho^{(p)} P_M \end{bmatrix} \tag{6.29}$$

$$\boldsymbol{B}_\theta[k] = \begin{bmatrix} -\frac{M-1}{M}\rho^{(p)} \boldsymbol{s}_1[k] + \frac{1}{M}\rho^{(p)} \boldsymbol{s}_2[k] \cdots - \frac{1}{M}\rho^{(p)} \boldsymbol{s}_M[k] \\ \frac{1}{M}\rho^{(p)} \boldsymbol{s}_1[k] - \frac{M-1}{M}\rho^{(p)} \boldsymbol{s}_2[k] \cdots - \frac{1}{M}\rho^{(p)} \boldsymbol{s}_M[k] \\ \vdots \\ \frac{1}{M}\rho^{(p)} \boldsymbol{s}_1[k] + \frac{1}{M}\rho^{(p)} \boldsymbol{s}_2[k] \cdots - \frac{M-1}{M}\rho^{(p)} \boldsymbol{s}_M[k] \end{bmatrix} \tag{6.30}$$

which is a DT system with forced input ($\boldsymbol{B}_\theta$). We can analyze the stability of $\mathcal{A}_\theta$ with the same tools used for $\mathcal{A}_\lambda$. For it to be stable, the eigenvalues must reside inside the unit circle.

We can also do the same analysis of the sum of the rows and columns. If we observe the sum of rows, we acknowledge it is equal to 1. So, 1 is one of the left eigenvalues of the matrix:

$$\mathbf{1}^T \mathcal{A}_\theta = \mathbf{1}^T \tag{6.31}$$

As expected, the system conserves mass, i.e., the θ_i must respect the global equality constraint. This way, the linear dependence between θ_i is explicit. If we analyze the columns, on the other hand, we see that they do not sum to 1. This means if the system converges, it will not necessarily converge in such a way the θ_i are equal. Again, it would be the case if the P_i were equal.

As we can see, the values of the eigenvalues depend only on the values of P_i and $\rho^{(p)}$. Since the P_i are system parameters, the designer of the control can only tune $\rho^{(p)}$.

Once $\rho^{(p)}$ is well tuned for the corresponding system, the matrix $\mathcal{A}_\theta$ always evolves in the same way. So if there is a change in this evolution, it means the system presents an anomalous behavior.

As can be observed, any disturbance in P_i (change in the objective function or constraints) changes the values of $\mathcal{A}_\theta$, thus the system's dynamic. This change may destabilize it if it drives the eigenvalues outside the unit circle, as shown in §5.3.1. On the other hand, perturbations in $\boldsymbol{s}_i[k]$ (modification of objective function, constraints, or state/reference) may change the steady-state value (change in $\boldsymbol{B}_\theta[k]$).

If we suppose the system is under attack, we again use the attack model (5.17). We suspect all agents, and we modify (6.19) accordingly. Then we substitute it into (5.15). Then, we can have the dynamics of an attacked system

$$\boldsymbol{\theta}^{(p+1)} = \tilde{\mathcal{A}}_\theta[k]\boldsymbol{\theta}^{(p)} + \tilde{\boldsymbol{B}}_\theta[k] \tag{6.32}$$

where

$$\tilde{\mathcal{A}}_\theta[k] = \begin{bmatrix} I - \frac{M-1}{M}\rho^{(p)}T_1[k]P_1 & \frac{1}{M}\rho^{(p)}T_2[k]P_2 & \dots & \frac{1}{M}\rho^{(p)}T_M[k]P_M \\ \frac{1}{M}\rho^{(p)}T_1[k]P_1 & I - \frac{M-1}{M}\rho^{(p)}T_2[k]P_2 & \dots & \frac{1}{M}\rho^{(p)}T_M[k]P_M \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{M}\rho^{(p)}T_1[k]P_1 & \frac{1}{M}\rho^{(p)}T_2[k]P_2 & \dots & I - \frac{M-1}{M}\rho^{(p)}T_M[k]P_M \end{bmatrix} \tag{6.33}$$

$$\tilde{\boldsymbol{B}}_\theta[k] = \begin{bmatrix} -\frac{M-1}{M}\rho^{(p)}T_1[k]\boldsymbol{s}_1[k] + \frac{1}{M}\rho^{(p)}T_2[k]\boldsymbol{s}_2[k] \cdots - \frac{1}{M}\rho^{(p)}T_M[k]\boldsymbol{s}_M[k] \\ \frac{1}{M}\rho^{(p)}T_1[k]\boldsymbol{s}_1[k] - \frac{M-1}{M}\rho^{(p)}T_2[k]\boldsymbol{s}_2[k] \cdots - \frac{1}{M}\rho^{(p)}T_M[k]\boldsymbol{s}_M[k] \\ \vdots \\ \frac{1}{M}\rho^{(p)}T_1[k]\boldsymbol{s}_1[k] + \frac{1}{M}\rho^{(p)}T_2[k]\boldsymbol{s}_2[k] \cdots - \frac{M-1}{M}\rho^{(p)}T_M[k]\boldsymbol{s}_M[k] \end{bmatrix} \tag{6.34}$$

As in the case with $\mathcal{A}_\theta$, by adjusting $T_i[k]$, an attacker can change the eigenvalues of $\mathcal{A}_\lambda$. It can also be adjusted to drive the negotiation to a new point, modifying $\mathcal{B}$, but as we see, it modifies at the same time the dynamics. This way, the attackers need to

compromise between their greediness and not breaking the system, which is worse for every agent, the attackers included.

As one can notice, we could create an anomaly detector by supervising the matrix $\mathcal{A}_\theta$. Once there is a change in some of the variables (or in its eigenvalues) an anomalous behavior is detected.

Unfortunately, this method gives us only detection, but not necessarily from which agent the anomaly comes, which could be useful for a mitigation algorithm. Besides, if we were to estimate all elements of $\mathcal{A}_\theta$, we would need to estimate $M^2(c)^2$ elements [3]. Exploiting the known structures of the matrix and their blocks, we could reduce to at least $2M(\frac{c+c^2}{2})$ elements [4].

In the next section, we derive a detection mechanism, also based on estimation, which can not only detect the attacks but also isolate the perpetrators, estimating only half of the elements cited.

6.2 A detection mechanism

Using (5.17) we can rewrite (6.19) as

$$\tilde{\boldsymbol{\lambda}}_i[k] = -\tilde{P}_i[k]\boldsymbol{\theta}_i[k] - \tilde{\boldsymbol{s}}_i[k], \tag{6.35}$$

where $\tilde{P}_i[k] = T_i[k]P_i$ and $\tilde{\boldsymbol{s}}_i[k] = T_i[k]\boldsymbol{s}_i[k]$. We can interpret it as if the cheating matrix $T_i[k]$ alters both P_i and $\boldsymbol{s}_i[k]$.

Since P_i is constant, any change between two different negotiations is a consequence of anomalous behavior, in this case, caused by $T_i[k]$.

Using the relationship between $\boldsymbol{\theta}_i[k]$ and $\boldsymbol{\lambda}_i[k]$, we can supervise the exchanges between coordinator and agents and find estimates $\widehat{\tilde{P}}_i[k]$ and $\widehat{\tilde{\boldsymbol{s}}}_i[k]$ such

$$\tilde{\boldsymbol{\lambda}}_i[k] = -\tilde{P}_i[k]\boldsymbol{\theta}_i[k] - \tilde{\boldsymbol{s}}_i[k], \tag{6.36}$$

If we have access to a nominal value of P_i, noted $\bar{P}_i$, we can create a detection rule for each agent i. Given an arbitrary bound ϵ_{P_i}, we can define the error

$$E_i[k] = \left\| \widehat{\tilde{P}}_i[k] - \bar{P}_i \right\|_F, \tag{6.37}$$

and an detection function d_i:

$$d_i = \mathbb{1}_{\{E_i[k] \geqslant \epsilon_{P_i}\}}, \tag{6.38}$$

detecting an attack when the bound is disrespected, i.e., $d_i = 1$.

3. M^2 blocks of size c
4. (M diagonal + M off-diagonal) symmetric blocks of size c (only upper triangle)

Remark 6.5

Observe that as shown in Remark 4.3, the choice of ϵ_{P_i} can influence the false positives. For instance, it needs to be bigger than the estimation error in the attack-free scenario. The calculation of the errors and their propagation is not in the scope of this work.

Remark 6.6

Observe that if the estimation does not converge, it means the linear relation ceased to exist, and there is an anomalous behavior, we can also set the indicator to 1.

But yet, we need to estimate $\hat{\bar{P}}_i[k]$. So, a brief discussion about the estimation follows.

Remark 6.7

The nominal values $\bar{P}_i$ may be given or estimated using the same method shown next but using reliable attack-free historical data, for instance, as a cold start of the system.

6.2.1 Considerations about parameter estimation

As commented, we can supervise the exchanges and gather multiple values of $\boldsymbol{\theta}_i$ and $\boldsymbol{\lambda}_i$ to serve as an input dataset to the chosen estimation method.

For instance, we can choose any method in the Least Squares (LS) family [ÅW89]. We have chosen Recursive Least Squares (RLS) because for the normal LS, we would need at least as many input-output tuples $(\boldsymbol{\theta}_i, \boldsymbol{\lambda}_i)$ as parameters to estimate ($\#\, P_i \,\#\, \boldsymbol{s}_i[k]$ elements in the worst case). However, since the values of P_i should not vary from one negotiation to other, if we use the last estimation as the initial condition for a RLS algorithm, we can gain in the time for estimation convergence.

Remark 6.8

Observe that since we are simultaneously estimating $\boldsymbol{s}_i[k]$, which changes w.r.t. time k for smaller systems, the difference between convergence times may be negligible.

In RLS, to estimate a parameter $\boldsymbol{\nu}_i$, we use a regression model using the inputs and outputs (regressors B_i and regressands $\boldsymbol{\lambda}_i$) that enable us to calculate a residual ϵ. This residual, allied with a gain Φ, is used to update estimates of $\boldsymbol{\nu}_i$. Since the exact parameter values are not known initially, an initial estimate is used. To reduce errors caused by the initial estimate and possible discrepant input-output data (noisy data), a forgetting term ϕ is used to update the gain Φ, forgetting past events and maintaining new ones.

Initially, we use (A.46) directly as a regression model, transforming it into the usual form

$$B_i \boldsymbol{\nu}_i = \boldsymbol{\lambda}_i, \tag{6.39}$$

where $\boldsymbol{\nu}_i$ is the parameter vector we want to estimate ($\widehat{\tilde{P}}_i[k]$ and $\widehat{\tilde{\boldsymbol{s}}}_i[k]$ vectorized), and B_i is created using $\boldsymbol{\theta}_i$ but taking into account the entries corresponding $\widehat{\tilde{\boldsymbol{s}}}_i[k]$ inside $\boldsymbol{\nu}_i$, i.e.,

$$\boldsymbol{\nu}_i = \begin{bmatrix} \mathrm{vec}(\widehat{\tilde{P}}_i[k]) \\ \widehat{\tilde{\boldsymbol{s}}}_i[k] \end{bmatrix}, \tag{6.40}$$

$$B_i = \begin{bmatrix} I_c \otimes \boldsymbol{\theta}_i^{\mathrm{T}} & I_c \end{bmatrix}. \tag{6.41}$$

Using this method, we need to estimate all elements of $\widehat{\tilde{P}}_i[k]$ and $\widehat{\tilde{\boldsymbol{s}}}_i[k]$, which results in $c^2 + c$ elements. But as the $\widehat{\tilde{P}}_i[k]$ are symmetric, we can estimate only their upper triangle, reducing the number of total estimated elements to $\frac{c^2+3c}{2}$.

Then, we restructure $\boldsymbol{\nu}_i$ accordingly:

$$\boldsymbol{\nu}_i = \begin{bmatrix} \widehat{\tilde{P}}_i[k]^{\mathrm{T}}_{(1,1:c)} \\ \widehat{\tilde{P}}_i[k]^{\mathrm{T}}_{(2,2:c)} \\ \vdots \\ \widehat{\tilde{P}}_i[k]^{\mathrm{T}}_{(c,c)} \\ \widehat{\tilde{\boldsymbol{s}}}_i[k] \end{bmatrix}. \tag{6.42}$$

However, for B_i, its form becomes more complex, needing an algorithm (Algorithm 6.1) to calculate it:

$$B_i = \begin{bmatrix} \texttt{structDataSym}(\boldsymbol{\theta}_i) & I_c \end{bmatrix}. \tag{6.43}$$

Algorithm 6.1: Struct input for symmetric parameter.

```
structDataSym (x)
    s ← # x
    if s == 1 then
        Y ← x
    else
        Y ← [ [ [0_{s-1}  x_(1) I_{s-1}] ]   0^T_{(s^2-s)/2}
              [ structDataSym(x_(2:s)) ] ]
```

To better grasp the structure, we give a simple example. If we take a $\boldsymbol{\theta}_i$ with $c = 3$

elements, by applying `structDataSym`, it would result in

$$\texttt{structDataSym}(\theta_i) = \begin{bmatrix} \theta_{i(1)} & \theta_{i(2)} & \theta_{i(3)} & 0 & 0 & 0 \\ 0 & \theta_{i(1)} & 0 & \theta_{i(2)} & \theta_{i(3)} & 0 \\ 0 & 0 & \theta_{i(1)} & 0 & \theta_{i(2)} & \theta_{i(3)} \end{bmatrix}. \tag{6.44}$$

Remark 6.9

Observe that for our detection, we do not need to estimate $\boldsymbol{s}_i[k]$. We could use the difference between two different values of $\boldsymbol{\lambda}_i$ to suppress the $\boldsymbol{s}_i[k]$ term. Preventing the estimation of $\widehat{\tilde{\boldsymbol{s}}}_i[k]$ would need some changes in $\boldsymbol{\nu}_i$ and B_i, thus reducing the number of elements to $\frac{c^2+c}{2}$. However, we foreshadow that $\widehat{\tilde{\boldsymbol{s}}}_i[k]$ may be used for the mitigation phase, so we estimate it anyway.

Using the construction of $\boldsymbol{\nu}_i$ and B_i above and applying to the RLS algorithm in [ÅW89], we can describe an iteration of the RLS algorithm to estimate the symmetric matrix $\widehat{\tilde{P}}_i[k]$ and the vector $\widehat{\tilde{\boldsymbol{s}}}_i[k]$ simultaneously. Algorithm 6.2 shows how to calculate a new parameter estimate $\boldsymbol{\nu}_i$.

Algorithm 6.2: Update of RLS to estimate $\widehat{\tilde{P}}_i[k]$ and $\widehat{\tilde{\boldsymbol{s}}}_i[k]$ simultaneously.

Input: Step h, Forgetting rate ϕ, Parameter estimate ${\boldsymbol{\nu}_i}^{(h)}$, Gain $\Phi^{(h)}$, Input $\boldsymbol{\theta}_i^{(h)}$, Output ${\boldsymbol{\lambda}_i}^{(h)}$

Output: New estimate ${\boldsymbol{\nu}_i}^{(h+1)}$, New gain $\Phi^{(h+1)}$

$B_i \leftarrow \left[\texttt{structDataSym}(\boldsymbol{\theta}_i^{(h)}), I_c\right]$

Calculate residual ϵ: $\epsilon \leftarrow \boldsymbol{\lambda}_i - B_i\boldsymbol{\nu}_i$

Update gain Φ: $\Phi^{(h+1)} \leftarrow \phi^{-1}\Phi^{(h)} - \phi^{-2}\Phi^{(h)}{B_i}^{\mathrm{T}}(I_c + \phi^{-1}B_i\Phi^{(h)}{B_i}^{\mathrm{T}})^{-1}B_i\Phi^{(h)}$

Calculate new estimate $\boldsymbol{\nu}_i$: ${\boldsymbol{\nu}_i}^{(h+1)} \leftarrow {\boldsymbol{\nu}_i}^{(h)} + \Phi^{(h+1)}{B_i}^{\mathrm{T}}\epsilon$

Observe that as much we could be tempted to use the algorithm to estimate during the negotiation, such an algorithm would not work throughout the negotiation exchanges. The fact that $\boldsymbol{\lambda}_i$ depends on $\boldsymbol{\theta}_i$ and vice-versa makes them both behave as DT system (as we saw in §6.1.4). This dependency decreases the excitation of the algorithm as the negotiation gets closer to a consensus (converging case) and less new information about the system is given. As seen in [ÅW89], this makes the solution of the LS problem not unique, and the regressor matrix becomes singular. One way to artificially increase the excitation of the inputs is to change the inputs. For instance, the coordinator would supervise the agents by sending $\boldsymbol{\theta}_i$ which do not follow (5.15).

We opted to use a random noise as input, and once the estimates converge, we can resume the negotiation. As seen in [ÅW89, §3.4], a random signal has persistent excitation of any order, i.e., any system will continue to be excited as long the signal is applied.

Remark 6.10

Other considerations about the initialization of $\boldsymbol{\nu}_i$ and the choice of forgetting parameter ϕ can be made but is out of the scope of this work.

6.2.2 Complete detection technique

Using all the pieces presented in the last subsections, we can finally put them together and create a detection algorithm. First, we estimate $\widehat{\tilde{P}}_i[k]$, and then we compare the estimate with the nominal value of P_i. We can systematize the detection technique in Algorithm 6.3.

Algorithm 6.3: Detection algorithm during a given step k (Possibly in parallel).

Input: Nominal $\bar{P}_i$, Maximum number of iterations $h_{\max}$, Bound $\epsilon_{\boldsymbol{\nu}_i}$, Bound ϵ_{P_i},
Output: Detection d_i, Estimates $\widehat{\tilde{P}}_i[k]$ and $\widehat{\tilde{\boldsymbol{s}}}_i[k]$
$h \leftarrow 0$
Initialize ${\boldsymbol{\nu}_i}^{(p)}$
repeat
  Coordinator sends random $\boldsymbol{\theta}_i^{(h)}$ to i
  Subsystem solves (5.6a), and sends $\boldsymbol{\lambda}_i^{\star}[k](\boldsymbol{\theta}_i^{(p)})$ back
  Coordinator updates estimates $\boldsymbol{\nu}_i$ using Algorithm 6.2
  if $[h \geqslant h_{\max}]$ **then**
    Coordinator sets d_i: $d_i \leftarrow 1$
    break
  $h \leftarrow h + 1$
until $\left\| {\boldsymbol{\nu}_i}^{(h+1)} - {\boldsymbol{\nu}_i}^{(h)} \right\| \leqslant \epsilon_{\boldsymbol{\nu}_i}$
Reconstruct $\widehat{\tilde{P}}_i[k]$ and $\widehat{\tilde{\boldsymbol{s}}}_i[k]$ from $\boldsymbol{\nu}_i$
$E_i[k] \leftarrow \left\| \widehat{\tilde{P}}_i[k] - \bar{P}_i \right\|_F$
Coordinator calculates d_i: $d_i \leftarrow d_i \vee \mathbb{1}_{\{E_i[k] \geqslant \epsilon_{P_i}\}}$

Once the attackers are identified via function d_i, we need to mitigate the possible effects of the attack. In the next section, we consider some hints on how to mitigate the effects while changing the decomposition minimally, and after the analysis, we propose a mitigation method.

6.3 Mitigation

The first effect we want to mitigate is the possible destabilization of matrix $\tilde{\mathcal{A}}_\theta[k]$. And the second is the drift caused by changes in $\tilde{\boldsymbol{B}}_\theta[k]$.

As we saw, from the coordinator's perspective, the attack comes from the modification of $\boldsymbol{\lambda}_i$ by the attacking agent. So, the first idea is that instead of using the $\boldsymbol{\lambda}_i$ received

from the attacker, the coordinator uses a modified version, here denoted $\overset{\text{mod}}{\boldsymbol{\lambda}}_i$.

The value of the modified $\overset{\text{mod}}{\boldsymbol{\lambda}}_i$ depends on the type of mitigation chosen. For instance, we can give a simple qualitative example.

Qualitative example

Imagine a simple system composed of two LTIDT 1-dimensional sub-systems, input constrained, controlled using the dMPC framework presented in this work, QP problems and all other conditions set in this chapter. To facilitate visualization, we suppose the prediction horizon is $N = 1$, and the agents have local constraints $\boldsymbol{u}_i \geqslant \mathbf{0}$.

If we plot the level curves of the system (Fig. 6.1), we can see how the unconstrained minimum is projected (weighted projection) onto the hyperplane. The projection yields the constrained solution to the problem (blue dot).

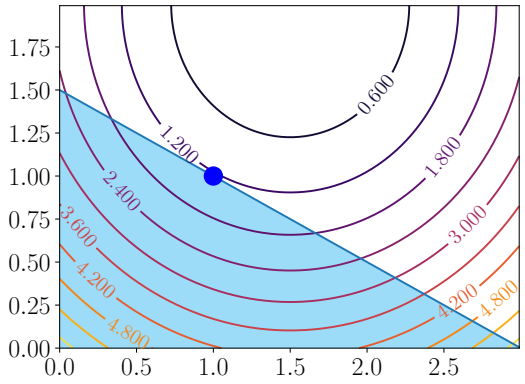


Figure 6.1 – Original minimum.

Once one of the agents attacks the system using (5.17), the level curves are distorted, as seen in Fig. 6.2. This distortion changes the projection, resulting in a new constrained solution (in green dot).

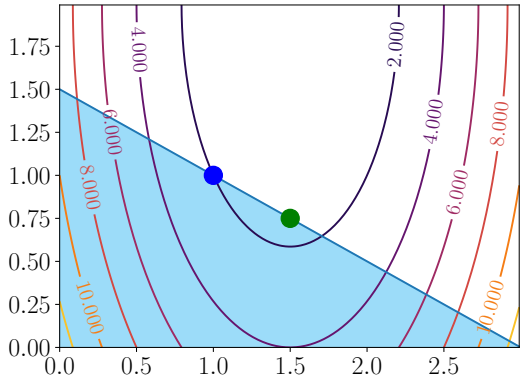


Figure 6.2 – Minimum after attack.

If the attacker is detected, one of our options to mitigate this drift is to substitute the $\tilde{\boldsymbol{\lambda}}_i$ received during the negotiation by $\mathbf{0}$. It can be interpreted as the coordinator

punishing the attacker by assuming it will always be satisfied independent of the allocation given. The negligence against the attacker will result in it receiving fewer and fewer resources. Asymptotically, the system will behave as if the attacker was unplugged since the negotiation will be maintained with the other agents, somehow similar to the examples in [Vel+18; Mae+21].

In Fig. 6.3, we see the new level curves, and the new solution found (orange dot) when ignoring the attacker.

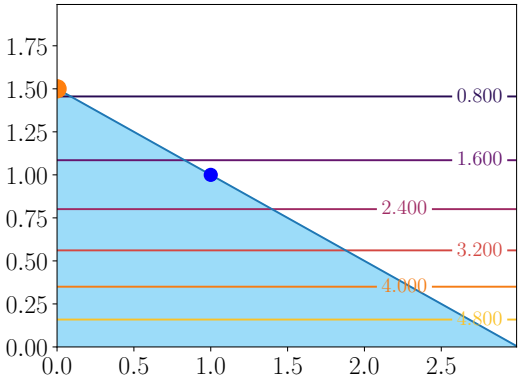


Figure 6.3 – Optimal value after ignoring attacker.

Although the idea of punishing the attacker by ignoring it may appear reasonable, since the system is a CPS, it may control some amenity that needs a minimal Quality of Service (QoS). If we are controlling a Heating, ventilation, and air conditioning (HVAC) system, for instance, depending on the region of the planet, turning the heat off of the attacker could make the users extremely cold, as in the introductory example in Chapter 1.

A solution would be to, before the negotiation, allocate minimal resources for the attacker and then ignore it throughout the negotiation using the remainder of the resources, subtracting the maximum by the allocated for the attacker. However, it would be necessary to assess somehow what would be a fair amount to give to the attacker, which would not prejudice other agents.

We propose yet another solution, in which we try to recover the nominal behavior if some assumptions are guaranteed. Passing from the level curves of the attacked system (Fig. 6.2) to the shown in Fig. 6.4, where the new minimum lies in a neighborhood of the original one (in Fig. 6.1).

Recovering nominal behavior

If we observe the formula of the linear attack (5.17), we can make some assumptions.

As seen in §5.2, $\boldsymbol{\lambda}_i$ correspond to dissatisfaction with the allocation $\boldsymbol{\theta}_i$ given. If $\boldsymbol{\lambda}_i = \mathbf{0}$, it means the agent is satisfied with the allocation. So, we do not expect that the attacker will use a cheating matrix $T_i[k]$ that generates a $\tilde{\boldsymbol{\lambda}}_i = \mathbf{0}$ when $\boldsymbol{\lambda}_i \neq \mathbf{0}$.

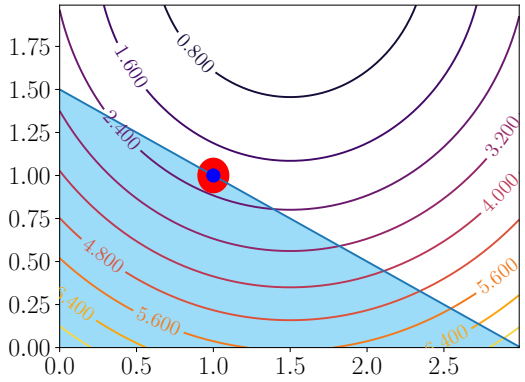


Figure 6.4 – Optimal value after trying to recover original behavior.

Given that we assume

$$\left[\tilde{\boldsymbol{\lambda}}_i = \mathbf{0}\right] \Leftrightarrow \left[\boldsymbol{\lambda}_i = \mathbf{0}\right], \tag{6.45}$$

which translates to

$$\tilde{\boldsymbol{\lambda}}_i = T_i[k]\boldsymbol{\lambda}_i = \mathbf{0}, \text{ if and only if } \boldsymbol{\lambda}_i = \mathbf{0}, \tag{6.46}$$

which implies $T_i[k]$ is **invertible**.

Observing (A.46), we also assume that the choice of $T_i[k]$ does not change the structure of P_i, i.e., $\tilde{P}_i[k]$ is also symmetric and thus also is $T_i[k]$.

Having the estimate $\hat{\tilde{P}}_i[k]$ and the nominal $\bar{P}_i$ and using these assumptions, we can try to estimate the inverse of $T_i[k]$ using

$$\widehat{T_i[k]^{-1}} = \bar{P}_i \hat{\tilde{P}}_i[k]^{-1}. \tag{6.47}$$

And with the estimate $\widehat{T_i[k]^{-1}}$, we can try to reconstruct $\boldsymbol{\lambda}_i$:

$$\overset{\text{rec}}{\boldsymbol{\lambda}}_i = -\bar{P}_i\boldsymbol{\theta}_i - \widehat{T_i[k]^{-1}}\hat{\tilde{\mathbf{s}}}_i[k] \tag{6.48}$$

So, we can modulate the negotiation algorithm by using $\overset{\text{mod}}{\boldsymbol{\lambda}}_i$ choosing what version of $\boldsymbol{\lambda}_i$ to choose, i.e.,

$$\overset{\text{mod}}{\boldsymbol{\lambda}}_i = \begin{cases} \tilde{\boldsymbol{\lambda}}_i, & \text{if } d_i = 0 \\ \overset{\text{rec}}{\boldsymbol{\lambda}}_i, & \text{if } d_i = 1 \end{cases} \tag{6.49}$$

Remark 6.11

Observe that if the estimates of P_i do not converge, we cannot reconstruct $\boldsymbol{\lambda}_i[k]$, so we need to use one of the other options proposed, ignore the attacker, or use minimal allocation. Since it may be difficult to assess the minimal allocation, we ignore the cases when the estimation does not converge.

6.4 Complete safe dMPC for deprived systems

Compiling the detection and mitigation schemes described in the last sections, we can finally have our RPdMPC-DS. The detection mechanism and conditional reconstruction of $\boldsymbol{\lambda}_i$ can be seen as a supervisor for each agent added inside the coordinator (Fig. 6.5)

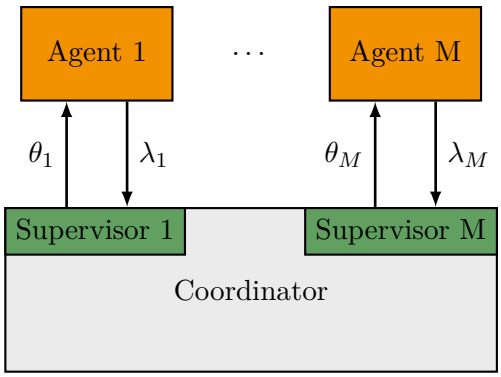


Figure 6.5 – Exchange between agents and coordinator in RPdMPC-DS.

Algorithm 6.4 systematizes the method.

Algorithm 6.4: Resilient Primal Decomposition-based dMPC for deprived systems.

Detection Phase:

Coordinator computes d_i and estimate $\hat{\tilde{P}}_i[k]$ using Algorithm 6.3

Negotiation Phase:

$p \leftarrow 0$

Coordinator initializes $\boldsymbol{\theta}^{(p)}$ and send to subsystems

repeat

Subsystems solve (5.6a), and send $\boldsymbol{\lambda}_i^\star(\boldsymbol{\theta}^{(p)})$

Coordinator updates allocation (5.15) using adequate versions of $\boldsymbol{\lambda}_i$: $\boldsymbol{\lambda}_i^\star(\boldsymbol{\theta}^{(p)})$, if $d_i = 0$ and $\overset{\text{rec}}{\boldsymbol{\lambda}}_i$, if $d_i = 1$ (A.59)

$p \leftarrow p + 1$

until $\left[\left\|\boldsymbol{\theta}[k]^{(p)} - \boldsymbol{\theta}[k]^{(p-1)}\right\| \leqslant \epsilon\right] \vee [p \geqslant p_{\max}]$

Agents apply on respective sub-systems the last calculated $\boldsymbol{u}_i^\star[0|k]$

$k \leftarrow k + 1$

6.5 Numerical experiment

We illustrate the operation of the resilient algorithm just presented with an academic example. This example is more complex and based on reality than the others presented so far. The case studied will be used for the rest of this work with some minor adjustments when needed (inclusion or suppression of assumptions).

6.5.1 District Heating Network

We analyze the case of a simple DHN composed of $M = 4$ distinct small-sized houses (called I, II, II, and IV). Each house has state $\boldsymbol{x}_i$, and a different temperature reference $\boldsymbol{w}_i(t)$ for the inside air (users have different needs). The houses are equipped with convectors, and the total heating power consumed by each house is $\boldsymbol{u}_i(t)$. Since we do not suppose there is any cooling apparatus, the inputs are restricted to be positive, i.e., $\boldsymbol{u}_i[k] \succeq \mathbf{0}$. Meaning that if we need to cool the environment, we can only use the thermic properties of the building. In Fig. 6.6 we see the houses consuming from a centralized heat provider.

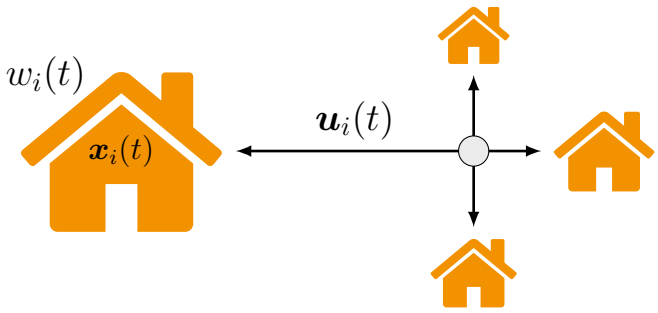


Figure 6.6 – District with 4 houses.

To model each house, we can use the 3R-2C monozone model with a window [GDU02], where the thermic elements are represented as equivalent electric elements in a 3 resistors and 2 capacitors electric circuit. The electric circuit is depicted in Fig. 6.7. Current sources are added to correspond to solar irradiation and heating power. A voltage source represents the temperature outside the house. The meaning of each component is given in Tab. 6.1, and the respective values for each one of the houses are given in Tab. 6.2.

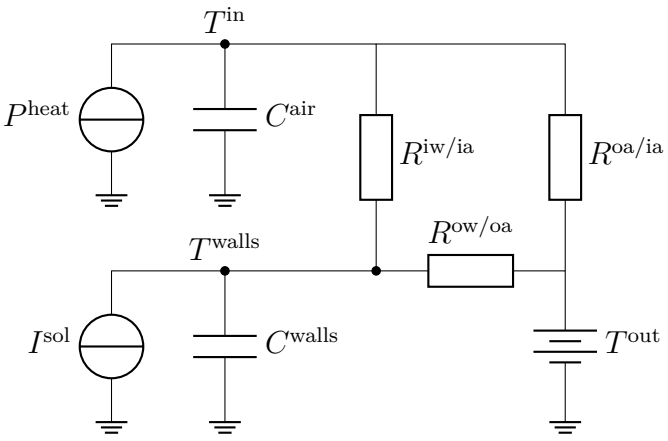


Figure 6.7 – Thermic Model 3R-2C of a house.

The electric circuit corresponds to a Linear Time-Invariant Continuous Time (LTICT)

Table 6.1 – Model Parameters

Symbol	Meaning
C_i^{air}	Heat capacity of inside air
C_i^{walls}	Heat capacity of external walls
$R_i^{\text{iw/ia}}$	Resist. between inside air and inside walls
$R_i^{\text{ow/oa}}$	Resist. between outside air and outside walls
$R_i^{\text{oa/ia}}$	Resist. between inside and out. air (from windows)
P^{heat}	Convector's heating power
I^{sol}	Sun's heating Power (by irradiance)
T^{in}	Mean temperature of Inside Air
T^{walls}	Mean temperature of Inside Walls

Table 6.2 – Parameters for each agent

Element	I	II	III	IV	Unit
C^{walls}	8	7	9	6	10^4J/K
C^{air}	5.0	4.0	4.5	4.7	10^4J/K
$R^{\text{oa/ia}}$	5	6	4	5	10^{-3}K/W
$R^{\text{iw/ia}}$	2.5	2.3	2.0	2.2	10^{-4}K/W
$R^{\text{ow/oa}}$	0.5	1.0	0.8	0.9	10^{-4}K/W

system, which can be represented in the state-space model by

$$\begin{array}{rcl} \dot{\boldsymbol{x}}_i(t) & = & A_{ci}\boldsymbol{x}_i(t) + B_{ci}\boldsymbol{u}_i(t) + D_{ci}\boldsymbol{d}_i(t) \\ \boldsymbol{y}_i(t) & = & C_{ci}\boldsymbol{x}_i(t) \end{array}, \tag{6.50}$$

where

$$A_{\text{c}_i} = \begin{bmatrix} -\frac{1}{C_i^{\text{walls}}R_i^{\text{oa/ia}}} - \frac{1}{C_i^{\text{walls}}R_i^{\text{iw/ia}}} & \frac{1}{C_i^{\text{walls}}R_i^{\text{iw/ia}}} \\ \frac{1}{C_i^{\text{air}}R_i^{\text{iw/ia}}} & -\frac{1}{C_i^{\text{air}}R^{\text{ow/oa}}o_i} - \frac{1}{C_i^{\text{air}}R_i^{\text{iw/ia}}} \end{bmatrix} \quad B_{\text{c}_i} = \begin{bmatrix} \frac{1}{C_i^{\text{air}}} \\ 0 \end{bmatrix}^{\text{T}}$$

$$D_{\text{c}_i} = \begin{bmatrix} 0 & \frac{1}{C_i^{\text{air}}R_i^{\text{ow/oa}}} \\ \frac{1}{C_i^{\text{air}}} & \frac{1}{C_i^{\text{air}}R^{\text{ow/oa}}} \end{bmatrix} \qquad C_{\text{c}_i} = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{6.51}$$

where $\boldsymbol{x}_i(t) = \left[T_i^{\text{in}}(t)\ T_i^{\text{walls}}(t)\right]^{\text{T}}$, $\boldsymbol{u}_i(t) = P_i^{\text{heat}}$, $\boldsymbol{d}_i(t) = \left[I^{\text{sol}}(t)\ T^{\text{out}}(t)\right]^{\text{T}}$.

Their heating inputs $\boldsymbol{u}_i(t)$ are constraint by the maximum heat power available by the provider, i.e.,

$$\sum_{i\in\mathcal{M}} \left[\boldsymbol{u}_i(t)\right] \preceq \boldsymbol{u}_{\max} \tag{6.52}$$

with $\boldsymbol{u}_{\max} = 40\text{kW}$.

Remark 6.12

For our examples, we ignore the disturbances caused by the outside temperature and the sun, i.e., $\boldsymbol{d}_i(t) = \mathbf{0}$.

We want to control the system using MPC, so we discretize the system using a zero-order holder and sampling time of $T_s = 0.25$h, resulting in the 3-tuples (A_i, B_i, C_i).

6.5.2 Applying RPdMPC-DS

First, to use the RPdMPC-DS, we assume the heating system is deprived, in other words, given an initial state $\boldsymbol{x}_i[0]$, references $\boldsymbol{w}_i[k]$ cannot be achieved due to input constraints.

For example, we suppose they have initial states $\boldsymbol{x}_{\mathrm{I}}[0] = [18.3 \quad 3.]^{\mathrm{T}}$, $\boldsymbol{x}_{\mathrm{II}}[0] = [19.6 \quad 5.9]^{\mathrm{T}}$, $\boldsymbol{x}_{\mathrm{III}}[0] = [18.4 \quad 5.3]^{\mathrm{T}}$, and $\boldsymbol{x}_{\mathrm{IV}}[0] = [17.4 \quad 5.3]^{\mathrm{T}}$, and references $w_{\mathrm{I}}[0] = 20$, $w_{\mathrm{II}}[0] = 20$, $w_{\mathrm{III}}[0] = 20$, and $w_{\mathrm{IV}}[0] = 20$. Using gains $Q_i = 10I$, and $R_i = I_2$, and choosing to control the system using a prediction horizon $N = 4$, we can structure the dMPC problem similar to the one in (A.29) and decompose the system using the usual primal decomposition, with local problems

$$\begin{aligned} \underset{\boldsymbol{U}_i[k]}{\text{minimize}} \quad & J_i = \tfrac{1}{2} \left\| \boldsymbol{U}_i[k] \right\|_{H_i}^2 + \boldsymbol{f}_i[k]^T \boldsymbol{U}_i[k] \\ \text{subject to} \quad & \bar{\Gamma}_i \boldsymbol{U}_i[k] = \boldsymbol{\theta}_i[k] : \boldsymbol{\lambda}_i[k] \\ & \boldsymbol{U}_i[k] \geq \mathbf{0} \end{aligned} \tag{6.53}$$

and update equation (5.15).

Results

We test the algorithm by simulating the system over a period of 5 hours, i.e., $k \in \{0, 5/T_s\}$, in some different scenarios:

1. Nominal behavior (denoted as N)
2. Agent I attacks system controlled by standard dMPC (denoted as S)
3. Agent I attacks system controlled by RPdMPC-DS (denoted as C)

The attack chosen for scenarios 2 and 3, are

$$T_I = \begin{cases} \left[\begin{smallmatrix} 4 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 4 \end{smallmatrix}\right], & \text{if } k > 5 \\ I_c, & \text{otherwise} \end{cases}. \tag{6.54}$$

In Fig. 6.8, we compare the temperature in house I ($\boldsymbol{y}_I[k]$) with its reference $\boldsymbol{w}_I[k]$, and then the detection variable $E_I(k)$ with the threshold $\epsilon_P = 10^{-4}$.

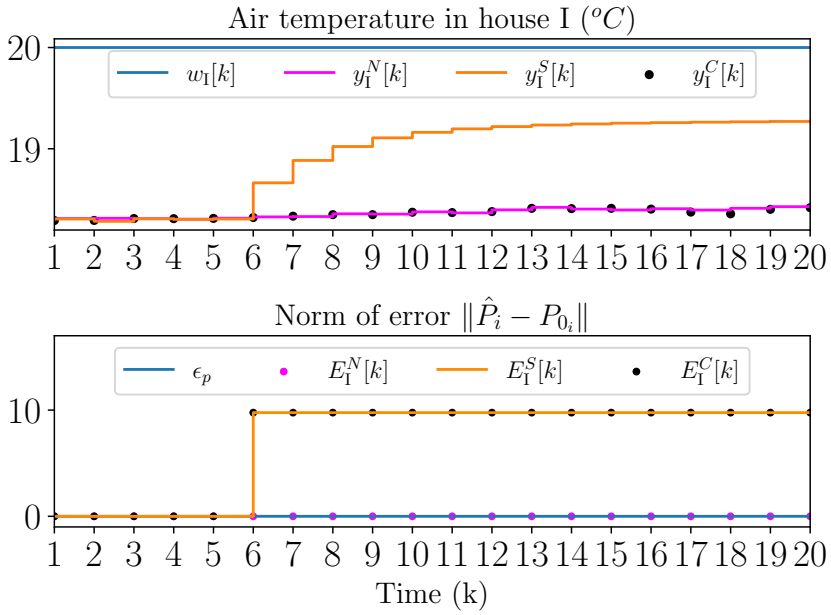


Figure 6.8 – Temperature in house I and the variable $E_I(k)$ for different scenarios.

As expected, the decision variable lies under the threshold $\epsilon_P = 10^{-4}$ in the nominal scenario. But for scenarios S and C, where the agent attacks the system, the detection variable surpasses the threshold, indicating the change of behavior of agent I. In scenario S, we can observe the tracking error $w_I - y_I$ reducing when the attack is activated, suggesting that the attacker drives the negotiation to get more resources. On the other hand, the other agents receive fewer resources, as we can see the tracking error increase in Fig. 6.9.

When the RPdMPC-DS is activated in scenario C, the temperatures recover behavior close to the nominal one, as we can see in Fig. 6.8 and 6.9.

We can also compare the accumulated local and global objectives J_i^{acc} and J^{acc} for the scenarios N, C, and S in Tab. 6.3. As expected, when no secure dMPC is applied, and the system is attacked, the objective function of the attackers decreases, while the others increase, together with the global objective J^{acc}. However, we can see for the scenarios where the RPdMPC-DS is activated, the objectives are close to the original.

Table 6.3 – Objective functions J_i (% error)

Agent	Scenario N	Scenario S	Scenario C
I	299.5	190.8 (−36.3)	301.0 (0.0)
II	192.4	234.1 (21.7)	191.4 (−0.5)
III	305.9	359.1 (17.4)	305.9 (−0.0)
IV	297.5	349.9 (17.6)	297.2 (−0.1)
Global	1095.3	1133.9 (3.5)	1095.5 (0.0)

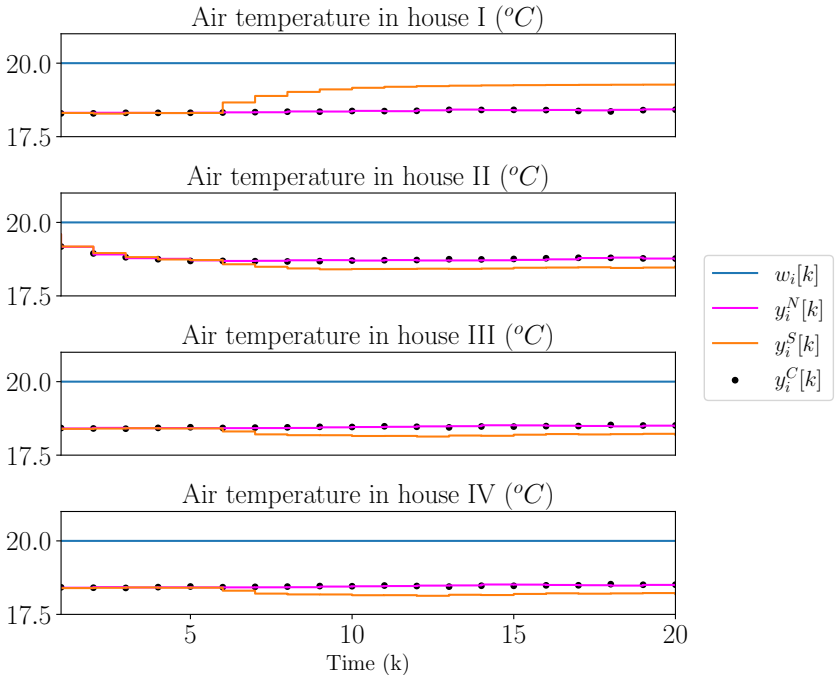


Figure 6.9 – Air temperature in all houses for different scenarios.

6.6 Conclusion

In this chapter, we analyzed the FDI attack proposed for a specific case where the controlled system is under scarcity. From the analytical solutions for the local optimization problems, we could assess more directly how the attack influences the optimal solution of the system and the dynamics of the primal decomposition.

The analysis gave us some insight into how to detect the attacks and mitigate given consequences of the attacks. And exploiting some assumptions, we proposed a secure dMPC algorithm. We assess its functioning through the simulation of a DHN under different scenarios, with and without attacks, and using the proposed algorithm.

In the next chapter, we try to relax some of the assumptions used in this chapter. As we will see, by alleviating one assumption, the problem to solve becomes exponentially more complex.

Chapter 7

Resilient Primal Decomposition-based distributed Model Predictive Control using Artificial Scarcity

What you thought
was way too much
is not enough

The Factory Gates
Kaiser Chiefs

In this chapter, we relax some assumptions to make the problem to be solved more generic. We compare the primal decomposition under nominal behavior and under the proposed attack model and the exponential increase of complexity.

Under some assumptions, we use an identification tool to circumvent the complexity of the system, allowing us to adapt the detection and mitigation just presented.

Contents

7.1 Relaxing the problem 106
7.1.1 Impact on local problems' solution 107
7.1.2 Impact on negotiation equations 112
7.2 Adapting detection and mitigation methods 118
7.2.1 Mitigation and artificial scarcity 118
7.2.2 Detection 119
7.2.3 Multiple Parameter Estimation 120
7.2.4 Complete safe dMPC under artificial scarcity 125
7.3 Numerical experiment 125
7.3.1 Applying RPdMPC-AS 126
7.4 Conclusion 129

7.1 Relaxing the problem

We begin again with the monolithic MPC equivalent problem (2.17). We reproduce yet another time the problem, but with a twist, now the problem also has a decomposable constraint set $\mathcal{U}$:

$$\begin{aligned} \underset{\boldsymbol{U}[k]}{\text{minimize}} \quad & \tfrac{1}{2}\left\|\boldsymbol{U}[k]\right\|_H^2 + \boldsymbol{f}[k]^T\boldsymbol{U}[k] \\ \text{subject to} \quad & \bar{\Gamma}\boldsymbol{U}[k] \preceq \boldsymbol{U}_{\max} \\ & \boldsymbol{U}[k] \in \mathcal{U} \end{aligned} \quad . \tag{7.1}$$

We decompose the system using the primal decomposition, and the local problems also have constraint sets, denoted $\mathcal{U}_i$:

$$\bar{J}_i^\star(\boldsymbol{\theta}_i[k])[k] = \begin{aligned} \underset{\boldsymbol{U}_i[k]}{\text{minimize}} \quad & \tfrac{1}{2}\left\|\boldsymbol{U}_i[k]\right\|_{H_i}^2 + \boldsymbol{f}_i[k]^T\boldsymbol{U}_i[k] \\ \text{subject to} \quad & \bar{\Gamma}_i\boldsymbol{U}_i[k] \preceq \boldsymbol{\theta}_i[k] : \boldsymbol{\lambda}_i[k] \\ & \boldsymbol{U}_i[k] \in \mathcal{U}_i \end{aligned} \quad , \tag{7.2}$$

and we use the projected subgradient method (5.9) (reproduced for convenience) to solve the main problem:

$$\boldsymbol{\theta}[k]^{(p+1)} = \text{Proj}^{\mathcal{S}}(\boldsymbol{\theta}[k]^{(p)} + \rho^{(p)}\boldsymbol{\lambda}[k]^{(p)}), \tag{5.9}$$

but with $\mathcal{S} = \{\boldsymbol{\theta}[k] \mid I_c^M\boldsymbol{\theta}[k] \preceq \boldsymbol{U}_{\max}\}$.

Remark 7.1

As seen in [Boy+15], if we have separable constraints sets, such as the $\mathcal{U}_i$ in our problem, we can modify the local objectives J_i so

$$J_i(\boldsymbol{\theta}_i^\star) = \infty, \text{ if } \boldsymbol{U}_i^\star \notin \text{dom}\, J_i.$$

Since this change modifies the objective functions, it also modifies the corresponding gradients and $\boldsymbol{\lambda}_i$ as consequence. So, this change guides the coordinator to choose $\boldsymbol{\theta}_i$, which respect dom J_i.

In this chapter, we still assume the original constraint (2.7) to have at most as many rows as columns and form cones. However, we relax one assumption made in §6.1.1. The systems are not necessarily deprived anymore, i.e., the unconstrained solutions $\mathring{\boldsymbol{U}}^\star = -H^{-1}\boldsymbol{f}[k]$ and $\mathring{\boldsymbol{U}}_i^\star[k] = -H_i^{-1}\boldsymbol{f}_i[k]$ may or may not respect the constraints.

As we will see in the next section, despite the relaxation of this assumption, making the problem more generic causes the solution to the local problems to be exponentially more complex.

7.1.1 Impact on local problems' solution

We can repeat the analysis made in the last chapter to see how the change in assumption influences the solutions to the problem.

Although now we have QP problems with inequality constraints, we still can use the same method to find an analytical solution.

We similarly define a function $g_i : \mathbb{R}^{\#\boldsymbol{U}_i[k]} \times \mathbb{R}^c$:

$$g_i(\boldsymbol{U}_i[k], \boldsymbol{\theta}_i[k]) = \bar{\Gamma}_i \boldsymbol{U}_i[k] - \boldsymbol{\theta}_i[k], \tag{7.3}$$

and use it to put the problem in standard form for inequality constraint QP programs:

$$\begin{array}{ll} \underset{\boldsymbol{U}_i[k]}{\text{minimize}} & J_i(\boldsymbol{U}_i[k], \boldsymbol{\theta}_i[k]) \\ \text{subject to} & g_i(\boldsymbol{U}_i[k], \boldsymbol{\theta}_i[k]) \preceq 0 : \boldsymbol{\lambda}_i[k] \end{array}, \tag{7.4}$$

with variables $\boldsymbol{\lambda}_i[k]$ associated to the constraints.

Remark 7.2

Usually, we make the distinction between the dual variables associated with the inequality and equality constraints by using different symbols. However, as in the problems, we have only one kind, we use the same symbol, and by context, the reader can distinguish to which kind the symbol corresponds.

We can again define the *Lagrangian* function $\mathscr{L} : \mathbb{R}^{\#\boldsymbol{U}_i[k]} \times \mathbb{R}^c \times \mathbb{R}^c \to \mathbb{R}$ as

$$\mathscr{L}(\boldsymbol{U}_i[k], \boldsymbol{\lambda}_i[k], \boldsymbol{\theta}_i[k]) = J_i(\boldsymbol{U}_i[k], \boldsymbol{\theta}_i[k]) + \boldsymbol{\lambda}_i[k]^T g_i(\boldsymbol{U}_i[k], \boldsymbol{\theta}_i[k]), \tag{7.5}$$

which embeds the inequality constraint into the objective function using $\boldsymbol{\lambda}_i[k]$, in this case, the lagrangian multiplier for *inequality*. Again, since there is only one type of lagrangian multiplier, we will call it simply the dual variable.

Remark 7.3

The complementary slackness still holds, $\boldsymbol{\lambda}_i^\star[k]^T g_i(\boldsymbol{U}_i^\star[k], \boldsymbol{\theta}_i[k])$ will always be zero. If any element of $g_i(\boldsymbol{U}_i^\star[k], \boldsymbol{\theta}_i[k])$ is smaller than zero, the corresponding element on the dual variable $\boldsymbol{\lambda}_i^\star[k]$ will be zero, indicating the constraint is inactive.

We can define the dual function $\mathscr{D} : \mathbb{R}^c \times \mathbb{R}^c \to \mathbb{R}$ also calculated by solving

$$\mathscr{D}(\boldsymbol{\lambda}_i[k], \boldsymbol{\theta}_i[k]) = \inf_{\boldsymbol{U}_i[k] \in \mathcal{F}} \mathscr{L}(\boldsymbol{U}_i[k], \boldsymbol{\lambda}_i[k], \boldsymbol{\theta}_i[k]), \tag{7.6}$$

but now where $\mathcal{F} = \operatorname{dom} g_i = \bigcap_{i=1}^{c} \mathcal{H}_i$, with $\mathcal{H}_i$ being the halfspaces $\mathcal{H}_i = \{\boldsymbol{x} \mid \bar{\Gamma}_{(i,\star)}\boldsymbol{x} \leqslant \boldsymbol{U}_{\max(i,\star)}\}$. Using the first order KKT optimality condition, we have

$$\nabla_{\boldsymbol{U}_i[k]} \mathscr{L}(\boldsymbol{U}_i[k], \boldsymbol{\lambda}_i[k], \boldsymbol{\theta}_i[k]) = 0 \tag{7.7}$$

yielding as well

$$\boldsymbol{U}_i[k] = -H_i^{-1}\boldsymbol{f}_i[k] + H_i^{-1}\bar{\Gamma}_i^T\boldsymbol{\lambda}_i[k], \tag{7.8}$$

which we know is

$$\boldsymbol{U}_i[k] = \mathring{\boldsymbol{U}}_i^{\star}[k] - H_i^{-1}\bar{\Gamma}_i^T\boldsymbol{\lambda}_i[k]. \tag{7.9}$$

Substituting it in $\mathscr{L}(\boldsymbol{U}_i[k], \boldsymbol{\lambda}_i[k], \boldsymbol{\theta}_i[k])$ results

$$\mathscr{D}(\boldsymbol{\lambda}_i[k], \boldsymbol{\theta}_i[k]) = -\frac{1}{2}\boldsymbol{\lambda}_i[k]^T\bar{\Gamma}_i H_i^{-1}\bar{\Gamma}_i{}^T\boldsymbol{\lambda}_i[k] - \boldsymbol{\lambda}_i[k]^T\bar{\Gamma}_i H_i^{-1}\boldsymbol{f}[k] - \boldsymbol{\lambda}_i[k]^T\boldsymbol{\theta}_i[k], \tag{7.10}$$

which is also concave.

Then, similarly, we can find $\boldsymbol{\lambda}_i{}^{\star}[k]$ by solving an optimization problem. However, since we are using inequalities, we can only have $\boldsymbol{\lambda}_i[k]$ positive [BV04]:

$$\boldsymbol{\lambda}_i{}^{\star}[k] = \begin{array}{ll} \underset{\boldsymbol{\lambda}_i[k]}{\operatorname{argmax}} & \mathscr{D}(\boldsymbol{\lambda}_i[k], \boldsymbol{\theta}_i[k]) \\ \text{subject to} & \boldsymbol{\lambda}_i[k] \geq \boldsymbol{0} \end{array}. \tag{7.11}$$

As we know, by complementary slackness, the value of the elements of $\boldsymbol{\lambda}_i[k]$ being 0 (or different of 0) will depend on the status of the corresponding constraint for the unconstrained solution $\mathring{\boldsymbol{U}}_i^{\star}[k]$. So the complete value of $\boldsymbol{\lambda}_i[k]$ will depend on the permutation of the status of the inequality constraint. Depending on the form of the constraints, if we have n_{ineq} constraints, we may have potentially $2^{n_{\text{ineq}}}$ permutations.

Remark 7.4

Here we say potentially because not necessarily all combinations are possible. In §7.1.1, we discuss briefly about the number of permutations and show some cases where the number of permutations is not equal to $2^{n_{\text{ineq}}}$.

To solve (7.11), we can use the fact that the status of the constraints depends on the value of $\boldsymbol{\theta}_i[k]$. This way, we can notice that the set of constraints divides the space of $\boldsymbol{\theta}_i[k]$ into multiple regions denoted $\mathcal{R}^n_{\boldsymbol{\lambda}_i}$, depending if the constraints are active or not for $\mathring{\boldsymbol{U}}_i^{\star}[k]$.

The regions $\mathcal{R}^n_{\boldsymbol{\lambda}_i}$ are polytopes defined as $\mathcal{R}^n_{\boldsymbol{\lambda}_i} = \{\boldsymbol{x} \mid G^n[k]\boldsymbol{x} \leq b^n[k]\}$ where the hyperplanes may vary with time k. As remarked in [Bem+02], the polytopes are non-overlapping, i.e., $\mathcal{R}^i_{\boldsymbol{\lambda}_i} \cap \mathcal{R}^j_{\boldsymbol{\lambda}_i} = \varnothing, \forall i \neq j$. This way, the regions form a partition of the $\boldsymbol{\theta}_i$ space, i.e., given $\boldsymbol{\theta}_i[k] \in \mathcal{Y} \subseteq \mathbb{R}^c$, $\bigcup_{i\in\{0:n_{\text{ineq}}-1\}} \mathcal{R}^i = \mathcal{Y}$.

Remark 7.5

Determining the exact polytopes/halfspaces is not in the scope of this work. But can be carried out by the methods in [LB19, §4.1.3.2].

Imagine an example where we have 2 constraints with associated $\boldsymbol{\lambda}_{i(1)}$ and $\boldsymbol{\lambda}_{i(2)}$, the $\boldsymbol{\theta}_i$ solution space is partitioned like in Fig. 7.1 (indices $[k]$ suppressed for brevity).

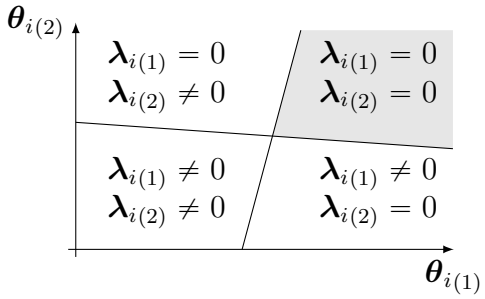


Figure 7.1 – Two constraints partitioning $\boldsymbol{\theta}_i$ solution space.

For any $\boldsymbol{\theta}_i$ inside the grey area, the solution to (7.4) will be unconstrained, i.e., the constraints will be inactive. Thus, for this case, by definition, the value of $\boldsymbol{\lambda}_i = \mathbf{0}$. For all other cases, where we know at least one constraint in (7.4) is active, we can remove the constraint in (7.11) for the element of $\boldsymbol{\lambda}_i$ corresponding to the active constraint. For instance, if all constraints in (7.4) are active, we can remove all constraints of (7.11), which will result in problem (6.17), whose solution we known already:

$$\boldsymbol{\lambda}_i[k] = -P_i\boldsymbol{\theta}_i[k] - \boldsymbol{s}_i[k], \tag{6.19}$$

By removing a set of constraints, we can solve a similar problem but with a reduced number of elements in $\boldsymbol{\theta}_i$ and $\boldsymbol{\lambda}_i$. For example, let us suppose the first constraint in (7.4) is inactive. We make $\boldsymbol{\lambda}_{i(1)} = 0$ and set it aside, we remove the corresponding lines of $\bar{\Gamma}_i$ and $\boldsymbol{\theta}_i$, and then we solve for all other elements of $\boldsymbol{\lambda}_i$, removing the remaining constraints in (7.11). This results in

$$\boldsymbol{\lambda}_{i(2:\text{end})}[k] = -\overset{2:c}{P}_i\boldsymbol{\theta}_{i(2:\text{end})}[k] - \overset{2:c}{\boldsymbol{s}}_i[k], \tag{7.12}$$

where $\overset{2:c}{P}_i = (\bar{\Gamma}_{i(2:\text{end},\star)}H_i^{-1}(\bar{\Gamma}_{i(2:\text{end},\star)})^T)^{-1}$ and $\overset{2:c}{\boldsymbol{s}}_i[k] = \overset{2:c}{P}_i\bar{\Gamma}_{i(2:\text{end},\star)}H_i^{-1}\boldsymbol{f}_i[k]$.

If we do this for all partitions, i.e., all possible permutations of active and inactive constraints, we will have as many equations similar to (7.12) as permutations. This result is the base for calculating the solution of the original problem (7.4) as shown in [Bem+02; AB09], which is the base for explicit MPC.

However, the equations have different sizes of elements (as many elements as active

constraints). To make all equations the same-sized, we recover the elements of $\boldsymbol{\lambda}_i$ set to zero and set aside. If we take the same example, Eq. (7.12) would become

$$\boldsymbol{\lambda}_i[k] = \begin{bmatrix} 0 \\ \boldsymbol{\lambda}_{i(2:\text{end})}[k] \end{bmatrix} = - \begin{bmatrix} 0 & 0 \\ 0 & \overset{2:c}{P_i} \end{bmatrix} \boldsymbol{\theta}_i[k] - \begin{bmatrix} 0 \\ \overset{2:c}{\boldsymbol{s}_i}[k] \end{bmatrix} \tag{7.13}$$

To facilitate the notation, let us use a binary representation such that if we have n_{ineq} constraints, we use n_{ineq} binary digits to represent their status. And we will mark the digits 1 if the constraint is inactive and 0 if active, indicating which constraints were removed or kept.

If we retake the 2-constraints example in Fig. 7.1, $00_{(2)}$ would represent both constraints active (bottom left quadrant), $01_{(2)}$ and $10_{(2)}$ represents one of the constraints are active (bottom right and top left quadrants, respectively) and $11_{(2)}$ where both constraints are inactive (grey area in top right quadrant). We can then use base 10 to represent such partitions. This way (7.13) can be written as

$$\boldsymbol{\lambda}_i[k] = -P_i^{(2^{n_{\text{ineq}}-1})} \boldsymbol{\theta}_i[k] - \boldsymbol{s}_i[k]^{(2^{n_{\text{ineq}}-1})}, \tag{7.14}$$

with index $(2^{n_{\text{ineq}}-1})$ indicating that only the first constraint is inactive.

Then, we can write the complete solution of (7.11) as a PWA function

$$\boldsymbol{\lambda}_i[k] = \begin{cases} -P_i^{(0)} \boldsymbol{\theta}_i[k] - \boldsymbol{s}_i^{(0)}[k], & \text{if } \boldsymbol{\theta}_i[k] \in \mathcal{R}^0_{\boldsymbol{\lambda}_i} \\ \quad \vdots & \quad \vdots \\ -P_i^{(2^{n_{\text{ineq}}}-1)} \boldsymbol{\theta}_i[k] - \boldsymbol{s}_i^{(2^{n_{\text{ineq}}}-1)}[k], & \text{if } \boldsymbol{\theta}_i[k] \in \mathcal{R}^{2^{n_{\text{ineq}}}-1}_{\boldsymbol{\lambda}_i} \end{cases}. \tag{7.15}$$

Here we can begin to regard the change caused by the relaxation, whereas we had only one equation relating $\boldsymbol{\lambda}_i$ to $\boldsymbol{\theta}_i$ when the system was deprived, now we have potentially $2^{n_{\text{ineq}}}$ different equations.

Remark 7.6

As one may expect, by definition, the $P_i^{(n)}$ and $\boldsymbol{s}_i^{(n)}[k]$ are increasingly sparse. When n increases, more constraints are inactive and more blocks are equal to zero, culminating in $P_i^{(2^{n_{\text{ineq}}}-1)} = 0_c$ and $\boldsymbol{s}_i^{(2^{n_{\text{ineq}}}-1)}[k] = \mathbf{0}_c$.

Considerations about status of constraints

It is important to observe that there are some sets of constraints in which the permutation is not necessarily $2^{n_{\text{ineq}}}$.

For example, in some cases, there is no partition where all constraints are inactive, i.e., the intersection of halfspaces is empty (see Fig. 7.2). Problems with such constraints are infeasible, thus we ignore them in this work.

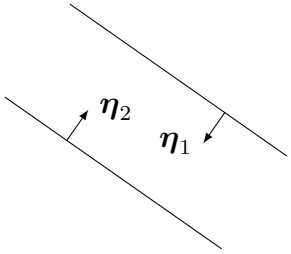


Figure 7.2 – Set of constraints with no intersection.

In other cases, there is no combination where all constraints are active. We give some different examples. For instance, for constraints with normals $\boldsymbol{\eta}_i$, if the angles of adjacent halfspaces $\langle_{\boldsymbol{\eta}_i}^{\boldsymbol{\eta}_j}$ is 180^o and the intersection is not nil (see Fig. 7.3), at least one of the constraints will always be inactive.

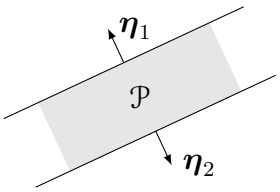


Figure 7.3 – Two constraints with $\langle_{\boldsymbol{\eta}_1}^{\boldsymbol{\eta}_2} = 180^o$.

Yet another example is when the constraints form a polyhedron. We present a minimal example of polyhedron in Fig. 7.4, a triangle in $\mathbb{R}^2$ formed by 3 inequality constraints. In this case, there will always be at least 1 inactive constraint.

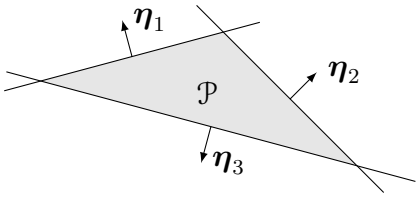


Figure 7.4 – A 3-sided polyhedron.

Since we suppose there are at most the number of constraints as dimensions, it is not possible to create polyhedra. [1] Although, for superior dimensions, prisms may also be taken into account.

1. For $\mathbb{R}^n$, the simplex has $n+1$ faces

7.1.2 Impact on negotiation equations

We can try to apply the same logic used in §6.1.4, by substituting (7.15) into (5.9) and vice-versa, and compare with the results in §6.1.4. In this attempt, we can fully contemplate how the relaxation of the assumptions increases the problem's complexity.

The first difficulty is already on the projected subgradient method, which due to the set $\mathcal{S}$ being the intersection of halfplanes, does not have a straightforward solution as in the equality constraint case (5.15). The solution is found by applying the definition of the Euclidean projection, which is by itself a QP problem

$$\text{Proj}^{\mathcal{X}}(\boldsymbol{x}_0) = \underset{\boldsymbol{x}}{\text{argmin}} \begin{Bmatrix} \underset{\boldsymbol{x}}{\text{minimize}} & \|\boldsymbol{x} - \boldsymbol{x}_0\|^2 \\ \text{subject to} & \boldsymbol{x} \in \mathcal{X} \end{Bmatrix}, \tag{7.16}$$

which translates to the problem

$$\text{Proj}^{\mathcal{S}}(\boldsymbol{\theta}[k]^{(p)} + \rho^{(p)}\boldsymbol{\lambda}[k]^{(p)}) = \underset{\boldsymbol{x}}{\text{argmin}} \begin{Bmatrix} \underset{\boldsymbol{U}[k]}{\text{minimize}} & \left\|\boldsymbol{x} - \boldsymbol{\theta}[k]^{(p)} + \rho^{(p)}\boldsymbol{\lambda}[k]^{(p)}\right\| \\ \text{subject to} & I_c^M \boldsymbol{x} \preceq \boldsymbol{U}_{\max} : \boldsymbol{\mu} \end{Bmatrix}, \tag{7.17}$$

with corresponding dual variable $\boldsymbol{\mu}$. For notation sake we use $\boldsymbol{x}_0 = \boldsymbol{\theta}[k]^{(p)} + \rho^{(p)}\boldsymbol{\lambda}[k]^{(p)}$.

The problem is solved similarly to how we made in §7.1.2. Skipping some steps (calculating lagrangians and optimizing for the dual variables), we will find that the solution of $\boldsymbol{\mu}$ has the same form that (7.15). However, we need to change $\bar{\Gamma}_i$ for I_c^M, H_i for I, and $\boldsymbol{f}_i$ for $-(\boldsymbol{\theta}[k]^{(p)} + \rho^{(p)}\boldsymbol{\lambda}[k]^{(p)})$. And the partitions are defined by the active and inactive constraints in $I_c^M \boldsymbol{x}_0 \preceq \boldsymbol{U}_{\max}$.

Substituting the solution of the dual variables into the lagrangian, and making the gradient vanish, will also result in a PWA function with at most 2^c different modes ($\#\, {I_c^M}_{(*,1)} = c$):

$$\boldsymbol{\theta}[k]^{(p+1)} = \begin{cases} \boldsymbol{x}_0 + I_c^{M(0)}\left[-P_{\boldsymbol{\mu}}^{(0)}\boldsymbol{U}_{max} + \boldsymbol{s}_{\boldsymbol{\mu}}^{(0)}[k]\right], & \text{if } \boldsymbol{x}_0 \in \mathcal{R}_{\boldsymbol{\mu}}^0 \\ \quad\vdots & \quad\vdots \\ \boldsymbol{x}_0, & \text{if } \boldsymbol{x}_0 \in \mathcal{R}_{\boldsymbol{\mu}}^{2^c-1} \end{cases}, \tag{7.18}$$

similarly with non-overlapping regions $\mathcal{R}_{\boldsymbol{\mu}}^n = \{\boldsymbol{x} \mid G_{\boldsymbol{\mu}}^n[k]\boldsymbol{x} \preceq \boldsymbol{U}_{\max}^n[k]\}$ [2] which partition the space of $\boldsymbol{x}_0$. The non-zero elements (coded by 0 digits in the binary representation of (n)) of $P_{\boldsymbol{\mu}}^{(n)}$ are $\left(I_c^{M(n)} {I_c^{M(n)}}^{\mathrm{T}}\right)^{-1}$, and those of $\boldsymbol{s}_{\boldsymbol{\mu}}^{(n)}[k]$ are $\left(I_c^{M(n)} {I_c^{M(n)}}^{\mathrm{T}}\right)^{-1} I_c^{M(n)}\boldsymbol{x}_0$, $I_c^{M(n)}$ is constructed by keeping/removing the constraints as coded by index (n).

Observe that when all constraints are active ($n = 0$), the solution is the same as in the equality case (5.14) ($\#\, I_c^M = c \times Mc$). And when all constraints are inactive ($n = 2^c - 1$), the solution is the same as the unconstrained solution, since $\boldsymbol{\mu} = \boldsymbol{0}$ (see Remark 7.6).

2. Also not defined in this work

So, we can conclude that the next value $\boldsymbol{\theta}_i[k]^{(p+1)}$ depends on where $\boldsymbol{\theta}[k]^{(p)} + \rho^{(p)}\boldsymbol{\lambda}[k]^{(p)}$ will lie, giving at most 2^c possibilities. In turn, $\boldsymbol{\theta}[k]^{(p)} + \rho^{(p)}\boldsymbol{\lambda}[k]^{(p)}$ depends on the values of each $\boldsymbol{\lambda}_i[k]$, which conversely can have $2^{n_{\text{ineq}}}$ different modes each.

As in §6.1.4, when substituting the equations, one in another gave us expressions of DT systems, similarly here, if we substitute the results for $\boldsymbol{\lambda}_i[k]$ in (7.18) we will have a switched DT systems. One with constant (in the same mode) input for $\boldsymbol{\theta}_i[k]$, and conversely, a switched homogenous system for the dynamics of $\boldsymbol{\lambda}_i[k]$.

If we develop the calculations, we can estimate an upper bound on the number of total different permutations $\bar{\epsilon}$, which totals to

$$\bar{\epsilon} = \overbrace{2^c}^{\text{regions in projection}} \times \underbrace{2^{n_{\text{ineq}}} \times \cdots \times 2^{n_{\text{ineq}}}}_{M \times \text{regions in each } \boldsymbol{\lambda}_i} = 2^{c + M n_{\text{ineq}}}. \tag{7.19}$$

Remark 7.7

Probably some of those permutations may not be possible, but still, we have exponential growth in the number of modes of the switched systems. The exact calculation for the regions is not the scope of this work.

The case for deprived systems seen in the last chapter settles the solution of the projection to the case (0), where $\boldsymbol{\theta}[k]^{(p)} + \rho^{(p)}\boldsymbol{\lambda}[k]^{(p)}$ is projected onto the intersection of all halfspaces. And due to another assumption (that $\bar{\Gamma}_i \mathring{\boldsymbol{U}}_i^{\star}[k] \succ \boldsymbol{\theta}_i[k], \forall i \in \mathcal{M}, \forall k$), we fix all $\boldsymbol{\lambda}_i$ to the first solution in (7.15). This consecutive choice of assumptions reduces the total number of switching modes to exactly one, presented by equations (6.22) and (6.28), reproduced here.

$$\boldsymbol{\lambda}^{(p+1)} = \mathcal{A}_\lambda \boldsymbol{\lambda}^{(p)}, \tag{6.22}$$

$$\boldsymbol{\theta}^{(p+1)} = \mathcal{A}_\theta \boldsymbol{\theta}^{(p)} + \boldsymbol{B}_\theta[k] \tag{6.28}$$

If we concatenate the binary representations of the modes of the projection and of each local problem, we will have a number with $c + M n_{\text{ineq}}$ digits. And thus, the solutions in (6.22) and (6.28) can be seen as a specific case of the following solutions for the 0-th modes:

$$\boldsymbol{\lambda}[k]^{(p+1)} = \begin{cases} \mathcal{A}_\lambda^0 \boldsymbol{\lambda}[k]^{(p)}, & \text{if } \boldsymbol{\theta} \in \mathcal{R}_{\boldsymbol{\lambda}}^0 \\ \vdots & \vdots \\ \mathcal{A}_\lambda^{N_{\boldsymbol{\lambda}}} \boldsymbol{\lambda}[k]^{(p)}, & \text{if } \boldsymbol{\theta} \in \mathcal{R}_{\boldsymbol{\lambda}}^{N_{\boldsymbol{\lambda}}} \end{cases} \tag{7.20}$$

$$\boldsymbol{\theta}[k]^{(p+1)} = \begin{cases} \mathcal{A}_\theta^0 \boldsymbol{\theta}[k]^{(p)} + \boldsymbol{B}_\theta^0[k], & \text{if } \boldsymbol{\theta} \in \mathcal{R}_{\boldsymbol{\theta}}^0 \\ \vdots & \vdots \\ \mathcal{A}_\theta^{N_{\boldsymbol{\theta}}} \boldsymbol{\theta}[k]^{(p)} + \boldsymbol{B}_\theta^{N_{\boldsymbol{\theta}}}[k], & \text{if } \boldsymbol{\theta} \in \mathcal{R}_{\boldsymbol{\theta}}^{N_{\boldsymbol{\theta}}} \end{cases} \tag{7.21}$$

where $N_{\boldsymbol{\lambda}} = N_{\boldsymbol{\theta}} = 2^{c+Mn_{\text{ineq}}} - 1$. The regions described by $\mathcal{R}_{\boldsymbol{\lambda}}^n = \{\boldsymbol{x} \mid G_{\boldsymbol{\lambda}}^n[k]\boldsymbol{\theta}[k]^{(p)} \preceq \boldsymbol{b}_{\boldsymbol{\lambda}}^n[k]\}$, and $\mathcal{R}_{\boldsymbol{\theta}}^n = \{\boldsymbol{x} \mid G_{\boldsymbol{\theta}}^n[k]\boldsymbol{\theta}[k]^{(p)} \preceq \boldsymbol{b}_{\boldsymbol{\theta}}^n[k]\}$ coincide, and are formed by the superposition of the regions $\mathcal{R}_{\boldsymbol{\lambda}_i}^n$ in (7.18) and $\mathcal{R}_{\boldsymbol{\mu}}^n$ in (7.15). We highlight again that not necessarily all permutations are possible (some regions are empty).

Observe that in some cases, it is straightforward to define the values of the matrices in (7.20) and (7.21). The 0-th case, as mentioned, is the same as we found in the last chapter, giving $\mathcal{A}_{\boldsymbol{\lambda}}^0 = \mathcal{A}_{\boldsymbol{\lambda}}$, $\mathcal{A}_{\boldsymbol{\theta}}^0 = \mathcal{A}_{\boldsymbol{\theta}}$, and $\boldsymbol{B}_{\boldsymbol{\theta}}^0[k] = \boldsymbol{B}_{\boldsymbol{\theta}}[k]$. Similarly for the $N_{\boldsymbol{\theta}}$-th case where all the constraints of all systems are inactive (implying all $\boldsymbol{\lambda}_i = \mathbf{0}$), and the projection of $\boldsymbol{\theta}[k]^{(p)} + \rho^{(p)}\boldsymbol{\lambda}[k]^{(p)}$ is itself, yielding $\mathcal{A}_{\boldsymbol{\lambda}}^{N_{\boldsymbol{\lambda}}} = 0_c$, $\mathcal{A}_{\boldsymbol{\theta}}^{N_{\boldsymbol{\theta}}} = I_c$, and $\boldsymbol{B}_{\boldsymbol{\theta}}^{N_{\boldsymbol{\theta}}}[k] = \mathbf{0}$. The $N_{\boldsymbol{\theta}}$-th case is trivial since it means the optimal value was already found, $\boldsymbol{\theta}[k]^{(p+1)} = \boldsymbol{\theta}[k]^{(p)}$.

Applying the attack (5.17), using cheating matrices T_i we can have simular results

$$\tilde{\boldsymbol{\lambda}}[k]^{(p+1)} = \begin{cases} \tilde{\mathcal{A}}_\lambda^0 \tilde{\boldsymbol{\lambda}}[k]^{(p)}, & \text{if } \boldsymbol{x}_0 \in \mathcal{R}_{\boldsymbol{\lambda}}^0 \\ \vdots & \vdots \\ \tilde{\mathcal{A}}_\lambda^{N_{\boldsymbol{\lambda}}} \tilde{\boldsymbol{\lambda}}[k]^{(p)}, & \text{if } \boldsymbol{x}_0 \in \mathcal{R}_{\boldsymbol{\lambda}}^{N_{\boldsymbol{\lambda}}} \end{cases} \tag{7.22}$$

$$\boldsymbol{\theta}[k]^{(p+1)} = \begin{cases} \tilde{\mathcal{A}}_\theta^0 \boldsymbol{\theta}[k]^{(p)} + \tilde{\boldsymbol{B}}_\theta^0[k], & \text{if } \boldsymbol{\theta} \in \mathcal{R}_{\boldsymbol{\theta}}^0 \\ \vdots & \vdots \\ \tilde{\mathcal{A}}_\theta^{N_{\boldsymbol{\theta}}} \boldsymbol{\theta}[k]^{(p)} + \tilde{\boldsymbol{B}}_\theta^{N_{\boldsymbol{\theta}}}[k], & \text{if } \boldsymbol{\theta} \in \mathcal{R}_{\boldsymbol{\theta}}^{N_{\boldsymbol{\theta}}} \end{cases} \tag{7.23}$$

One can observe that since we multiply $\boldsymbol{\lambda}_i$ by $T_i[k]$, we do not change the partition of the spaces. We can similarly see that $\tilde{\mathcal{A}}_{\boldsymbol{\lambda}}^0 = \tilde{\mathcal{A}}_{\boldsymbol{\lambda}}$, $\tilde{\mathcal{A}}_{\boldsymbol{\theta}}^0 = \tilde{\mathcal{A}}_{\boldsymbol{\theta}}$, $\tilde{\boldsymbol{B}}_{\boldsymbol{\theta}}^0[k] = \tilde{\boldsymbol{B}}_{\boldsymbol{\theta}}[k]$, and all analysis made in the last chapter also holds for this mode.

Remark 7.8

One interesting fact we can observe is that for cases where the corresponding $\boldsymbol{\lambda}_i$ is already $\mathbf{0}$, the multiplication of T_i will not affect its value. A satisfied agent, i.e., an agent whose local problem has the same solution as its unconstrained version, has no motive to get more resources since the allocated resources will satisfy its needs. So, in the most extreme case, specifically the $N_{\boldsymbol{\theta}}$-th mode, we see that even if all agents cheat at the same time, there is no impact on the system since they are all satisfied, $\boldsymbol{\lambda}_i = \mathbf{0}\ \forall i \in \mathcal{M}$, resulting in $\tilde{\mathcal{A}}_{\boldsymbol{\lambda}}^{N_{\boldsymbol{\lambda}}} = \mathcal{A}_{\boldsymbol{\lambda}}^{N_{\boldsymbol{\lambda}}}$, $\tilde{\mathcal{A}}_{\boldsymbol{\theta}}^{N_{\boldsymbol{\theta}}} = \mathcal{A}_{\boldsymbol{\theta}}^{N_{\boldsymbol{\theta}}}$, and $\tilde{\boldsymbol{B}}_{\boldsymbol{\theta}}^{N_{\boldsymbol{\theta}}}[k] = \boldsymbol{B}_{\boldsymbol{\theta}}^{N_{\boldsymbol{\theta}}}[k]$.

We could analyze stability in a case-by-case fashion, seeing what constraints are active/inactive for which agent and how the cheating matrices influence the stability. But due to the extension of the analysis (exponential number of cases), we skip it for brevity's sake and will use a single example, different from the trivial solution, to illustrate how it could be done.

Denoting as the zone (or region) n, or the n-zone, the non-overlapping polytopes defined by $\mathcal{R}^n = \{\boldsymbol{x} \mid G^n_{\boldsymbol{\theta}}[k]\boldsymbol{x} \preceq \boldsymbol{b}^n_{\boldsymbol{\theta}}[k]\}$, we retake our 2-constraints example in Fig. 7.1. Since there are 2 constraints, we set for example $c = 1$ and $N = 2$, then we have at most 4 different modes on the solution for $\tilde{\boldsymbol{\lambda}}_i$:

$$\tilde{\boldsymbol{\lambda}}_i[k] = \begin{cases} -\widetilde{P}_i^{(0)}\boldsymbol{\theta}_i[k] - \widetilde{\boldsymbol{s}}_i^{(0)}[k], & \text{if } \boldsymbol{\theta}_i[k] \in \mathcal{R}^0 \\ -\widetilde{P}_i^{(1)}\boldsymbol{\theta}_i[k] - \widetilde{\boldsymbol{s}}_i^{(1)}[k], & \text{if } \boldsymbol{\theta}_i[k] \in \mathcal{R}^1 \\ -\widetilde{P}_i^{(2)}\boldsymbol{\theta}_i[k] - \widetilde{\boldsymbol{s}}_i^{(2)}[k], & \text{if } \boldsymbol{\theta}_i[k] \in \mathcal{R}^2 \\ -\widetilde{P}_i^{(3)}\boldsymbol{\theta}_i[k] - \widetilde{\boldsymbol{s}}_i^{(3)}[k], & \text{if } \boldsymbol{\theta}_i[k] \in \mathcal{R}^3 \end{cases} \tag{7.24}$$

where

$$\widetilde{P}_i^{(0)} = T_i(\bar{\Gamma}_{i(\star,\star)}H_i^{-1}(\bar{\Gamma}_{i(\star,\star)})^T)^{-1} = T_i(\bar{\Gamma}_i H_i^{-1}\bar{\Gamma}_i{}^T)^{-1} \tag{7.25}$$

$$\widetilde{P}_i^{(1)} = T_i\begin{bmatrix}(\bar{\Gamma}_{i(1,*)}H_i^{-1}(\bar{\Gamma}_{i(1,*)})^T)^{-1} & 0 \\ 0 & 0\end{bmatrix} = \begin{bmatrix}T_{i(1,1)}(\bar{\Gamma}_{i(1,*)}H_i^{-1}(\bar{\Gamma}_{i(1,*)})^T)^{-1} & 0 \\ T_{i(2,1)}(\bar{\Gamma}_{i(1,*)}H_i^{-1}(\bar{\Gamma}_{i(1,*)})^T)^{-1} & 0\end{bmatrix} \tag{7.26}$$

$$\widetilde{P}_i^{(2)} = T_i\begin{bmatrix}0 & 0 \\ 0 & (\bar{\Gamma}_{i(2,*)}H_i^{-1}(\bar{\Gamma}_{i(2,*)})^T)^{-1}\end{bmatrix} = \begin{bmatrix}0 & T_{i(1,2)}(\bar{\Gamma}_{i(2,*)}H_i^{-1}(\bar{\Gamma}_{i(2,*)})^T)^{-1} \\ 0 & T_{i(2,2)}(\bar{\Gamma}_{i(2,*)}H_i^{-1}(\bar{\Gamma}_{i(2,*)})^T)^{-1}\end{bmatrix} \tag{7.27}$$

$$\widetilde{P}_i^{(3)} = T_i\begin{bmatrix}0 & 0 \\ 0 & 0\end{bmatrix} = \begin{bmatrix}0 & 0 \\ 0 & 0\end{bmatrix} \tag{7.28}$$

$$\widetilde{\boldsymbol{s}}_i^{(0)}[k] = T_i\boldsymbol{s}_i^{(0)}[k] = T_i(\bar{\Gamma}_i H_i^{-1}\bar{\Gamma}_i{}^T)^{-1}\bar{\Gamma}_i H_i^{-1}\boldsymbol{f}_i[k] \tag{7.29}$$

$$\widetilde{\boldsymbol{s}}_i^{(1)}[k] = T_i\boldsymbol{s}_i^{(1)}[k] = \begin{bmatrix}T_{i(1,1)}(\bar{\Gamma}_{i(1,*)}H_i^{-1}(\bar{\Gamma}_{i(1,*)})^T)^{-1}\bar{\Gamma}_{i(1,*)}H_i^{-1}\boldsymbol{f}_i[k] \\ T_{i(2,1)}(\bar{\Gamma}_{i(1,*)}H_i^{-1}(\bar{\Gamma}_{i(1,*)})^T)^{-1}\bar{\Gamma}_{i(1,*)}H_i^{-1}\boldsymbol{f}_i[k]\end{bmatrix} \tag{7.30}$$

$$\widetilde{\boldsymbol{s}}_i^{(2)}[k] = T_i\boldsymbol{s}_i^{(2)}[k] = \begin{bmatrix}T_{i(1,2)}(\bar{\Gamma}_{i(2,*)}H_i^{-1}(\bar{\Gamma}_{i(2,*)})^T)^{-1}\bar{\Gamma}_{i(2,*)}H_i^{-1}\boldsymbol{f}_i[k] \\ T_{i(2,2)}(\bar{\Gamma}_{i(2,*)}H_i^{-1}(\bar{\Gamma}_{i(2,*)})^T)^{-1}\bar{\Gamma}_{i(2,*)}H_i^{-1}\boldsymbol{f}_i[k]\end{bmatrix} \tag{7.31}$$

$$\widetilde{\boldsymbol{s}}_i^{(3)}[k] = T_i\boldsymbol{s}_i^{(3)}[k] = \begin{bmatrix}0 \\ 0\end{bmatrix} \tag{7.32}$$

with

$$\boldsymbol{s}_i^{(0)}[k] = (\bar{\Gamma}_{i(\star,\star)} H_i^{-1} (\bar{\Gamma}_{i(\star,\star)})^T)^{-1} \bar{\Gamma}_{i(*,*)} H_i^{-1} \boldsymbol{f}_i[k] \tag{7.33}$$

$$\boldsymbol{s}_i^{(1)}[k] = \begin{bmatrix} (\bar{\Gamma}_{i(1,*)} H_i^{-1} (\bar{\Gamma}_{i(1,*)})^T)^{-1} \bar{\Gamma}_{i(1,*)} H_i^{-1} \boldsymbol{f}_i[k] \\ 0 \end{bmatrix} \tag{7.34}$$

$$\boldsymbol{s}_i^{(2)}[k] = \begin{bmatrix} 0 \\ (\bar{\Gamma}_{i(2,*)} H_i^{-1} (\bar{\Gamma}_{i(2,*)})^T)^{-1} \bar{\Gamma}_{i(2,*)} H_i^{-1} \boldsymbol{f}_i[k] \end{bmatrix} \tag{7.35}$$

$$\boldsymbol{s}_i^{(3)}[k] = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{7.36}$$

One can observe that for zones 1 and 2, when the agent attacks, the non-zero elements of $P_i^{(n)}$ and $\boldsymbol{s}_i^{(n)}[k]$ influence the other elements using the columns of T_i equivalent to the active constraints. For example, in zone 1, $T_{i(1,1)}$ and $T_{i(2,1)}$ are used, and for zone 2, $T_{i(1,2)}$ and $T_{i(2,2)}$.

Remark 7.9

Observe that if T_i is diagonal, there is no injection of "active elements" into any "inactive elements".

So, instead of always propagating to the negotiation all the matrix T_i, as in the case of the last chapter, only the columns related to the active constraints are propagated.

If we have $M = 2$ subsystems, one in the 1-zone and the other in the 2-zone (let us call them agents I and II), and we suppose the solution of the projection is in its 0-zone, i.e., $G_{\boldsymbol{\mu}}^0[k](\boldsymbol{\theta}[k]^{(p)} + \rho^{(p)}\boldsymbol{\lambda}[k]^{(p)}) \preceq \boldsymbol{U}_{\max}^0[k]$, we code the permutation as

$$\underbrace{00}_{\text{projection's zone}} \overbrace{01}^{\text{agent I's zone}} \underbrace{10}_{\text{agent II's zone}} = 000110_{(2)} = 6_{(10)}. \tag{7.37}$$

Applying the projection, we will have the same solution (5.14), reprised here

$$\boldsymbol{\theta}^{(p+1)} = \boldsymbol{\theta}^{(p)} + \rho^{(p)}\boldsymbol{\lambda}^{(p)} + I_c^{M\mathrm{T}}/(I_c^M I_c^{M\mathrm{T}}) \left(I_c^M(\boldsymbol{\theta}^{(p)} - \rho^{(p)}\boldsymbol{\lambda}^{(p)}) - \boldsymbol{U}_{\max} \right), \tag{5.14}$$

which can be rewritten for each $\boldsymbol{\theta}_i$ as

$$\boldsymbol{\theta}_i^{(p+1)} = \boldsymbol{\theta}_i^{(p)} + \frac{1}{M} \left(\sum_{j \in \mathcal{M}} \boldsymbol{\theta}_j^{(p)} - \boldsymbol{U}_{\max} \right) + \rho^{(p)} \left(\boldsymbol{\lambda}^{(p)} - \frac{1}{M} \sum_{j \in \mathcal{M}} \boldsymbol{\lambda}_j^{(p)} \right) \tag{7.38}$$

and substituting the $\boldsymbol{\lambda}_i$ by $\tilde{\boldsymbol{\lambda}}_i$:

$$\boldsymbol{\theta}_i^{(p+1)} = \boldsymbol{\theta}_i^{(p)} + \frac{1}{M}\left(\sum_{j\in\mathcal{M}} \boldsymbol{\theta}_j^{(p)} - \boldsymbol{U}_{\max}\right) + \rho^{(p)}\left(-\widetilde{P}_i^{\,(n(i))}\boldsymbol{\theta}_i^{(p)} - \widetilde{\boldsymbol{s}}_i^{\,(n(i))}[k]\right) - \frac{\rho^{(p)}}{M}\sum_{j\in\mathcal{M}}\left(-\widetilde{P}_j^{\,(n(j))}\boldsymbol{\theta}_j^{(p)} - \widetilde{\boldsymbol{s}}_j^{\,(n(j))}[k]\right) \tag{7.39}$$

where $n(i)$ indicates the coding of the zone of agent i.

Grouping the terms we have

$$\boldsymbol{\theta}_i^{(p+1)} = \boldsymbol{\theta}_i^{(p)} + \rho^{(p)}\left(-\widetilde{P}_i^{\,(n(i))}\boldsymbol{\theta}_i^{(p)} + \frac{1}{M}\sum_{j\in\mathcal{M}}\widetilde{P}_j^{\,(n(j))}\boldsymbol{\theta}_j^{(p)}\right) + \frac{1}{M}\left(\sum_{j\in\mathcal{M}}\boldsymbol{\theta}_j^{(p)}\right) \tag{7.40}$$

$$+\rho^{(p)}\left(-\boldsymbol{s}_i[k] + \frac{1}{M}\sum_{j\in\mathcal{M}}\boldsymbol{s}_j[k]\right) - \frac{\boldsymbol{U}_{\max}}{M} \tag{7.41}$$

resulting in

$$\boldsymbol{\theta}^{(p+1)} = \tilde{\mathcal{A}}_\theta^{(6)}\boldsymbol{\theta}^{(p)} + \tilde{\boldsymbol{B}}_\theta^{(6)}[k] \tag{7.42}$$

where

$$\tilde{\mathcal{A}}_\theta^{(6)} = \begin{bmatrix} 2I - \frac{1}{2}\rho^{(p)}\widetilde{P_I}^{(1)} & I + \frac{1}{2}\rho^{(p)}\widetilde{P_{II}}^{(2)} \\ I + \frac{1}{2}\rho^{(p)}\widetilde{P_I}^{(1)} & 2I - \frac{1}{2}\rho^{(p)}\widetilde{P_{II}}^{(2)} \end{bmatrix} \tag{7.43}$$

$$\tilde{\boldsymbol{B}}_\theta^{(6)}[k] = \begin{bmatrix} -\frac{1}{2}\rho^{(p)}\widetilde{\boldsymbol{s}_I}^{(1)}[k] + \frac{1}{2}\rho^{(p)}\widetilde{\boldsymbol{s}_{II}}^{(2)}[k] - \frac{\boldsymbol{U}_{\max}}{2} \\ \frac{1}{2}\rho^{(p)}\widetilde{\boldsymbol{s}_I}^{(1)}[k] - \frac{1}{2}\rho^{(p)}\widetilde{\boldsymbol{s}_{II}}^{(2)}[k] - \frac{\boldsymbol{U}_{\max}}{2} \end{bmatrix} \tag{7.44}$$

And we can see that, as expected, the columns of T_i are propagated to $\mathcal{A}_{\boldsymbol{\theta}}$ and may change its dynamics. We could do the same for the dynamics of $\boldsymbol{\lambda}_i$, but we let this for the reader.

For the overall system dynamics, we may use any tools for analysis of hybrid/ switched systems [BBM17].

Remark 7.10

Observe that since the matrices $\mathcal{A}_{\boldsymbol{\theta}}^n$ and $\mathcal{A}_{\boldsymbol{\lambda}}^n$ may be sparse, the eigenvalue analysis to indicate the stability of the mode is accomplished by keeping only the non-zero corresponding blocks.

As we see, the relaxation of the assumption makes the problem exponentially more complicated. In the next section, we will discuss ways to adapt the detection and mitigation strategies to overcome the complexity presented.

7.2 Adapting detection and mitigation methods

We propose to use a similar strategy to the one used in the last chapter. However, some difficulties arise. We do not have one tuple $(P_i, \boldsymbol{s}_i[k])$ for each subsystem, but we have 2^{Nc} different tuples. Here are proposals on how to counter the difficulties.

7.2.1 Mitigation and artificial scarcity

We begin by the end with the proposal for the mitigation because the proposal follows a straightforward logic and impacts the proposal for the detection, simplifying its solution.

We still want to mitigate the effects of the attack by reconstructing $\boldsymbol{\lambda}_i$. So, if we assume we can somehow estimate the inverse of the cheating matrix T_i, which we still assume is invertible (See (A.57)), we can easily try to reconstruct $\boldsymbol{\lambda}_i$ by applying

$$\overset{\text{rec}}{\boldsymbol{\lambda}}_i = \widehat{T_i[k]^{-1}}\tilde{\boldsymbol{\lambda}}_i. \tag{7.45}$$

However, estimating $\widehat{T_i[k]^{-1}}$ is a bit more difficult than before. We cannot use (6.47) anymore (reprised for clarity) since we do not have P_i.

$$\widehat{T_i[k]^{-1}} = \bar{P}_i\hat{\tilde{P}}_i[k]^{-1}. \tag{6.47}$$

If we estimate any $P_i^{(n)}$, we cannot necessarily invert it. As seen in §7.1.2, almost all $P_i^{(n)}$ are sparse with rows and columns equal to zero, thus singular. The only one that is invertible is $P_i^{(0)}$. So if we can estimate $\widetilde{P}_i^{(0)}$ and we have a nominal $\bar{P}_i^{(0)}$ (probably estimated the same way during an attack-free trial), we can estimate $\widehat{T_i[k]^{-1}}$ similarly:

$$\widehat{T_i[k]^{-1}} = \bar{P}_i^{(0)}\hat{\tilde{P}}_i^{(0)}[k]^{-1}. \tag{7.46}$$

So, we assume that $P_i^{(0)}$ exists and that it is possible to estimate it.

The necessary condition for the existence of the 0-th region (0-zone) is that we can choose $\boldsymbol{\theta}_i$ such that the constraints $g_i(\boldsymbol{U}_i[k], \boldsymbol{\theta}_i[k])$ are active for the unconstrained solution $\mathring{\boldsymbol{U}}_i^\star[k]$. Since we can only choose $\boldsymbol{\theta}_i$ that respects $\operatorname{dom} J_i$, to assure feasibility, we assume the unconstrained solution $\mathring{\boldsymbol{U}}^\star$ and $\mathring{\boldsymbol{U}}_i^\star[k]$ lie inside the respective constraint sets

$$\mathring{\boldsymbol{U}}^\star \in \mathcal{U} \tag{7.47}$$

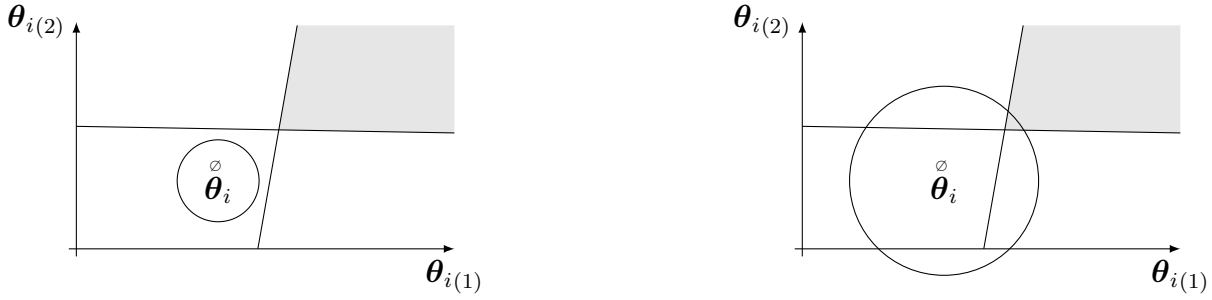
$$\mathring{\boldsymbol{U}}_i^\star[k] \in \mathcal{U}_i. \tag{7.48}$$

We call *artificial scarcity* the method of forcing the constraints to be active. And supposing there is a value $\overset{\varnothing}{\boldsymbol{\theta}}_i$ that activates all constraints, we call it the *artificial scarcity point*.

So, if we can generate points in a ball ($\mathcal{B}(\boldsymbol{x}_c, r) = \{\boldsymbol{x} \mid \|\boldsymbol{x} - \boldsymbol{x}_c\| \leqslant r\}$) around $\overset{\varnothing}{\boldsymbol{\theta}}_i$ (See

Fig. 7.5a), we can use the same method of supervision of communication of the coordinator and agents used in the last chapter to estimate $P_i^{(0)}$.

However, one complication arises when choosing the size of the ball around $\overset{\varnothing}{\boldsymbol{\theta}}_i$, since it may traverse one of the frontiers of the polytope and generate points corresponding to other modes (See Fig. 7.5b).



(a) Ball $\mathcal{B}(\overset{\varnothing}{\boldsymbol{\theta}}_i, r)$.

(b) Ball $\mathcal{B}(\overset{\varnothing}{\boldsymbol{\theta}}_i, r)$ traversing zones.

Figure 7.5 – Effects of different radii r when generating points close to $\overset{\varnothing}{\boldsymbol{\theta}}_i$.

Therefore, we need an estimation method that can also simultaneously classify the points, indicating to which zone they correspond. Then, we take only the parameters that interest us, that is $P_i^{(0)}$.

In §7.2.3, we will discuss estimation methods, and we describe the proposed method to solve this problem.

7.2.2 Detection

For the detection, we can use the same logic presented in the last chapter.

As we can observe in (7.15), the affine parts $P_i^{(n)}$ are time-invariant. So, if we estimate the affine parts $\widehat{\widetilde{P}}_i^{(n)}[k]$, we can detect an attack when there is any difference between the estimated and nominal values $\bar{P}_i^{(n)}$. We can compare the estimates with nominals from any non-zero mode [3]. However, since we are going to use $\widehat{\widetilde{P}}_i^{(0)}[k]$ for the mitigation part, we can also use it for the detection.

Analogously, we can set an arbitrary bound $\epsilon_{P_i^{(0)}}$, and define the error

$$E_i^{(0)}[k] = \left\| \widehat{\widetilde{P}}_i^{(0)}[k] - \bar{P}_i^{(0)} \right\|_F, \tag{7.49}$$

which associated with an detection function $\mathfrak{d}_i^{(0)}$:

$$\mathfrak{d}_i^{(0)} = \mathbb{1}_{\{E_i^{(0)}[k] \geqslant \epsilon_{P_i^{(0)}}\}}, \tag{7.50}$$

3. 2^{Nc}-zones have empty elements, which do not change when the cheating matrices are applied (§7.1.2).

detects an attack when the corresponding bound is disrespected, i.e., $\mathfrak{d}_i^{(0)} = 1$.

7.2.3 Multiple Parameter Estimation

The problem of estimating affine parameters on a system with a known number of different modes is called in [LB19] as a *PWA regression with fixed modes*. The problem is formalized in an optimization problem which we adapt for our case. To find the n different parameters $\boldsymbol{\nu}_i$ of the PWA function with non-overlapping regions $\mathcal{R}^i$ such $\bigcup_{i=1}^{n} \mathcal{R}^i = \mathcal{R}$, we take a set of N inputs/output tuples $(B_k, \boldsymbol{\gamma}_k)|_{k\in\{1:N\}} \subset \mathbb{R}^{\#B} \times \mathbb{R}^{\#\gamma}$ and solve

$$\underset{\boldsymbol{\nu}\in\mathbb{R}^{n\,\#B_{(1,*)}}}{\text{minimize}} \; \frac{1}{N} \sum_{k=1}^{N} \sum_{j=1}^{n} \mathbb{1}_{g(B_k)=j} \left\| \|\boldsymbol{\gamma}_k - B_k \boldsymbol{\nu}_j\|_1 \right\|. \tag{7.51}$$

where $g : \mathbb{R}^{\#B} \to \{1 : n\}$ is a linear classification function, which classifies the membership of its argument, returning the corresponding index of the region it is a member of. The definition of such a function can be seen in [LB19]. Observe that since the output of g is an integer ($\{1 : n\} \subset \mathbb{Z}$), the problem is a Mixed Integer Quadratic Programming (MIQP) program. In this work, we are not interested in determining the classification functions but just the parameters.

As commented in [LB19], since the program is a MIQP if we try to solve the problem by brute force (trying all the combinations), the complexity increases exponentially depending on the number of modes n and inputs N. So, some clever methods were created to exploit the problem structure, by successively solving different optimization problems for example. Some strategies are listed in [RBL04].

The most known examples of these strategies are the methods in the *partitional family*. The methods in this family solve partition and regression problems successively. Notable members of the family are the widely used k-means [Fer+01] and k-medoids [Bis06] methods.

In these methods, estimates of the parameters are used to classify the membership of all points, assigning a region to each input, and then, the points are used to solve different regression problems, updating the parameter estimates of their corresponding region.

Another similar method [Bis06], which can also be categorized in the same family, is the Expectation Maximization (EM) method [DLR77]. The EM is the method of choice in this chapter to estimate the parameters needed for the detection and mitigation phases.

We could have used any of the other methods cited, but we chose EM for it not making a hard assignment of the points, but instead a soft choice based on probabilities, what we will see gives us favorable properties on convergence.

This method is not commonly used by the control community, being more frequently used in the machine learning/data mining areas. However, an inspiring example was proposed for the estimation of switched systems in [NTK05].

In the next subsection, we describe the method more precisely.

Expectation Maximization

The objective of the EM algorithm is to find, from a set of observable data $\mathcal{B}$, estimators of a set of parameters $\mathcal{P}$ that maximize the log marginal likelihood of the observed data $\ln \mathbb{P}(\mathcal{B}; \mathcal{P})$. The models generally have latent variables (unobservable) in a set $\mathcal{U}$.

The problem is that maximizing $\ln \mathbb{P}(\mathcal{B}; \mathcal{P})$ does not have an analytical solution. So, the algorithm solves the optimization problem in an iterative manner.

Instead of estimating the $\mathcal{P}$ that maximizes $\ln \mathbb{P}(\mathcal{B}; \mathcal{P})$, we find estimates that maximize the expectation of the complete-data log-likelihood $\ln \mathbb{P}(\mathcal{B}, \mathcal{U}; \mathcal{P})$ w.r.t. the posterior probabilities $\mathbb{P}(\mathcal{U}|\mathcal{B}; \mathcal{P})$.

Though, as we need to have $\mathcal{P}$ to calculate the posterior probabilities, we calculate it using current estimates of $\mathcal{P}$ denoted $\mathcal{P}_{\text{cur}}$. And repeating the same strategy with the newfound estimate, we can approach the estimates which maximize the log marginal likelihood.

The successive steps of calculating the posterior probabilities (expectation) and maximizing the complete-data log-likelihood give the method the name Expectation Maximization. These steps provide a monotonic increase of the log-marginal likelihood at each iteration, yielding guaranteed convergence for a local maximum.

The iterative algorithm can be summarized in Algorithm 7.1.

Algorithm 7.1: Expectation Maximization

Initialize parameters $\mathcal{P}_{\text{new}}$
repeat
  $\mathcal{P}_{\text{cur}} \leftarrow \mathcal{P}_{\text{new}}$
  E step:
    Evaluate $\mathbb{P}(\mathcal{U}|\mathcal{B}; \mathcal{P}_{\text{cur}})$
  M step:
    Estimate $\mathcal{P}_{\text{new}}$:

$$\mathcal{P}_{\text{new}} = \underset{\mathcal{P}}{\operatorname{argmax}} \ \mathbb{E}_{\mathbb{P}(\mathcal{U}|\mathcal{B};\mathcal{P}_{\text{cur}})} \left[\ln \mathbb{P}(\mathcal{B}, \mathcal{U}; \mathcal{P})\right] \tag{7.52}$$

until $\mathcal{P}_{\text{cur}}$ *converges*

Remark 7.11

The method has guaranteed convergence for local maxima, which depend on the initialization of the estimates of $\mathcal{P}$ [BC15; GP21]. The choice of initial estimates is not in the scope of this work, the reader is referred to [KX03; BC15], where the authors analyze the choice for initialization.

Adapting the problem to use the method

Since the EM is a probabilistic method, we need to have a probabilistic regression model, which is not our case. The idea is to incorporate probabilistic behavior into our model (7.15).

As the method will be used for all agents at every negotiation, we drop the subscript i and the time dependency $[k]$, simplifying the notation.

First, we exchange our original deterministic variables $\boldsymbol{\theta}_i$ and $\boldsymbol{\lambda}_i$, for their stochastic counterparts denoted $\underline{\boldsymbol{\theta}_i}$ and $\underline{\boldsymbol{\lambda}_i}$.

If we watch O exchanges between an agent and the coordinator, we can observe the input and response variables, identified as $\underline{\boldsymbol{\theta}}_o$ and $\underline{\boldsymbol{\lambda}}_o$, $o \in \mathcal{O} = \{0 : O-1\}$. The input and response can be organized in $\underline{\Theta}, \underline{\Lambda} \in \mathbb{R}^{c \times O}$. The tuple $(\underline{\Theta}, \underline{\Lambda})$ is the observable data $\mathcal{B}$.

Since each $\underline{\boldsymbol{\theta}}_o$ belongs to a given region, we associate a variable $z \in \mathcal{Z} = \{0 : Z-1\}$ which indicates the number of a zone. We can then construct our regression model as

$$\underline{\boldsymbol{\lambda}}_o = \begin{cases} -\tilde{P}^0 \underline{\boldsymbol{\theta}}_o - \tilde{\boldsymbol{s}}^0, & \text{if in zone } 0 \\ \vdots & \vdots \\ -\tilde{P}^{Z-1} \underline{\boldsymbol{\theta}}_o - \tilde{\boldsymbol{s}}^Z, & \text{if in zone } Z-1 \end{cases}. \tag{7.53}$$

In [FS10], a somehow similar model is called *mixture of linear regressions*, but since we are using PWA functions, which have linear and constant terms, we will call the model *mixture of affine regressions*.

Since we don't know the partitions or the belonging of each observed point, we associate the index z of each observation $(\underline{\boldsymbol{\lambda}}_o, \underline{\boldsymbol{\theta}}_o)$ to a latent random variable $\underline{z}_o$. We can organize the $\underline{z}_o$ variables into $\underline{Z} \in \mathbb{R}^{1 \times O}$, which is our set of latent variables $\mathcal{U}$.

We suppose the latent variables $\underline{z}_o$ follow a categorical prior distribution, with associated probabilities $\Pi = \{\pi^z | z \in \mathcal{Z}\}$ such

$$\mathbb{P}(\underline{z}_o = z) = \pi^z \in [0, 1], \qquad \sum_{z \in \mathcal{Z}} \pi^z = 1.$$

We can also create a parameter set $\mathcal{P} = \{\mathcal{P}^z \mid z \in \mathcal{Z}\}$, with $\mathcal{P}^z = (\tilde{P}^z, \tilde{\boldsymbol{s}}^z, \pi^z)$.

Remark 7.12

We estimate π^z and $\tilde{\boldsymbol{s}}^z$ purely for updating the model and constraining the identification of $\tilde{P}^z$, since they are used neither for the detection nor for the mitigation phases.

As the coordinator controls the input $\boldsymbol{\theta}$, we consider a non-informative improper probability density function [Chr+10]

$$\mathbb{P}(\underline{\boldsymbol{\theta}}_o) \propto 1,$$

which means that we known almost surely the value of $\underline{\boldsymbol{\theta}}_o$.

Once defined the input and latent variables, we model the response $\underline{\lambda}_o$ as a multivariate normal random variable with probability density function

$$\mathbb{P}(\underline{\boldsymbol{\lambda}}_o|\underline{\boldsymbol{\theta}}_o, \underline{z}_o = z; \mathcal{P}^z) = \mathcal{N}(\underline{\boldsymbol{\lambda}}_o; f(\underline{\boldsymbol{\theta}}_o; \mathcal{P}^z), \Sigma^z), \tag{7.54}$$

where, following the mixture of affine regressions model (7.53), the mean vector is

$$f(\underline{\boldsymbol{\theta}}_o; (P, \boldsymbol{s}, \pi)) = -P\underline{\boldsymbol{\theta}}_o - \boldsymbol{s}, \tag{7.55}$$

The covariance matrices Σ^z tend to 0_c, approaching the original values of the original functions in (7.15).

An example of the representation of such function for a 1 dimension PWA function can be seen in Fig. 7.6, where darker points present more probability than lighter ones.

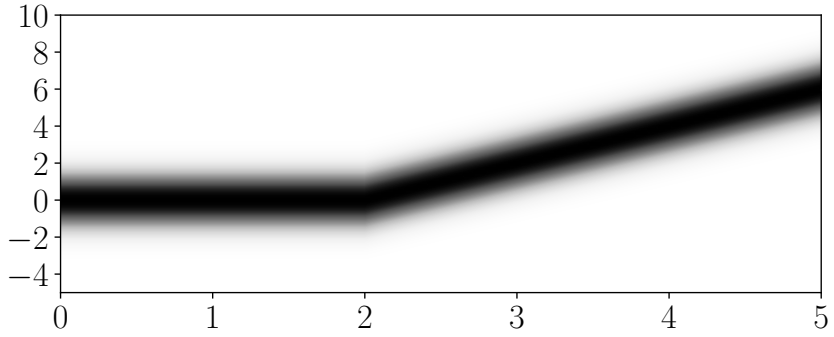


Figure 7.6 – Gaussian Mixture for a 1D PWA function with 2 modes.

The posterior probabilities $\zeta_{zo}(\mathcal{P}) = \mathbb{P}(\underline{z}_o = z|\underline{\boldsymbol{\lambda}}_o, \underline{\boldsymbol{\theta}}_o; \mathcal{P})$, also called *responsibilities*, are calculated as:

$$\zeta_{zo}(\mathcal{P}) = \frac{\pi_z \mathcal{N}(\underline{\boldsymbol{\lambda}}_o; f(\underline{\boldsymbol{\theta}}_o; \mathcal{P}^z), \Sigma^z)}{\sum_{j=1}^{Z} \pi_j \mathcal{N}(\underline{\boldsymbol{\lambda}}_o; f(\underline{\boldsymbol{\theta}}_o; \mathcal{P}^j), \Sigma^j)}. \tag{7.56}$$

We call them responsibilities because we calculate the probability of the zone z being responsible for generating observation o.

We can calculate the expectation of $\ln \mathbb{P}(\underline{\Theta}, \underline{\Lambda}, \underline{Z}; \mathcal{P})$ w.r..t $\zeta_{zo}(\mathcal{P}_{\text{cur}})$ [Bis06, Chapter 9] using

$$\mathbb{E}_{\zeta_{zo}(\mathcal{P}_{\text{cur}})}\left[\ln \mathbb{P}(\underline{\Theta}, \underline{\Lambda}, \underline{Z}; \mathcal{P})\right] = \sum_{o \in \mathcal{O}} \sum_{z \in \mathcal{Z}} \zeta_{zo}(\mathcal{P}_{\text{cur}}) \alpha_{zo}, \tag{7.57}$$

where $\alpha_{zo} = \ln \pi_z + \ln \mathcal{N}(\underline{\boldsymbol{\lambda}}_o; f(\underline{\boldsymbol{\theta}}_o; \mathcal{P}^z), \Sigma^z)$.

To facilitate the math, we introduce a variable $\boldsymbol{\phi}^z$ to embed the parameters, similar to

what we made for the RLS case (§6.2.1)

$$\boldsymbol{\phi}^z = \begin{bmatrix} \text{vec}(\tilde{P}^z)^T \\ \tilde{\boldsymbol{s}}^z \end{bmatrix}. \tag{7.58}$$

To find the optimal $\boldsymbol{\phi}^z$ for the problem in (7.52), we take the gradients of (A.107) with respect to vectors $\boldsymbol{\phi}^z$ and make them vanish.

Since the problem is multidimensional, some matrix operations are needed. After those operations, we can find a matricial solution yielding the optimal estimates $\boldsymbol{\phi}^z_{\text{new}}$:

$$\boldsymbol{\phi}^z_{\text{new}} = (\Xi^z \underline{\Omega})^\dagger \Xi^z \text{vec}(\underline{\Lambda}), \tag{7.59}$$

where $\underline{\Omega} = [(\Upsilon \underline{\Theta} \Delta) \circ Y; G]$, with matrices $\Upsilon = \mathbf{1}_c^T \otimes I_c$, $\Delta = I_O \otimes \mathbf{1}_c^T$, $Y = G \otimes \mathbf{1}_c$, $G = \mathbf{1}_O^T \otimes I_c$, and

$$\Xi^z = \text{diag}(\sqrt{\zeta(z_{z1}; \mathcal{P}_{\text{cur}})} I_c, \cdots, \sqrt{\zeta(z_{zO}; \mathcal{P}_{\text{cur}})} I_c).$$

Doing the same for π^z we get

$$\pi^z = \sum_{o \in \mathcal{O}} \frac{\zeta_{zo}(\mathcal{P}_{\text{cur}})}{O}.$$

One can observe that (7.59) is the solution of a weighted LS, with the responsibilities as weights, adjusting the contribution of all observations to the regression models.

There are some similarities to the K-planes (k-means) algorithm (see [BM00]), but EM is more compromising. Instead of affecting the observed data to a zone with 100% of certainty (*hard assignment*), EM uses each zone's responsibilities (*soft assignment*).

Other methods, such as stochastic Expectation Maximization (sEM) and Classification Expectation Maximization (CEM), make a hard choice based on the responsibilities. sEM chooses at random one of the regions depending on the responsibilities, while CEM chooses the region with greater responsibility.

When estimates $\boldsymbol{\phi}_z^{\text{new}}$ converge, we can retrieve the estimates $\tilde{P}^z$ and $\tilde{\boldsymbol{s}}^z$, and use them in our detection and mitigation scheme.

Remark 7.13

Observe that the parameters indices z do not necessarily converge to the parameters with the same number in (7.15) since it depends on the initialization of $\mathcal{P}$ and the observation set. For example, if we generate points inside one single region j, all Z regions would have the estimated parameter of the given region j.

Since the z indices do not necessarily correspond to the ones in (7.15), we need to recover the z that corresponds to the 0-zone. We take the observation o corresponding to

the $\overset{\varnothing}{\boldsymbol{\theta}}_i$, which we know belongs to the 0-zone, and we find its most probable region z with

$$\arg\max_z. \ \zeta_{zo}(\mathcal{P}).$$

The parameter with the given index corresponds to our $\widehat{\widetilde{P}}_i^{(0)}[k]$.

Possible issues

There are two main possible difficulties with methods of the partional family.

The first is the known slow convergence for some cases [CG92; Bis06; BC15], depending on the size of the problem. In [FS10], the authors compare the convergence of EM with other variants such sEM and CEM.

The other is the dependability of the solution to the initial estimate conditions, which was already commented on in Remark 7.11.

Another issue that we can encounter is the inversion of some variables when working with high orders (notably the covariance Σ). The order depends on the number of observations O and a number of zones Z, and some variables can come close to being singular.

One way to counteract this effect is to include it in the estimated parameters. Another way is to consider the covariance as a diagonal matrix with the same values for all entries [KX03]. Yet another way is to use the technique called *Simulated annealing* [CG92; OF10], where the covariances are initialized with arbitrarily significant values indicating the uncertainty of the parameters, and it is reduced at each iteration. In this work, we used the latter two techniques combined.

7.2.4 Complete safe dMPC under artificial scarcity

Putting together all parts defined in the last sections, we can come up with Algorithm 7.2, which resumes the RPdMPC-AS. Again we can see all the detection and conditional reconstruction of $\boldsymbol{\lambda}_i$ as a supervisor, as shown in the last chapter.

7.3 Numerical experiment

We exemplify the action of the RPdMPC-AS through an academic example. As a study case, we use the same example of DHN presented in §6.5.1, with slight modifications to respect the assumptions made in this chapter.

Algorithm 7.2: Resilient Primal Decomposition-based dMPC under artificial scarcity.

Coordinator initialization:
  Initialize k, p, a, b, $p_{\max}$ and ϵ
while $k \geqslant 0$ **do**
  Detection Phase:
    Initialize size of ball r
    for $o \in \mathcal{O}$ **do**
      Coordinator sends $\boldsymbol{\theta}_i^o \in \mathcal{B}(\overset{\varnothing}{\boldsymbol{\theta}}_i, r)$
      Subsystems solve (5.6a), and send $\tilde{\boldsymbol{\lambda}}_i^o$
    Coordinator estimates $\widehat{\widetilde{P}}_i^{(0)}[k]$ and $\widehat{\tilde{\boldsymbol{s}}}_i^{(0)}[k]$ with EM (Algorithm 7.1)
    Coordinator computes $\mathfrak{d}_i$ using (A.101)
  Negotiation Phase:
    Coordinator initializes $\boldsymbol{\theta}^{(p)}$ and sends to subsystems
    repeat
      Agents solve local sub-problems (5.6a) and send $\boldsymbol{\lambda}_i[k]^{(p)}$ to coordinator
      Coordinator updates allocation (5.9) using $\overset{\text{mod}}{\boldsymbol{\lambda}}_i$:
        $\boldsymbol{\lambda}_i^\star(\boldsymbol{\theta}^{(p)})$, if $\mathfrak{d}_i = 0$ and $\overset{\text{rec}}{\boldsymbol{\lambda}}_i$, if $\mathfrak{d}_i = 1$ (7.45)
      $p \leftarrow p + 1$
    until $\left[\left\|\boldsymbol{\theta}[k]^{(p)} - \boldsymbol{\theta}[k]^{(p-1)}\right\| \leqslant \epsilon\right] \vee [p \geqslant p_{\max}]$
    Agents apply on respective sub-systems the last calculated $\boldsymbol{u}_i^\star[0|k]$
    $k \leftarrow k + 1$

7.3.1 Applying RPdMPC-AS

The first assumption to use the RPdMPC-AS is that the local unconstrained solutions $\mathring{\boldsymbol{U}}_i^\star[k]$ respect the local constraints. As noted in §6.5.1, we do not have cooling apparatus. Thus the inputs are restricted to be positive, i.e., $\boldsymbol{u}_i[k] \geq \mathbf{0}$, so we need to force $\mathring{\boldsymbol{U}}_i^\star[k]$ to respect such constraint the $\mathring{\boldsymbol{U}}_i^\star[k] \geq \mathbf{0}$.

This is accomplished by changing the initial states $\boldsymbol{x}_i[k]$ and references $\boldsymbol{w}_i[k]$ such the output $\boldsymbol{y}_i[k]$ (air temperature inside the house) never surpasses the references, what would yield a solution with negative inputs.

Therefore, we change the initial states to be $\boldsymbol{x}_{\mathrm{I}}[0] = [17.\ \ 3.2]^{\mathrm{T}}$, $\boldsymbol{x}_{\mathrm{II}}[0] = [20.\ \ 5.3]^{\mathrm{T}}$, $\boldsymbol{x}_{\mathrm{III}}[0] = [15.\ \ 3.1]^{\mathrm{T}}$, and $\boldsymbol{x}_{\mathrm{IV}}[0] = [17.\ \ 5.7]^{\mathrm{T}}$, and the references as $w_{\mathrm{I}}[0] = 25.5$, $w_{\mathrm{II}}[0] = 24.0$, $w_{\mathrm{III}}[0] = 18.0$, and $w_{\mathrm{IV}}[0] = 20.4$. All other parameters are the same, which, when decomposing the MPC problem, we have

$$\begin{aligned} \underset{\boldsymbol{U}_i[k]}{\text{minimize}} \quad & J_i = \tfrac{1}{2}\left\|\boldsymbol{U}_i[k]\right\|_{H_i}^2 + \boldsymbol{f}_i[k]^T\boldsymbol{U}_i[k] \\ \text{subject to} \quad & \bar{\Gamma}_i\boldsymbol{U}_i[k] \succeq \boldsymbol{\theta}_i[k] : \boldsymbol{\lambda}_i[k] \\ & \boldsymbol{U}_i[k] \succeq \mathbf{0} \end{aligned} \tag{7.60}$$

and update equation (5.9).

Results

We test the algorithm using the same simulation period of 5.0 hours, i.e., $k \in \{0, 5.0/T_s\}$. The same scenarios are used:

1. Nominal behavior (denoted as N)
2. Agent I attacks system controlled by standard dMPC (denoted as S)
3. Agent I attacks system controlled by RPdMPC-AS (denoted as C)

The attack model used for scenarios 2 and 3, are

$$T_I = \begin{cases} \begin{bmatrix} 14.43288267 & 0. & 0. & 0. \\ 0. & 13.4590903 & 0. & 0. \\ 0. & 0. & 6.93065061 & 0. \\ 0. & 0. & 0. & 3.4447393 \end{bmatrix}, & \text{if } k > 10 \\ I_c, & \text{otherwise} \end{cases} \tag{7.61}$$

Observing Fig. 7.7, we can compare the air temperature in house I with its reference w_I. We can also see in the figure the values of the decision variable $E_I^{(0)}[k]$ with the threshold $\epsilon_{P_i^{(0)}}$. Notice how the reference w_I is not reached even in the nominal behavior (in magenta) due to initial states and power constraints, indicating that the unconstrained solution $\mathring{\boldsymbol{U}}_i^\star[k] \succeq \mathbf{0}$. As expected, the decision variable lies under the threshold $\epsilon_{P_i^{(0)}} = 10^{-4}$ with values of order $E_I^N(k) \approx 10^{-10}$ (no attack detected).

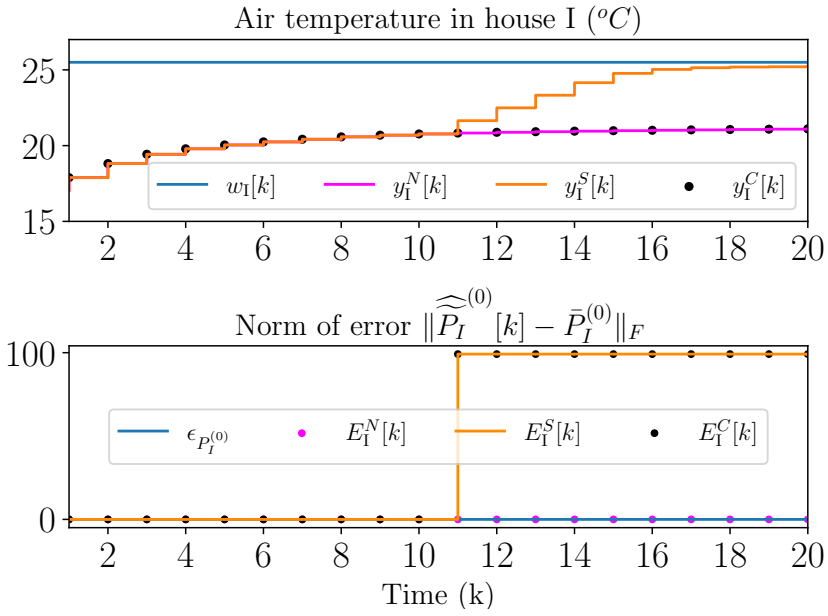


Figure 7.7 – Air temperature in house I and the decision variable $E_I[k]$ for different scenarios.

When agent I attacks the system (in orange), its tracking error $w_I - y_I$ decreases, and it almost reaches the reference. In Fig. 7.8, we see how the resources from house I were transferred from all other houses, which lost performance.

In this case, the detection variable surpasses the bound $\epsilon_{P_i^{(0)}}$, $E_I^S \approx 200$, indicating a change of behavior, a possible attack.

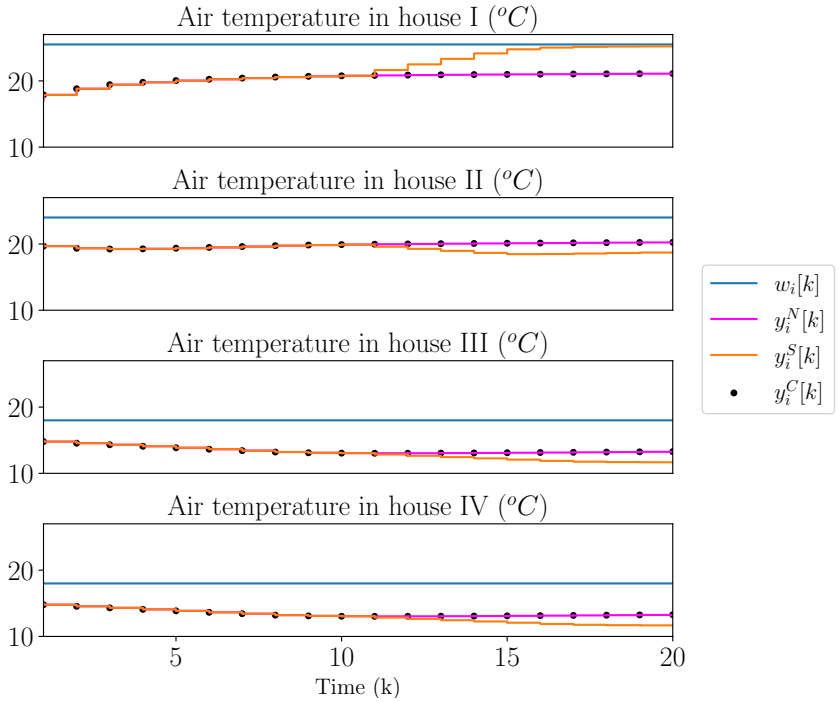


Figure 7.8 – Air temperature in all houses for different scenarios.

If the proposed resilient algorithm is activated (black dots), the temperatures approach their nominal value y_I^N. We can also compare the inputs in Fig. 7.9. When agent I is selfish, its control increases while other houses' controls decrease, displaying a shift in the negotiation, and giving more resources to the attacker. However, when the correction is activated, it approaches the nominal.

Another way to represent the performances of our proposition, however, more palpably is to compare the accumulated objective functions during the period of simulation for each of the three scenarios (Table 7.1). We also calculate the percentual error $\frac{J^i - J^N}{J^N}$ for the cases S and C. When agent I attacks the system, we see a decrease in its objective ($\approx -40\%$), degrading the overall performance in $\approx +10\%$. When the correction mechanism is activated, the absolute percentual error is in the order $|\frac{J^C - J^N}{J^N}| \leqslant 10^{-8}$.

Table 7.1 – Objective functions J_i (% error).

Agent	Scenario N	Scenario S	Scenario C
I	19868.2	12618.5 (−36.5)	19868.2 (−0.0)
II	13784.5	18721.1 (35.8)	13784.5 (0.0)
III	17276.0	22324.9 (29.2)	17276.1 (0.0)
IV	10086.0	13872.4 (37.5)	10086.0 (0.0)
Global	61014.7	67536.9 (10.7)	61014.7 (−0.0)

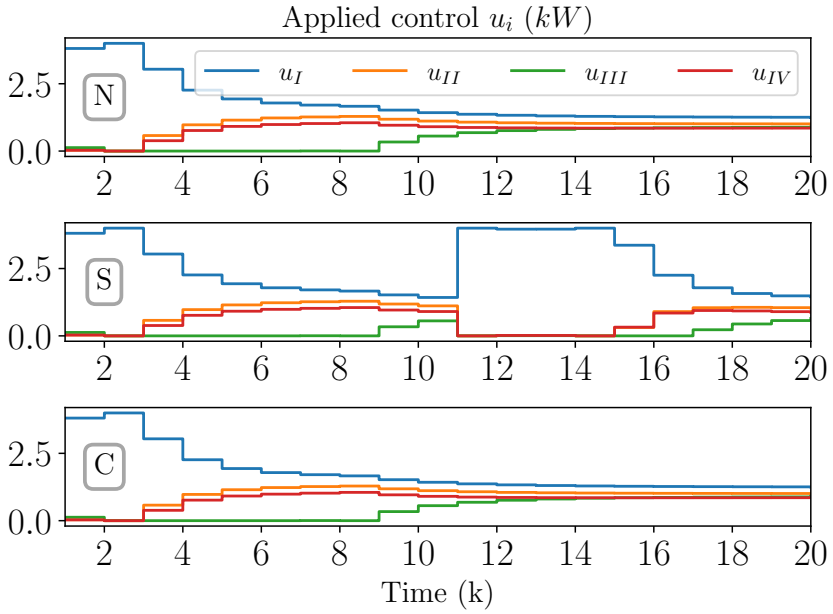


Figure 7.9 – Control applied in all houses for the 3 scenarios.

7.4 Conclusion

In this chapter, we relaxed an assumption from the problem in Chapter 6. We analyzed the impacts of this relaxation on the local problems and on the negotiation. We could observe how the complexity of the problem increases exponentially with the number of constraints of the local problems, which is defined by the global constraints and the prediction horizon.

We created the assumption of artificial scarcity, so we can counter some difficulties and propose detection and mitigation strategies.

However, part of the complexity reappears when estimating parameters. Our affine regression model becomes a PWA regression model, and we need to use an adequate estimation method that can estimate parameters using such a regression model.

We presented the *algorithm, which, although it may be adequate for such estimation, uses a probabilistic model. We adapt our model to respect its needs and use it to estimate the parameters needed for our detection and mitigation phases.*

Finally, we presented our resilient algorithm using artificial scarcity and tested it using our DHN study case.

Chapter 8

Conclusion

But your new shoes are worn at the heels
And your suntan does rapidly peel
And your wise men don't know how it feels
To be thick as a brick

Thick as a Brick
Jethro Tull

8.1 Retrospective and Contributions

In this thesis, we studied the security of CPS under dMPC. In particular, we studied attacks in primal decomposition, which was never studied before to the best of our knowledge.

We began with a general introduction to MPC, showing different MPC problems and presenting the need to decompose them when the systems are large. We then presented the optimization decomposition framework, which is used in various works in different flavors. We generalize those methods and show the need to divide the computation into different entities.

After we studied the distribution of the computation into those entities, discussing what the designer can choose and the possible communication those entities need to carry out to solve the problems. We also discussed the part trust took when dividing communication between entities, some examples of the real world were given as analogies to ease comprehension.

We then focused on anomalous behaviors, how they impact the communication of such entities and how they influence the solutions to the problems. We began with definitions of such behaviors and security in CPS. We generalized the vulnerabilities of CPS to the behaviors above.

After this, we shifted our attention to the most critical types of anomalous behaviors, attacks. We defined what attacks are, the most known attacks in the literature, with some examples to illustrate, and ways to classify them. Then, we discussed the study of attacks on the dMPC community, which is still in its first baby steps (less than 5 years).

Then, we presented the more usual ways to secure CPS, with examples, from the

prevention of anomalous behaviors to robust and resilient strategies to mitigate the effects of these behaviors.

We then presented the MPC decomposed using the primal decomposition, not studied under attacks before. We presented the decomposition, giving some interpretation of its different steps. Finally, we presented some of its vulnerabilities and illustrated by an example the effects attacks can produce under this decomposition method.

Inspired on reducing the effects of the attacks, we took an attack model and study some strategies on how to recover a nominal behavior.

We created some assumptions to ease the analysis of the system under the given attack model. We analyzed the main effects of the attacks and how they are connected with the attack model by changing its values slightly. From these analyses, some insights emerged on how to detect the attacks under those assumptions and mitigate their possible effects by reconstructing some variables that were corrupted by the attack.

We relaxed then one of those assumptions. We studied how the relaxation impacts the complexity of the solutions to the problem. We concluded the relaxation increased the complexity of the problem exponentially. We again analyzed the attacks under this exponentially more complex problem, and we proposed to add some less restraining assumptions to aid in circumventing the complexity of the system. With these assumptions, we could adapt the detection and mitigation strategies already proposed.

8.1.1 Contributions

Apart from the systematization of the regard needed to the design of decomposed CPS, we can list the contributions of this work as follow:

- Study of vulnerabilities of Primal decomposition-based dMPC
- Resilient strategies for two complexity-increasing different kind of systems
  - Deprived systems (where demand is greater than total resources)
  - Systems with possible artificial scarcity (sensible optimal demand)

8.1.2 Shortcomings

The main shortcoming of the approach is its need for some scarcity information of the system. First, it was given by the nature of the system, having constraints that deprived it of achieving its setpoints. Then, although the system was not deprived, the complete scarcity could be imposed by the coordinator by artificially changing the allocations. However, we needed to suppose the artificial scarcity point to respect the local constraints. The open question is what happens when we cannot impose complete scarcity, when there are no artificial scarcity points.

8.2 Possible Future directions

As with any research work, many questions emerged during the study, and we were confronted with many crossroads, which obliged us to choose one direction.

Bearing in mind the shortcomings presented here as well as the choices we list some of those questions and paths not chosen. And hope it serves as inspiration for future works. We list them in order of least backtracking needed, i.e., how fundamental the changes would need to be to achieve the study.

— Study of the robustness of the resilient strategy under communication noise

— Partial reconstruction of the cheating matrix, when $\widehat{\widetilde{P}}_i^{(0)}[k]$ is not available

— Adaptation of resilient strategy when using soft constraints [AB09]

— Use recursive EM (or alternative) to estimate parameters

— Study of adapting our resilient strategy to use accelerated stop (as in [DBG17])

— Study of our resilient strategy under other decomposition methods

— Study of other attack models (initially with affine cheating function)

Appendices

Annexe A

Résumé étendu en français

A.1 Contexte et Motivation

L'évolution de la computation dans les dernières deux décades est indéniable. Dispositifs avec la même puissance des ordinateurs qui ont possibilité l'humanité à toucher le sol lunaire dans la décade de 1970 sont aujourd'hui à la distance d'une main.

Cette évolution a proportionné l'utilisation de la commande prédictive [GPM89], en anglais connue comme Model-based Predictive Control ou MPC, en problèmes avec échelles de temps réduites, quelquefois en temps-réel [BLM08], et en utilisant dispositifs plus petits qu'une pièce [BM14].

Aussi comme conséquence la Model-based Predictive Control (MPC) est envisagée pour contrôler systèmes dans une myriade d'applications avec l'échelle de villes, comme les réseaux de distribution d'eau [Zha+21b], en anglais Water Distribution Networks (WDNs), et de chaleur [Tay+21], en anglais District Heating Networks (DHNs). Ça présente l'insertion de la MPC dans les systèmes cyber-physiques, en anglais Cyber-Physical Systems (CPS), où ordinateurs et machines sont étroitement couplées.

Cependant, pour quelques systèmes de grande échelles, le calcul peut encore être coûteuse, nécessitant de stratégies, comme diviser le calcul en plusieurs dispositfs, à fin de faciliter la computation. Comme on verra cette stratégie est appelée commande prédictive distribuée, ou distributed Model based Predictive Control (dMPC), et cela vient en differents moutures.

Néanmoins, pas trop d'études ont été faites pour la sécurité des stratégies dMPC, quand les agents ne communiquent honnêtement. Ce travail étudie ce que se passe quand les unités ne travaillent pas ensemble.

Pour illustrer on donne un example d'un réseau DHN avec 4 maisons (Fig. A.20) qui n'a pas assez de puissance pour répondre aux nécessités de ses résidents, nécessitant d'un compromis.

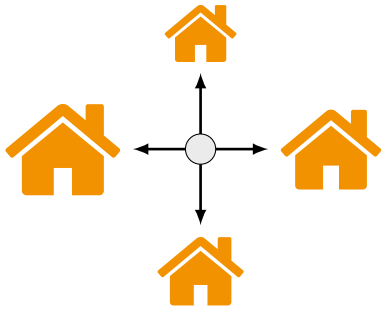


FIGURE A.1 – DHN avec 4 maisons.

La commande est faite utilisant MPC décomposée avec un coordinateur e un agent local pour chaque maison. On voit le schéma de communication entre les entités dans la Fig. A.2. Les allocations de ressources $\boldsymbol{\theta}_i$ sont proposées pour chaque maison par le

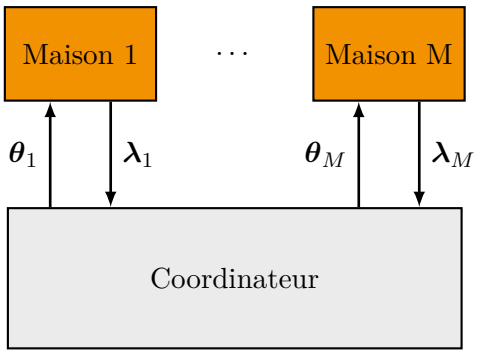


FIGURE A.2 – Échange entre contrôleurs locaux et coordinateur.

coordinateur. Le contrôleurs locaux respondent avec des indices $\boldsymbol{\lambda}_i$ qui indiquent leur satisfaction. Les allocations sont mises à jour basées sur ces indices jusq'un consensus soit établi.

Dans un certain moment, un des agents, plus spécifiquement celui correspondant à la maison I, commence a modifier la valeur de $\boldsymbol{\lambda}_i$ envoyée. Il l'augmente a chaque jour jusqu'à un jour la commande est arretée due à une manque de consensus entre les agents.

En utilisant des métriques comme les fonctions objectifs, dans laquelle valeures plus petites signifiquent une meilleure performance, on peut voir que la performance de l'agent I a amélioré à partir d'une certaine date à la fin d'octobre (Fig. A.3). Par contre, la performance de toutes les autres agents a empiré. On peut voir aussi dans l'aire hachuré quand le consensus n'est plut trouvé.

Cet example simple nous donne les effets d'un attaque : perte de performance et cassure du fonctionnement du système. Le même peut arriver causé par des fautes. Motivé par exemples réels comme les black-out de 2010 au Brési [Con10], l'attaque Stuxnet au Iran en 2011 [Lan11], et les attaques à une central électrique en Ukraine in 2016 [Bin17] et mal-heusement beaucoup d'autres [Din+18 ; Dib+19], on veut étudier la sécurité des CPS.

Dans ce travail on va focaliser sur la securité des CPS commandé par dMPC. On

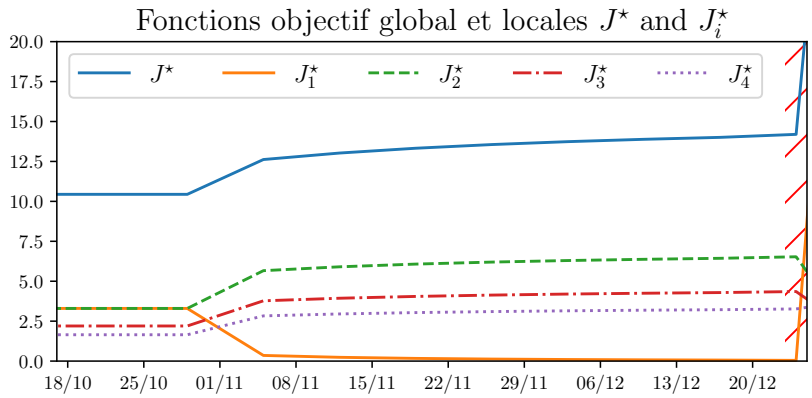


FIGURE A.3 – Graphique des fonctions objectifs dans une période de 10 semaines.

focalise dans le cas qu'ils sont attaqués par un agent mal intentionné, qui change le valeurs de quelques variables pour son propre bénéfice. Comme on verra cet attaque normalement est appelé an attaque de type False Data Injection (FDI).

Utilisant la connaisance acquise on propose des méthodes pour sécuriser ses systèmes. On utilise comme guide les questions suivantes :

— **Peut-on detecter l'attaque ?**
— **Peut-on identifier l'attaquant ?**
— **Peut-ont miniser les effets du dit attaque ?**

Cette thèse a comme objectif répondre ces questions dans un cas spécifique qui sera formalement presenté. Pour mieux comprendre et répondre ces questions, on divise ce travail en deux parties :

La première partie (§A.2 à A.4) sert comme une gentille introduction au déssin des dMPC, à sa sécurité et à des attaques.

Pour un · e lecteur · ice pas familiarisé · e, Section A.2 explique ce qui est une Commande Prédictive et les défis pour la décomposer. Section A.3 discute les topologies possibles pour fair la décomposition. Section A.4 définit comportements anormaux, donne quelques exemples, les catégorise et présente les méthodes pour les prévenir et combattre.

La seconde partie (§A.6 à A.8) contient les contributions de cette thèse.

Section A.6 présente la décomposition étudié (décomposition primale). On presents ses vulnérabilités et comment elles peuvent affecter la performance du système. On formalise l'example en donnant une approche plus quantitative. Une fois les vulnérabilités et les possibles effets des attaques découverts, on divise la mitigation en parties plus gérable. Premièrement en Section A.7, on analyse un problème simple, pour qu'on puisse apprécier

les possibles difficultés qui peuvent être rencontrées pendant le problème de mitigation. À partir de cette analyse, on propose des mécanismes de detection et mitigation, suivis par un example académique pour illustrer leur fonctionnement.
Après, en Section A.8, on analyse un problème similaire mais avec un coup de théâtre. Comme on verra, une petite modification du problème inital cause une augmentation exponentiel de sa complexité. L'analyse de ce nouvel problème résulte une stratégie similaire, mais avec modifications adéquates pour renfermer la nature exponentiel du problème.

Finallement, on conclue le travail avec Section A.9, où on discute les résultats de l'étude, ses bénéfices et inconvénients. La discussion ouvre des questions qui peuvent inciter des nouveau travaux.

A.1.1 Contributions

Comme un contribution mineure, on démontre l'étude de la vulnérabilité des dMPC basées sur la décomposition primale, qu'au meilleur de nos connaissance n'a pas jamais été étudié.

Comme contributions majeures, deux stratégies de mitigation pour types différents de système de complexité croissante.

La première pour des systèmes lesquels les ressources ne sont pas suffisants pour répondre aux nécessités des agents locaux, on les appelle **Systèmes Dépourvus**.

La seconde, pour une classe des systèmes exponentielment plus complexes, où on assume que les demandes locales respect au moins quelques contraintes. On utilise tel hypothèse pour acquérir des informations en utilisant une méthode qu'on appelle **pénurie artificielle**.

A.1.2 Publications

Les discussion dans cette thèse a donné comme résultats les publications suivantes :

— Publiés

[NBG21] Conference article for the SysTol'21

[Nog+22] Conference article for the NecSys'22

A.1.3 Code

Le code pour tout ce travail (texte, exemples, figures et algorithmes) est disponible en https://github.com/Accacio/thesis/

A.2 Commande Prédictive et sa décomposition

La commande prédictive basée en modèle, ou commande prédictive, ou encore Model-based Predictive Control en anglais, est une stratégie de commande en boucle fermée basée dans la solution des problèmes d'optimisation. Ayant un modèle du système à commander et une fonction objectif, la stratégie utilise le modèle pour prévoir l'évolution de ses états et computer une séquence optimale de command qui optimise la fonction donnée. Comme on utilise des problèmes d'optimisation, il est naturel d'ajouter des restriction en forme de contraintes d'égalité et d'inégalité.

Cette stratégie a une place spéciale dans l'industrie et est présente dans une plethora d'applications (gestion d'énérgie [Ana+18], commande des quadrotors [BM14] et autres).

Dans ce travail, on focalise sur la MPC pour des systèmes linéaires avec entrées contraintes.

A.2.1 MPC pour des systèmes linéaires

Comme la MPC est normalement développée en utilisant un ordinateur digital et la transmission des signaux continus peuvent demander une bande passante infinite [HML22], naturellement on utilise un modèle de temps discret pour le système a commander :

$$\begin{aligned} \boldsymbol{x}[k+1] &= f(\boldsymbol{x}[k], \boldsymbol{u}[k]) &= A\boldsymbol{x}[k] + B\boldsymbol{u}[k] \\ \boldsymbol{y}[k] &= C\boldsymbol{x}[k] \end{aligned} \tag{A.1}$$

où $\boldsymbol{x}[k] \in \mathbb{R}^{n_x}$ est l'état du système et $\boldsymbol{u}[k] \in \mathbb{R}^{n_u}$ est l'entrée.

On suppose que les entrèes du système sont contraintes par des contraintes affines comme

$$\Gamma \boldsymbol{u}[k] \preceq \boldsymbol{u}_{\max}, \tag{A.2}$$

avec Γ de taille adéquate $\mathbb{R}^{\#\boldsymbol{u}_{\max} \times n_u} = \mathbb{R}^{n_c \times n_u}$.

On suppose qu'on a un objectif de commande $\boldsymbol{v}$, qui peut être *rejeter perturbations*, où $\boldsymbol{v}[k] = \boldsymbol{x}[k]$, ou *suivi de trajectoire*, où $\boldsymbol{v}[k] = \boldsymbol{w}[k] - \boldsymbol{x}[k]$, soit $\boldsymbol{w}[k]$ la trajectoire à suivre. On utilise comme example le suivi de trajectoire de sortie (utilisant les relations entre $\boldsymbol{x}[k]$ et $\boldsymbol{y}[k]$), mais tout les résultats de ce travail sont aussi valable pour l'autre objectif.

La MPC trouve une séquence d'entrées $\boldsymbol{U}^\star[k] = [\boldsymbol{u}^\star[0|k]; \dots; \boldsymbol{u}^\star[N-1|k]]$ prédite dans un horizon $\mathcal{N} = \{1 : N\}$ qui minimiser la fonction $J : \mathbb{R}^{n_x} \times \mathbb{R}^{N \cdot n_u} \to \mathbb{R}$, définie comme

$$J(\boldsymbol{x}[0|k], \boldsymbol{U}[k]) = \sum_{i \in \mathcal{N}} \left[\|\boldsymbol{v}[i|k]\|_Q^2 + \|\boldsymbol{u}[i-1|k]\|_R^2 \right]. \tag{A.3}$$

Cette fonction, quand minimisée, doit assurer l'objectif $\boldsymbol{v}[k]$ choisi en minisant l'énérgie de l'entrée. Les matrices $Q \in \mathbb{S}_+$ et $R \in \mathbb{S}_{++}$ sont des poids qui représentent les coûts

respectives de chaque terme de l'équation.

En mettant en form matriciel, le problème résolu par la MPC est le suivant

$$\begin{aligned} \underset{\boldsymbol{U}[k]}{\text{minimiser}} \quad & \|\boldsymbol{V}[k]\|_{\bar{Q}}^2 + \|\boldsymbol{U}[k]\|_{\bar{R}}^2 \\ \text{sous} \quad & \boldsymbol{x}[i|k] = A\boldsymbol{x}[i-1|k] + B\boldsymbol{u}[i-1|k] \quad \forall i \in \mathcal{N} \quad , \\ & \bar{\Gamma}\boldsymbol{U}[k] \preceq \boldsymbol{U}_{\max} \end{aligned} \tag{A.4}$$

où

$$\bar{\Gamma} = I_N \otimes \Gamma \tag{A.5}$$

$$\boldsymbol{U}_{\max} = \mathbf{1}_N \otimes \boldsymbol{u_{max}} \tag{A.6}$$

$$\bar{Q} = I_N \otimes Q \tag{A.7}$$

$$\bar{R} = I_N \otimes R. \tag{A.8}$$

En choisissant l'approche *batch* [BBM17, Chapter 8.2], on peut récrire le problème comme

$$\begin{aligned} \underset{\boldsymbol{U}[k]}{\text{minimiser}} \quad & \|\boldsymbol{U}[k]\|_H^2 + 2\boldsymbol{f}[k]^T\boldsymbol{U}[k] + c[k] \\ \text{sous} \quad & \bar{\Gamma}\boldsymbol{U}[k] \preceq \boldsymbol{U}_{\max} \end{aligned} \tag{A.9}$$

où, pour le suivi de trajectoire,

$$H = \|\mathcal{Y}^u\|_{\bar{Q}}^2 + \bar{R} \tag{A.10}$$

$$\boldsymbol{f}[k] = \mathcal{Y}^{uT}\bar{Q}(\mathcal{Y}^x\boldsymbol{x}[0|k] - \boldsymbol{W}[k]) \tag{A.11}$$

$$c[k] = \|\mathcal{Y}^x\boldsymbol{x}[0|k]\|_{\bar{Q}}^2 - 2\boldsymbol{W}[k]^T\bar{Q}\mathcal{Y}^x\boldsymbol{x}[0|k] + \|\boldsymbol{W}[k]\|_{\bar{Q}}^2 . \tag{A.12}$$

avec les prédictions

$$\underbrace{\begin{bmatrix} \boldsymbol{y}[1|k] \\ \boldsymbol{y}[2|k] \\ \vdots \\ \boldsymbol{y}[N|k] \end{bmatrix}}_{\boldsymbol{Y}[k]} = \underbrace{\begin{bmatrix} CA^1 \\ CA^2 \\ \vdots \\ CA^N \end{bmatrix}}_{\mathcal{Y}^x} \boldsymbol{x}[0|k] + \underbrace{\begin{bmatrix} CA^0B & 0 & \dots & 0 \\ CA^1B & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ CA^{N-1}B & \dots & \dots & CA^0B \end{bmatrix}}_{\mathcal{Y}^u} \boldsymbol{U}[k] \quad . \tag{A.13}$$

Si on divise la fonction objectiv par 2 et on ignore le terme constant $c[k]$, on a la structure standard d'une forme de programmation quadratique, connue comme Quadratic Programming (QP) :

$$\begin{aligned} \underset{\boldsymbol{U}[k]}{\text{minimiser}} \quad & \tfrac{1}{2}\|\boldsymbol{U}[k]\|_H^2 + \boldsymbol{f}[k]^T\boldsymbol{U}[k] \\ \text{sous} \quad & \bar{\Gamma}\boldsymbol{U}[k] \preceq \boldsymbol{U}_{\max} \end{aligned} \quad . \tag{A.14}$$

Remarque 1.1

Observe que les problèmes (A.14) et (A.9) sont equivalents (même solution), mais ils ne sont pas le même problème, alors si on doit calculer la fonction objectif (A.9) on doit utiliser la bonne version.

La communauté de la MPC utilise beaucoup les QP, une fois qu'existe une grande quantité de solveurs mathematiques pour les résoudre, on peut citer comme exemples le solveur interne de MATLAB [1], OSQP [2], MOSEK [3] et aussi ECOS [4].

Par contre, la solution dans cette forme **monolithique** peut être intensive au niveau de calcul, dépendant des valeurs de n_x, n_u, n_c and N. Pour améliorer la solution on utilise des méthodes de décomposition. On peut décomposer en différents échelles de temps [Che+11], utilisant théorie des jeux [MMC11] ou algorithmes génétiques associés avec des observateur d'états [Xie+16]. Mais dans ce travail on utilise des techniques de décomposition basées dans la décompostion de problèmes d'optimisation [Gis+13].

A.2.2 Décomposition de problèmes d'optimisation

Comme montré en [Con+06] et [Boy+15], un problème d'optimisation est décomposable s'il y a plus qu'une variable de decision et on peut le diviser en au moins deux sub-problèmes.

Si les sous-problèmes ne sont pas couplés, on peut les résoudre en parallel et la solution est trouvée, sans aucune interaction entre les sous-problèmes. Cependant, s'ils sont couplés, on doit utiliser une méthodologie pour trouver la bonne solution.

Les problèmes peuvent être couplés par variables ou par contraintes, ces variables et contraintes sont appelées compliquants. Comme vue en [Boy+15], une contrainte compliquant le problème peut être transformée en une variable qui complique le problème et vice versa. Dans ce travail on utilise juste les contraintes compliquantes.

L'idée générale est de prendre un problème décomposable comme

$$\begin{array}{lll} \underset{\text{variables de decision}}{\text{minimiser}} & \text{objective}_1 & + \quad \text{objective}_2 \\ \text{sous} & \text{constraintes}_1, & \text{constraintes}_2 \\ & \text{contraintes compliquantes}_{12} & \end{array} \tag{A.15}$$

et le récrire comme un problème équivalent avec des variables auxiliaires, comme

$$\underset{\substack{\text{variables de decision} \\ + \\ \text{variables auxiliaires}}}{\text{minimiser}} \quad \text{objective}_1^*(\text{variables auxiliaires }) + \text{objective}_2^*(\text{variables auxiliaires}) \tag{A.16}$$

1. https://fr.mathworks.com/help/optim/ug/quadprog.html
2. https://osqp.org
3. https://www.mosek.com
4. https://github.com/embotech/ecos

où objective$_1^*$(variables auxiliaires) et objective$_2^*$(variables auxiliaires) sont calculés en résolvant les sous-problèmes

$$\begin{aligned} &\underset{\text{variables de decision}_1}{\text{minimiser}} && \text{objective}_1 \\ &\text{sous} && \text{constraints}_1 \\ &&& \text{contraintes compliquants}_{12_1}(\text{variables auxiliaires}_1) \end{aligned} \tag{A.17a}$$

$$\begin{aligned} &\underset{\text{variables de decision}_2}{\text{minimiser}} && \text{objective}_2 \\ &\text{sous} && \text{constraints}_2 \\ &&& \text{contraintes compliquants}_{12_2}(\text{variables auxiliaires}_2) \end{aligned} \tag{A.17b}$$

dont les solutions dépendent de ces variables auxiliaires ajoutées. Pour trouver la valeure optimale des variables auxiliaires, il est nécessaire la coordination des sous-problèmes.

Le problème equivalent est appelé *problème principal*, il est résolu par la coordination entre les sous-problèmes en utilisant la mise à jour des variables auxiliaires, appelées *variables d'interface*. Les méthodes de mis à jour sont basées sur un algorithme d'optimisation, comme la bisection, plans sécants, ou d'autres méthodes qui utilisent le (sub)gradient ou des approximations. La dernière classe est la plus utilisée, et quelques exemples sont l'iteration de Arrow-Hurwicz-Uzawa [BGB12], sub-gradient projeté [Bie+12], et la plus forte descente [Boy+11]. La forme exacte depend de la topologie utilisée (discuté en §A.3).

Les sous-problemès, on les appelle *problèmes locaux*. Ils sont obtenus en utilisant des méthodes d'equivalence entre problèmes d'optimisation [BV04], comme le problème original (primal) [Pau+16 ; CNN22], le dual [Mor+11 ; BGB12 ; Vel+18], utilisant des opérateur comme l'opérateur proximal [Iid19 ; OV14], ou d'autres stratégies.

Les *problèmes locaux* ont un ensemble de contraintes formé par les contraintes derivées des contraintes décomposables du problème original, appelées *contraintes locales*, et quelques autres contraintes derivées de la partie compliquante, *contraintes globales d'accouplement*. La solution de cette partie des contraintes d'accouplement dépendent des variables d'interface.

Les variables d'interface dépendent des problèmes locaux choisis. Par example, pour des problèmes primaux, normalement les variables duales sont utilisées [Coh78] ; pour les problèmes duals, le résidu des contraintes [Boy+15] ; et pour la méthode d'alternance de direction des multiplicateurs, en anglais Alternating Direction Method of Multipliers (ADMM), variables primales et autres multiplicateurs sont utilisés (la mise à jour de ses multiplicateurs donnent le nom de la méthode) [Boy+11].

Ce travail se concentre à la décomposition primale et utilise le sub-gradient projeté. Cette décomposition sera expliquée dans une section future (§A.6). Pour d'autres exemples on réfère [MN+14] ou [Con+06].

A.3 Differents topologies

Comme dit, la forme de la décomposition dépend dans la topologie utilisé pour résoudre le problème. Dans cette section on montre le différentes classification des topologies.

A.3.1 Par distribution des unités de computation

La première forme de catégoriser la topologie est par comme on distribue les unités de computation.

Ces unités peuvent être géographiquement séparés ou pas (même hardware) dépendant de l'échelle du CPS controllé. Si le système a une grande échelle et peut être distribué géographiquement, naturellement on essaye de positionner le hardware stratégiquement pour correspondre aux sous-systèmes, mais pas toujours il est possible. On peut voir dans la Fig. A.4 des exemples quand les sous-problèmes correspondent ou pas aux sous-systèmes.

(a) Non correspondant.

(b) Correspondant.

FIGURE A.4 – Décomposition du problèmes en (P_i) et du système en (S_i).

Dans la littérature, c'est normal de choisir les unités de façon que ça correspond comme en Fig. A.4b [ACM21], alors, dans ce travail on considère aussi que les sous-problèmes et sous-systèmes **correspondent**. De cette façon, les termes sous-problèmes, sous-systèmes, agents et unités seront utilisés indistinctement.

A.3.2 Par communication

Normalement, comme montré en §A.2.2, la communication n'est pas nécessaire si les sous-problèmes ne sont pas couplés. Cependant, il existe des cas pour la dMPC où on peut exploiter des propriétés de robustess de la MPC pour computer la solution sans communication [VSK14]. Pour d'autres decomposition d'optimisation il est possible montrer que dans certaines circonstances la communication n'est pas nécessaire non plus [VE22].

Remarque 1.2

Généralement, dans la littérature de la dMPC le terme “décentralisé” se refère aux méthodes où les agents ne se communiquent [Chr+13, §4],[NM14]. Mais le terme peut être confus et dans certains cas même “mal-utilisé”. Donc, on opte pour une nomenclature différente, an appellant ces méthodes comme “contrôle non coordonné”, une fois qu’il n’y a pas de coordination entre agents ni un agent coordinateur. On utilise le terme “décentralisé” comme opposé de “centralisé” (monolithique), i.e. pour décrire structure au lieu de la communication.

Comme dans la majorité de la littérature où grande part des contrôles son coordonnés [NM14 ; ACM21], dans **ce travail**, on suppose aussi que les agents se communiquent. On considère aussi qu’il existe au moins un chemin de communication entre tous les agents (le **graphe** équivalent est **connecté**).

A.3.3 Par relations de pouvoir

Une autre classification est par l’influence que les agents exercent les uns sur les autres. Par example, quelques agents peuvent avoir plus de pouvoir (être plus importants) qu’autres. Ce genre de relation est appellé hiérarchique, normalement le pouvoir est démontré par le différentes types d’actions réalisées par les agents. Normalement les plus importants ont des rôles de médiateur, certificateur, régulateur et etc. Pour les systèmes où il n’y a pas de différence de comportement ou d’influence entre les agents, on les appelle anarchiques. Dans Fig. A.5 on voit des exemples de ces deux types de structures.

(a) Hierarchie. (b) Anarchie

FIGURE A.5 – Topologies hierarchiques et anarchiques.
Noeuds plus grands représentent plus de pouvoir/influence sur autres.

Quelques fois la structure est impliquée par la décomposition. Par example, il y a des décomposition où des agents résolvent leur problèmes un après l’autre jusqu’à qu’il aye une convergence (Fig. A.6a) [LMC09]. Ces schémas sont appelés *séquentiels*. L’hierarchie est impliqué une fois que le premier agent résout son problème en ignorant les autres, lui donnant, même que juste temporairement, une position de pouvoir.

En quelques autres les agents résolvent ses problèmes indépendamment et partagent quelques résultats jusqu'à un consensus soit trouvé, itérativement ou pas (Fig. A.6b) [Liu+10]. Ces structure sont appelées *parallèles* et il n'a pas d'hierarchie.

(a) Topologie séquentiel. (b) Topologie parallèle.

FIGURE A.6 – Topologies séquentiel et parallèle.

Dans d'autres cas, on peut choisir par l'hiérarchie ou anarchie. Ça dépendant normalement de comment se passent les relations de confiance entre les possibles agents. On peut choisir faire confiance à quelques agents pour l'échange des informations, ainsi comme on fait dans la vrai vie avec le système financier, agences d'inspection et régulation dans différent domaines. En [McN+18] et [Ola+18], on voit le parallèle entre les structures de commande et les relations politiques/sociétales.

Dans ce travail on utilise une structure hierarchique qui sera mieux décrite en §A.6.

A.3.4 Par type de communication

Le type de communication peut aussi être utilisé pour catégoriser les topologies, Les agents peuvent se communiquer uni ou bi-directionnellement. Si les agents utilisent un schéma de communication bi-directionel, il est plus simple à gérer, une fois qu'ils peuvent se communiquer directement. Par contre si le schéma est uni-directionel il faut utiliser des autres moyens pour prouver leur comunication (s'il existe un chemin entre eux), ou pour prouver la convergence des algorithmes [GS10].

Une autre classification par type de communication c'est si la communication est synchrone ou asynchrone. En générale, les schémas synchrone sont plus simples, alors que les asynchrone dépendent de politique de communication spécifiques. Broadcast et gossiping [GS10] sont des exemples de ce type de communication.
Pour le broadcast (uni-directionnel), un agent transmet l'information à tout ses voisins sortant ($\mathcal{N}_{\text{sortie}}$) et chaque destinataire met à jour ses variables locales.
Gossiping peut être divisé en deux, asymétrique et symétrique. La version asymétrique (uni-directionel), l'agent choisit un des ses voisins sortant, et lui envoie l'information, le destinataire met à jour ses variables locales quand le message arrive. De l'autre côté, dans le cas symétrique (bi-directionel), après avoir mis à jour les variables, le destinataire devient transmetteur, envoyant ses informations au expéditeur original, qui aussi met à jour ses propres variables une fois que ce nouvel message arrive.

Dans ce travail on suppose que les agents se communiquent bi-directionnellement et de manière synchrone.

A.4 Comportements anormaux

Comportements anormaux sont n'import quel comportement ou changement de comportement inattendu d'un système.

Les deux causes principales sont *défauts* et *attaques*. La différence principale entre les deux est *l'intention*. Alors que défauts se passent involontairement, non coordonés et sans objectif, attaques sont intentionnel, généralement coordonés avec un objectif malicieuse.

Comme les deux peuvent changer le comportement du système nous sommes intéressés en maintenir l'opération normale des CPS.

A.4.1 Security

La sécurité évoque l'opération sûre d'un système. Pour ça, on utilise comme base la définition pour la cyber-sécurité et on l'étend pour les CPS.

La *cyber-security* a trois pilliers : confidentialité, intégrité, et disponibilité, en anglais Confidentiality, Integrity and Availability (CIA) [Bis05].

Confidentialité des données et ressources. Parfois on a besoin de mettre en secret quelques parties du système et on veut qu'il soit acessible pour un groupe privilégié. Ça peut être pour raisons de privacité, pour eviter la diffusion des données personnels ou pour éviter l'exposition des vulnérabilités du système (un exemple dans la culture pop sont les plans de l'étoile noire de la saga Star Wars).

Intégrité des données et ressources. Un équipement nouveau est plus digne de confiance qu'un ancient qui a probablement souffert plus des intempéries. Dans la transmission de données l'intégrité peut être divisé dans l'intégrité de l'information (l'information reçue est la même qu'a été envoyée) et l'intégrité de l'identité du expéditeur/destinataire (authentification).

Disponibilité des données et ressources quand demandés par un utilisateur. Comme mentionné en [Bis05], "un système indisponible est au moins aussi mauvais qu'aucun système du tout".

Les vulnérabilités d'un système peut compromettre n'importe lequel de ces pilliers, parfois multiples au même temps.

A.4.2 Vulnérabilités des CPS

Les CPS ont des composant des domaines physiques et cyber. On peut les diviser en partie matériel et immatériel. Chacune de ces parties sont des points de vulnerabilité d'un CPS.

Dispositifs physiques comme des transducteurs (capteur/actionneur) peuvent être cibles de sabotage. Par example un attaquant peut utiliser un objet frais pour changer la lecture des capteurs de témperature d'une salle. Mais ils peuvent aussi se détériorer avec le temps ou d'autres accidents naturelles (chute d'arbres ,glissement de terres, attaques d'animaux, ouragans etc) et causer des défauts.

Le software peut aussi être une source de vulnérabilité à cause des *bogues*, qui peut interrompre le fonctionnement du système ou même permettre un *hacker* de s'infiltrer et gagner accès au système. L'ordinateur utilisé pour la commande peut être mal dimensionné (puissance de calcul insuffisante), et recevoir plus de requêtes qu'attendu, arrivant à un état de surcharge cessant de répondre.

A.4.3 Attaques in CPS

Il existe plusiers modèles de catégorisation de attaques en CPS. Ici on montre les deux plus connus. Le premier modèle est basé dans la cyber-security appélé 3D ou DDDpar l'acronyme anglais Disclosure, Deception and Denial of Service (DDD). **Disclosure** est tout accès non autorisé à l'information (rupture de confidentialité). **Deception** est l'idée d'utiliser de l'information fausse pour tromper quelqu'un (rupture d'intégrité). **Disruption** est l'interruption du fonctionnement du système (rupture de la disponibilité). Aussi on peut ajouter l'**usurpation** d'identité (rupture de la authenticité).

On définit comme un attaque n'importe laquelle action mal-intentionné qui utilise les vulnérabilités d'un CPS à fin de violer sa sécurité. L'agent est appellé attaquant.

Utilisant comme base [Bis05], [CAS08], [ACS09], et autres, on peu créer une abstraction des CPS dans un réseau (Fig. A.7).

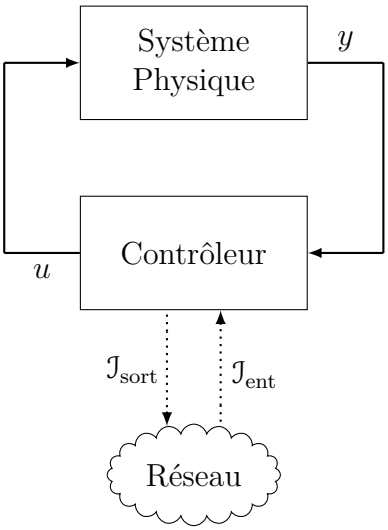


FIGURE A.7 – Abstraction Générale d'un CPS dans un réseau.

Dans le diagramme, u correspond à la commande calculé par le contrôleur et appliquée au système physique à travers des actionneurs. y correspond aux sorties des capteurs du

système. Les signaux I_{ent} et I_{sort} sont les informations reçu et envoyées utilisant le réseau (envoyées ou venues d'autres sous-systèmes).

Avec le modèle DDD, on peut catégoriser les attaques par où dans le système ils surviennent. Dans la Fig. A.8 on voit différents types d'attaques énumérés de A1 à A13. Les attaques de A1 à A4 sont des attaques du type *disclosure*, qui se passent au niveau de l'information. L'attaquant observe le trafic d'information pour une utilisation postérieure. Ce genre d'attaque est aussi appellé **snooping** ou **eavesdropping** (furetage et écoute illicite). A5 à A8 representent attaques du type *deception*. Aussi au niveau informationnel, mais dans ce cas le signaux sont modifiés (representés par un ˜ dans le symbole de la variable). Les attaques A10 à A12 sont du type *disruption*, quand l'attaquant attaque le canal de c communication, empêchant la communication entre deux entités. Et finallement, A13 sont des attaques physiques, ou le système physique est attaqué directement. Cet attaque change physiquement le comportement du système.

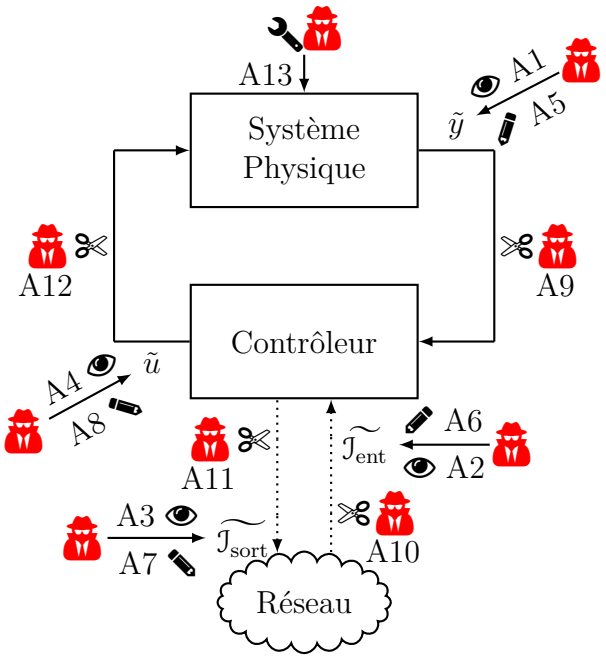


FIGURE A.8 – Attaques à un CPS dans un réseau.

Un autre modèle utilisé est celui présenté par [Tei+15], où les auteurs font valeurs que les attaques ne sont pas pures, ils peuvent être mélangés. Un attaquer peut, par exemple, enregistrers les informations (attaque *disclosure*) pour utiliser les données acquises pour les renvoyées dans le futur avec intention de tromper le contrôleur (attaque *deception*). On appelle cet attaque un attaque du type **Replay** [ZM14]. Pour ça les auteurs utilises troix axes pour répérer les types d'attaques, Connaissance du modèle, Perturbation de ressources et divulgation (disclosure) (Fig. A.9).

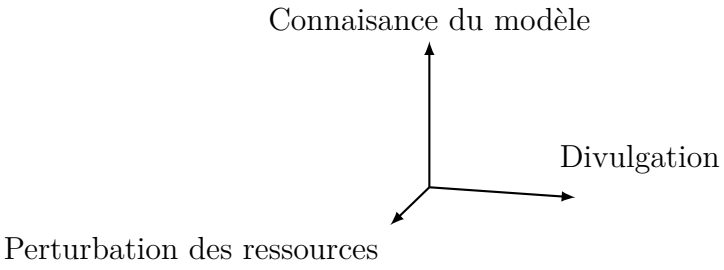


FIGURE A.9 – Espace d'attaques avec 3 axes.

Pour illustrer on peut peupler le space d'attaque avec des différents types d'attaque (Fig. A.10).

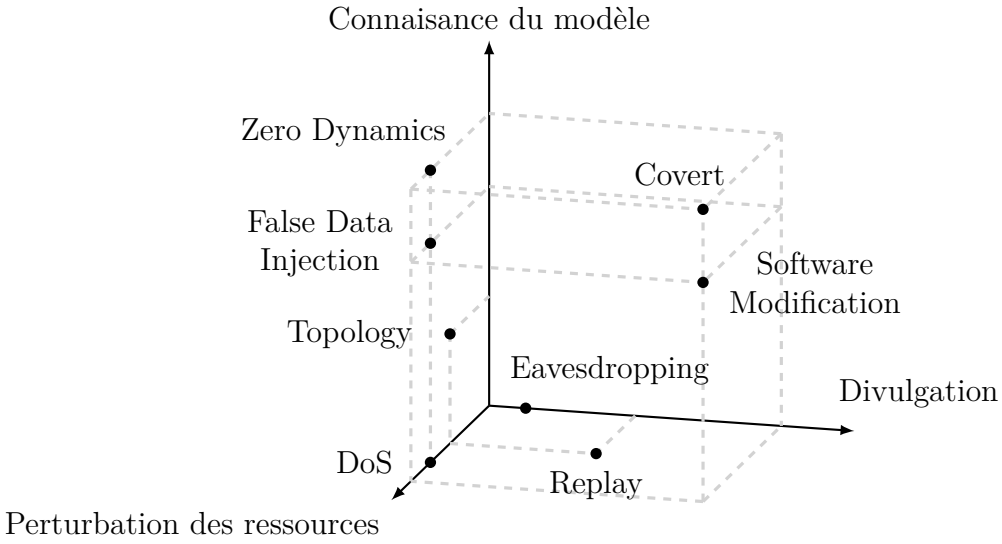


FIGURE A.10 – Espace d'attaques peuplé.

Les attaques montrés dans la figure peuvent être rétrouvés dans les travaux suivants

Denial of Service (DoS) [SY19 ; Zha+20 ; Yan+19]

Topology [KT13 ; WDL16 ; ZLW21].

Software Modification [Lan11].

FDI [PDB13].

Covert ,[Smi15 ; HZ16 ; Bar+20].

Zero Dynamics [Tei+12 ; Tei+15 ; HZ16]

La liste des attaques cités n'est pas extensive mais répresente le attaques plus étudié dans la littérature. Autres attaques peuvent être trouvé en [Tei+15 ; ZLW21] et dans autres textes cités dasn cette section.

Dans ce travail, nous nous intéressons aux attaques FDI, où avec un peu de connaissance sur le système, l'attaquant modifie le trafique d'informations pour altérer son comportement pour son propre bénéfice.

A.4.4 Attaques à la dMPC

Comme montré par [ACM21], la sécurité des CPS a été étudie au cadre des MPC [SY19 ; FLT22], mais la communauté de la dMPC ne l'a pas étudié suffisamment.

Dans la plupart des travaux les agents coopèrent et aucune vulnérabilité est exploré. Cependant, dans les dernières années un effort a été fait pour explorer la vulnérabilité de quelques méthodes décomposition, de manière plus perceptible la décomposition duale [Vel+17b ; Vel+17a ; Vel+18 ; Ana+18 ; Ana+19 ; Ana+20] et la décomposition de Jacobi-Gauß [CMI18].

Dans ces travaux les attaques ne sont pas comme origine un agent externe. Un (ou un groupe) des agents se comporte de manière non coopérative et essaye de piloter la décomposition de façon a déstabiliser le système ou se bénéficier. Ces attaques peuvent être appellés **attaques inter-agents** et parfoir considéré comme **attaques interne** (comme si un agent interne avec accès suffisant prend contrôle du système). Normalement les objectives sont accomplis utilisant des attaques du type *deception*.

Les travaux présentent quatre types de attaques FDI, que sont catégorisé à partir d'où l'information est modifiée.

L'attaque dénommé **Selfish** (égoïste) quand l'attaquant modifie sa fonction objectif (généralement la multipliant par une constante positive) pour qu'elle soit plus pénalisée, causant les autres agents d'augmenter ses effort pour équilibrer la désavantage et decroitre le coût local de l'attaquant.

L'attaque **Fake weights** (poids faux), l'agent non-cooperative utilise des poids faux pour sa fonction objectif locale. L'attaque *selfish* peut être considéré comme un cas spécifique de ce genre d'attaque.

Supposant que l'agent suit une référence, dans l'attaque **false reference** (référence fausse) l'agent malicieux utilise une valeur différente pour la référence pour calculer sa fonction objectif locale.

Pour l'attaque **fake constraints**, l'attaquant utilise des contraintes fausses pour résoudre le problème d'optmisation.

Le dernier attaque présenté est le **"liar" agent** (agent menteur) où après la fin de la négotiation entre les agents, l'attaquant utilise un contrôle différent de ce qui été convenu.

A.4.5 Sécurisant le CPS

Pour sécuriser le CPS, il faut d'abord évalue les vulnérabilités et choisir les plus riquées à être attaquées ou à souffert un défaut. Des méthodes existent pour diférrent types de systèmes, quelques exemples sont donnés en [Bis05] et [WL13].

Après on essaye de protéger les éléments plus susceptibles en utilisant des méthodes de **prévention**.

Comme exemples de protection on peut citer *renfermer* composants physiques entre

parois, et le portes utilisés pour les accéder doivent avoir *contrôle d'accès*. On peut aussi ajouter des *dispositfs de surveillance* (cameras et des alarmes) [CAS08 ; Din+18], persuadant l'attaquant à ne pas s'approcher des équipements. Les défauts causés par *détérioration* sont prévenu par la *maintenance préventive périodique* [Che+21], et la substitution des composantes détériorées par des nouvelles plus neuves.

Pour chaque vulnérabilité il existe une mésure de prévention, *rajeunissement de software* [Alo+12 ; Gri+20], et *cryptography* [Din+18] sont d'autres méthodes utilisés.

Si la prévention faille, il est nécessaire d'avoir d'autres stratégies. Dans la littérature, ces statégies sont presque toujours accompagné de deux mots : robustesse et résilience. Bien que ces termes sont utilisé indistinctement, dans ce travail on donne des significations un peu différentes.

Robustesse peut être définie par l'habilité de tolérer des perturbation sans changer de fonction [Jen03]. **Résilience** est l'habilité de maintenir un niveau acceptable d'opération en réponse des perturbations, incluant les ménaces de nature inattendue et malicieuse [Rie10].

On interprète les méthodes robustes comme des stratégies plus passives, pendant que les méthodes résilientes plus adaptatifs, actifs de façon à rétablir un comportement plus optimal.

Alors, on divise les méthodes en deux catégories : méthodes actives et passives.

A.4.6 Stratégies Passives

Les méthodes passives sont normalement basées sur le contrôle robuste, où la loi de commande implémentée ne change pas quand en présence des fautes ni d'attaques.

La plupart des méthodes robustes sont utilisé pour la commande tolérante aux défauts, en anglais Fault Tolerant Control (FTC), par contre, à cause de quelques similarités, ces méthodes sont aussi utilisées contre des attaques [Tei+15 ; Din+18 ; ACM21].

Un exemple c'est pour attaques FDI qui changes une variable et alors utilisant une méthode robuste [VSK14] on peut contrôller le système malgré la déviation.

Un autre exemple est pour le réseau de distribution, on utilise un index de sécurité appelé N-1 statique, qui indique quant stable le réseau se comporte quand un des composants présent un défaut (ou est disconnecté) [Qia+22]. Velarde et al. en [Vel+18] utilisent une solution similaire basée sur la robustesse f des graphes [5] [DI15 ; WI19] pour créer une dMPC robuste basée sur la décomposition duale. Pendant la négotiation, les valeurs extrêmes (les f plus petites et plus grandes) sont ignorées.

Encore une autre stratégie est presentée en [Vel+17a ; Mae+21], où des scénarios historiques (des trajectoires de la MPC) sont connus avec des probabilités d'occurrence associées sont enregistrées et utilisées pour résoudre le problème avec le calcul des agents,

5. Le Graphe maintient la connectivité même si un nombre f d'agents est déconnecté

qui peut être peu fiable à cause des attaques.

A.4.7 Stratégies Actives

Les méthodes actives sont normalement bi-modales. Un mode pour quand il n'y a pas d'attaque et un autre pour quand le système change de comportement.

À fin d'identifier les comportements anormaux, on a besoin premièrement d'une information de comportement normal du système, par exemple la dynamique de quelques variables, des bornes ou n'importe quel autre information utile. Avec cette connaisance, on peut superviser les composants menacés et créer des moniteurs. Cette partie est appellée la phase de **Détection**, ou l'attaque peut être dectecté à partir des dits moniteurs et détecteurs. Dans quelques cas la phase est associé d'un pas d'isolation, ou l'agent qui se comporte mal est identifié.

Quand l'attaque est détecté, la loi de commande change pour qu'on puisse mitiger les effets du comportement anormal. Cette phase est appelée **Mitigation** ou **Récupération**.

A.4.8 Détection

La détection peut être catégorisée dans différentes façons. Si un signal additionnel doit être ajouté, on l'appéle **détection active** sinon **détection passive**.

Une autre division est s'elle est basée en événements (**event-triggered**) ou pas.

On donne [SY19 ; Hu+21 ; Sun+22] comme exemples basés en méthodes du type *event-triggered*.

Autre catégorisation c'est par l'utilisation de connaisance analytique du système ou par l'utilisation de méthodes d'apprentissage.

Exemples de méthodes actives est le *watermarking* (filigrane) où on surimpose un signal (pseudo) aléatoire d'authentification au signal monitoré [MS09 ; MWS15 ; SK17 ; KV17 ; LGG21]. Ce méthode est connu pour protèger contre les attaques du type replay, une fois que la semence du signal aléatoire sert comme un générateur unique d'horodatage.

Un exemple passive et analytique sont les the observateurs utilisées en [HZ16] ou les résidus de la dynamique des états [Tei+15 ; Boe+20], qui peuvent être associés à un test d'hypothèse [MS09] (détecteurs χ^2), ou encore la détection à partir de la détection d'appartenance à quelques ensembles [For+16 ; MTI18]. D'autres exemples de décteurs sont trouvés en [PBB12 ; PDB13 ; Zha+21a ; ACM21].

Pour les méthodes d'apprentissage, on peut citer [Ana+18 ; Ana+19 ; Ana+20] qui utilises une approche bayésiènne pour apprendre quelques bornes et détecter agents suspects. Un autre exemple et l'utilisation des réseaux neuronaux/apprentissage profonde pour créer des détecteurs. En [Hus+21], les auteurs entraînent un réseau neuronal convolutionnel pour détecter 91% des attaques Distributed Denial of Service (DDoS) d'un certain genre.

En [BAL20b ; BAL20a], les auteurs utilisent un problème d'optimisation pour identifier quel partie du système été attaquée.

Encore une autre méthode est d'utiliser le contrôle des coalitions [CMC21] qui peut potentiellement regrouper les agents sains et détecter changements de topologie.

Remarque 1.3

Comme indiqué en [ACM21] il est important d'observer que le choix de la borne de détection influencie les faux positifs associés.

A.4.9 Mitigation/Récupération

La phase de récupération consiste en contrôle de dégâts, modifiant le systme de façon à rétablir le comportement normal.

L'option principale de récupération est la substitution immédiate des composants en mauvais fonctionnement (ou attaqués) par leur redondance (s'il y a une) à fin de prévenir répercussions plus profondes. Après on peut réparer le composant pour le réutiliser (si possible).

Quand une substition immédiate n'est pas possible, la loi de commande est changée. Le composant est déconnecté et on utilise des stratégies robustes, qui peuvent mitiger les effets du comportement anormal. Ces stratégies font le système se comporterr pas forcément de la manière originale, mais d'une manière sub-optimale satisfaisante, jusque la normalité soit restaureée.

Des exemples peuvent être [MTI18] dans le quel une MPC robuste utilisant la robustesse *tube-based* est utilisé contre les perturbations dans une zone de sûreté. Similairement à [Vel+18], en [Ana+18 ; Ana+19 ; Ana+20] les auteurs basent leur solution dans la robustesse f-local des graphes, où les agents suspects sont ignorés pendant la négotiation de la dMPC.

Une autre option est d'utiliser les stratégies adaptatives qui essayaient d'estimer variables, ou ensembles et les utiliser pour compenser les perturbations causées par les comportement anormaux [YZG22 ; LCK20].

A.5 Comparaison avec état de l'art

Ce travail utlise [Vel+17b ; CMI18] comm inspiration pour démontrer les vulnérabilités du méthode de décomposition primale, qui n'ont jamais été présentées.

Comme les autres travaux adjacents, nous nous intéressons par des attaques entre agents du type **FDI**. Le modèle exact est présenté dans la prochaîne section.

En [Vel+17b ; Vel+18], les auteurs présentent des méthodes robustes baséees sur la théorie des graphes, où l'attaquant n'est pas detecté par soi, mais agents suspects sont déconnectés ou ignorés. En [Vel+17a ; Mae+21], une autre méthode robuste est utilisée, mais cette fois là avec une approche basée sur des scénarios, où séquences suspectes reçoivent poids différents.

Nous optons, par contre, comme dans [Ana+18 ; Ana+19 ; Ana+20], par une méthode **active** et **résiliente**. La stratégie est plus détailée dans les prochaines sections, mais pour la détection, on utilise une méthode **active**, où nous ajoutons des données pour détecter l'attaquant.. On utilise une méthode hybride de connaissance analytiques de la décomposition et aussi des méthodes d'apprentissage pour créer le dit détecteur. Pour la récupération, on utilise les mêmes connaissances analytiques utilisées pour la détection à fin de réconstruire quelques variables et compenser les perturbations causées par l'attaque.

Comme la liste de travaux adjacents à ce travail est curte, on peut les compiler et comparer dans un tableau. Le Tab. A.1 compare les travaux avec les méthodes présentées dans cette thèse.

	Décomposition	Présente vulnérabilités ?	Résiliente/Robuste	Détection	Mitigation
[Vel+17a] [Mae+21]	Duale	Oui	Robuste (Scénario)	NA	NA
[Vel+17b] [Vel+18]	Duale	Oui	Robuste (f-robustesse)	NA	NA
[CMI18]	Jacobi-Gauß	Oui	–	–	–
[Ana+18] [Ana+19] [Ana+20]	Duale	Oui	Résiliente	Analyt./Apprent.	Déconnexion (Robustesse)
Notre	Primale	Oui	Résiliente	Active Analyt./Apprent.	Reconstruction de données

TABLE A.1 – Comparaison entre travaux adjacents. NA signifie non applicable.

A.6 Vulnérabilités de la decomposition primale

A.6.1 La dMPC baséee sur la décomposition primale

On prend le problème de la MPC en sa forme monolitique présentée en (A.14), et on le divise en M parties de même taille, qui correspondent à une sousdivision du système initial (A.1). La dynamique des sous-systèmes est décrite par des matrices d'état (A_i, B_i, C_i), et sous états $\boldsymbol{x}_i[k]$, références $\boldsymbol{w}_i[k]$ et entrées $\boldsymbol{u}_i[k]$.

On donne un index dans l'ensemble $\mathcal{M} = \{1 : M\}$ pour chaque sous-système et on récrit (A.14) comme

$$\begin{array}{ll} \underset{\boldsymbol{U}_1[k],\dots,\boldsymbol{U}_M[k]}{\text{minimiser}} & \sum_{i\in\mathcal{M}} \left[\frac{1}{2}\left\|\boldsymbol{U}_i[k]\right\|^2_{H_i} + \boldsymbol{f}_i[k]^T\boldsymbol{U}_i[k]\right] \\ \text{sous} & \sum_{i\in\mathcal{M}} \left[\bar{\Gamma}_i\boldsymbol{U}_i[k]\right] \preceq \boldsymbol{U}_{\max} \end{array}, \tag{A.18}$$

avec des versions locales $\boldsymbol{U}_i[k] = [\boldsymbol{u}_i[0|k]; \dots; \boldsymbol{u}_i[N-1|k]]$, $\bar{\Gamma}_i$, H_i et $\boldsymbol{f}_i[k]$ des variables presentées en §A.2.1.

Pour la décomposition primale, on divise (A.18) en un problème principal (A.19b) et M problèmes locaux (A.19a) basés sur le problème original (primal) (A.18) et utilisant des variables d'interface $\boldsymbol{\theta}_i[k], \forall i \in \mathcal{M}$:

$$\bar{J}_i^\star(\boldsymbol{\theta}_i[k])[k] = \begin{array}{ll} \underset{\boldsymbol{U}_i[k]}{\text{minimiser}} & \frac{1}{2}\left\|\boldsymbol{U}_i[k]\right\|^2_{H_i} + \boldsymbol{f}_i[k]^T\boldsymbol{U}_i[k] \\ \text{sous} & \bar{\Gamma}_i\boldsymbol{U}_i[k] \preceq \boldsymbol{\theta}_i[k] : \boldsymbol{\lambda}_i[k] \end{array} \tag{A.19a}$$

$$\bar{J}^\star[k] = \begin{array}{ll} \underset{\boldsymbol{\theta}_i[k],\dots,\boldsymbol{\theta}_M[k]}{\text{minimiser}} & \sum_{i\in\mathcal{M}} J_i^\star(\boldsymbol{\theta}_i[k]) \\ \text{sous} & \sum_{i\in\mathcal{M}} \boldsymbol{\theta}_i[k] \preceq \boldsymbol{U}_{\max} \end{array}, \tag{A.19b}$$

où $\boldsymbol{\lambda}_i[k], \forall i \in \mathcal{M}$ sont des variables duales des problèmes, i.e. les multiplicateurs de Lagrange associés aux contraintes [BV04].

Le problème (A.19b) est résolu par mises à jour de θ_i jusqu'à une convergence. La méthode iterative choisie est le sous-gradient projeté, dans laquelle on utilise $\boldsymbol{\lambda}_i[k]$ comme sous-gradient [BV04 ; Boy+15]. La loi de mise à jour est donné par

$$\boldsymbol{\theta}[k]^{(p+1)} = \mathrm{Proj}^{\mathcal{S}}(\boldsymbol{\theta}[k]^{(p)} + \rho^{(p)}\boldsymbol{\lambda}[k]^{(p)}). \tag{A.20}$$

où p est un pas d'itération, $\rho^{(p)}$ est la largeur du pas d'itération [6], $\boldsymbol{\theta}[k] = [\boldsymbol{\theta}_1[k]; \dots; \boldsymbol{\theta}_M[k]]$, $\boldsymbol{\lambda}[k] = [\boldsymbol{\lambda}_1[k]; \dots; \boldsymbol{\lambda}_M[k]]$. La projection est faite sur l'ensemble $\mathcal{S} = \{\boldsymbol{\theta}[k] \mid I_c^M\boldsymbol{\theta}[k] \preceq \boldsymbol{U}_{\max}\}$ [7].

Comme chaque problème (A.19a) peut être résolu en parallèle, une unité de computation est attribuée pour résoudre chaque un des sous-problèmes. Pour la mise à jour (A.20),

6. choisie comme montré en [Con+06]
7. $c = \#\boldsymbol{\lambda}_i[k] = \#\boldsymbol{\theta}_i[k] = \#\boldsymbol{U}_{\max} = n_c N$, et $I_c^M = \mathbf{1}_M^{\mathrm{T}} \otimes I_c$

les agents ont besoin des $\boldsymbol{\lambda}_i[k]$ des autres agents pour mettre à jour leur propre $\boldsymbol{\theta}_i[k]$. On pourrait créer une solution anarchique comme celle en [Vel+18], où l'information $\boldsymbol{\lambda}_i[k]$ circulerait entre les agents et chaque agent mettrait à jour son $\boldsymbol{\theta}_i[k]$. Cependant, nous avons décidé d'utiliser une structure plus privative. On a opté pour une solution **hierarchique**, où un agent agrège les $\boldsymbol{\lambda}_i[k]$ et coordonne les mises à jour des $\boldsymbol{\theta}_i[k]$, on appelle cet agent le *coordinateur*. De cette façon, chaque agent a accès juste à ses propres valeurs de $\boldsymbol{\lambda}_i[k]$ et $\boldsymbol{\theta}_i[k]$.

On illustre la structure de communication en Fig. A.11. On peut voir la structure hierarchique vu en §A.3. L'Algorithm 5.2 resume la dMPC basée sur la décomposition primale

Algorithm A.1: La dMPC basée sur la décomposition primale.

Initialisation du Coordinateur:
Initialiser k, p, $p_{\max}$ et ϵ
tant que $k \geqslant 0$ **faire**
Initialiser $\boldsymbol{\theta}_i[k]^{(p)}$, $\forall i \in \mathcal{M}$, tel que $\boldsymbol{\theta}[k]^{(p)} \in \mathcal{S}$
Coordinateur envoie $\boldsymbol{\theta}_i[k]^{(p)}$ pour tous les agents
Échange entre Coordinateur et agents:
répéter
Agents résolvent sous-problèmes (A.19a) et envoient $\boldsymbol{\lambda}_i[k]^{(p)}$ au coordinateur
Coordinateut met à jour les $\boldsymbol{\theta}[k]$ utilisant (A.20) et les renvoie
$p \leftarrow p + 1$
jusqu'à $\left[\left\|\boldsymbol{\theta}[k]^{(p)} - \boldsymbol{\theta}[k]^{(p-1)}\right\| \leqslant \epsilon\right] \vee [p \geqslant p_{\max}]$
Agents appliquent la dernière commande $\boldsymbol{u}_i^\star[0|k]$ en son sous-système respectif
$k \leftarrow k + 1$

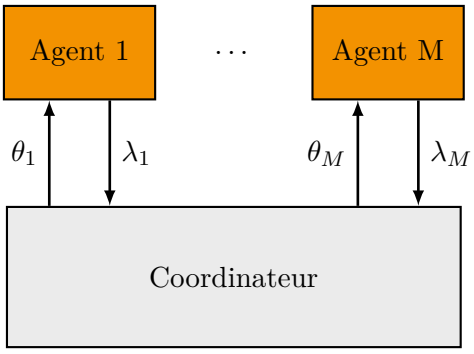
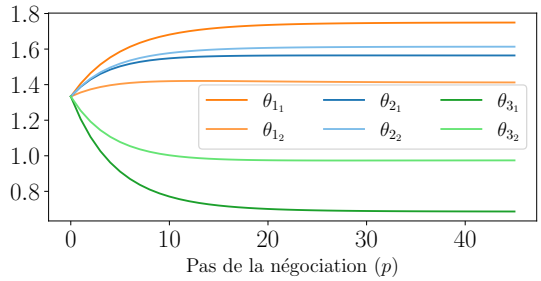


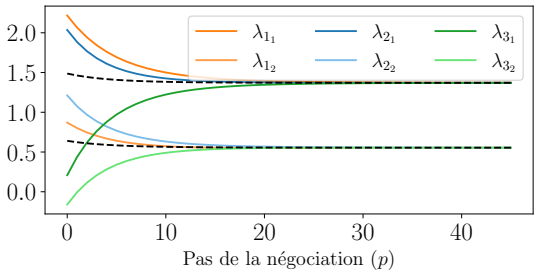
FIGURE A.11 – Échange entre agents et coordinateur pendant une dMPC basée sur la décomposition primale.

Si on donne un example, on peut voir la dynamique des $\boldsymbol{\theta}_i$ et $\boldsymbol{\lambda}_i$ en Fig. A.12.

Les valeurs de $\boldsymbol{\theta}_i$ représentent une allocation de l'entréé maximale donnée aux agents par le coordinateur, et $\boldsymbol{\lambda}_i$ est une mésure de insatisfaction de chaque agent à la valeur



(a) Évolution de $\boldsymbol{\theta}_i$.



(b) Évolution de $\boldsymbol{\lambda}_i$.

FIGURE A.12 – Évolution des variables pendant procès itératif.

allouée. Le rôle du coordinateur est de négocier les allocations jusqu'à que tous les agenst soient également insatisfaits. Le procès itératif est appellé la **phase de négociation**.

À cause de ce procès d'allocations itératives, autres noms sont donés à la décomposition, comme **allocation des ressources** ou **décomposition par quantités**.

A.6.2 Attaque d'intérêt

Comme les $\boldsymbol{\lambda}_i$ sont les sous-gradients des problème locaux (A.19a), il est possible de voir qu'ils portent de l'information du problème, i.e. il y a d'une certaine façon l'information de la 5-tuple $(H_i, \boldsymbol{f}_i[k], \bar{\Gamma}_i, \boldsymbol{\theta}_i[k])$, qu'on appelle l'information locale complète $\mathfrak{I}_i$. Alors, les attaques en [Vel+18] (montrés en §A.4.4) vont réfléchir en modifications du $\boldsymbol{\lambda}_i$.

Pour cette raison, nous nous intéressons en attaques du type FDI entre agents qui modifient leur propre $\boldsymbol{\lambda}_i$.

On suppose qu'avant chaque négociation, l'attaquant décide une fonction $\gamma : \mathbb{R}^c \times \mathbb{R} \to \mathbb{R}^c$, pour modifier $\boldsymbol{\lambda}_i$ avant de l'envoyer au coordinateur. Pour faciliter l'analyse on suppose la fonction est linéaire comme

$$\tilde{\boldsymbol{\lambda}}_i[k] = \gamma(\boldsymbol{\lambda}_i[k], k) = T_i[k]\boldsymbol{\lambda}_i[k]. \tag{A.21}$$

On appelle $T_i[k]$ la matrice de triche.

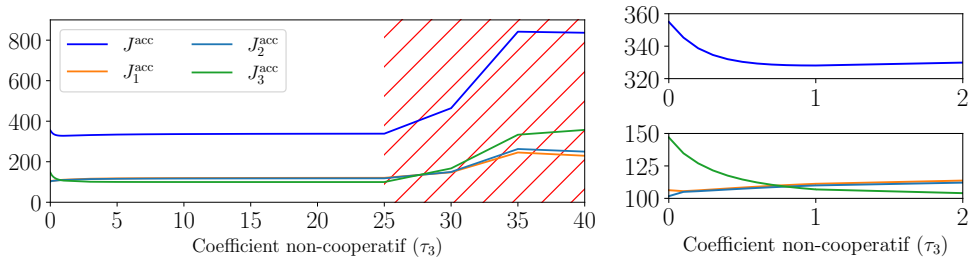
Cet attaque peut être interprété comme un cas générale du attaque dit selfish [Vel+18], où la constante positive qui multiplie la fonction objectif locale est traduite par une matrice de triche diagonale

$$T_i[k] = \tau_i[k] I_c. \tag{A.22}$$

On illustre avec un example d'un système avec 3 sous agents. On fait que l'agent 3 attaque le système avec (A.21). On simule pour scénarios différents. Pour chaque scénario, agent 3 utilise un coefficient de triche $\tau_3[k]$ distinct.

En Fig. A.13a et A.13b, on accumule les fonctions objectif locales $J_i^{\text{acc}} = \sum_{k \in \mathcal{K}} J_i^\star(\boldsymbol{\theta}_i^\star[k])[k]$ et aussi l'objectif total $J^{\text{acc}} = \sum_{i \in \mathcal{M}} J_i^{\text{acc}}$ pour chaque valeur différent de τ_3.

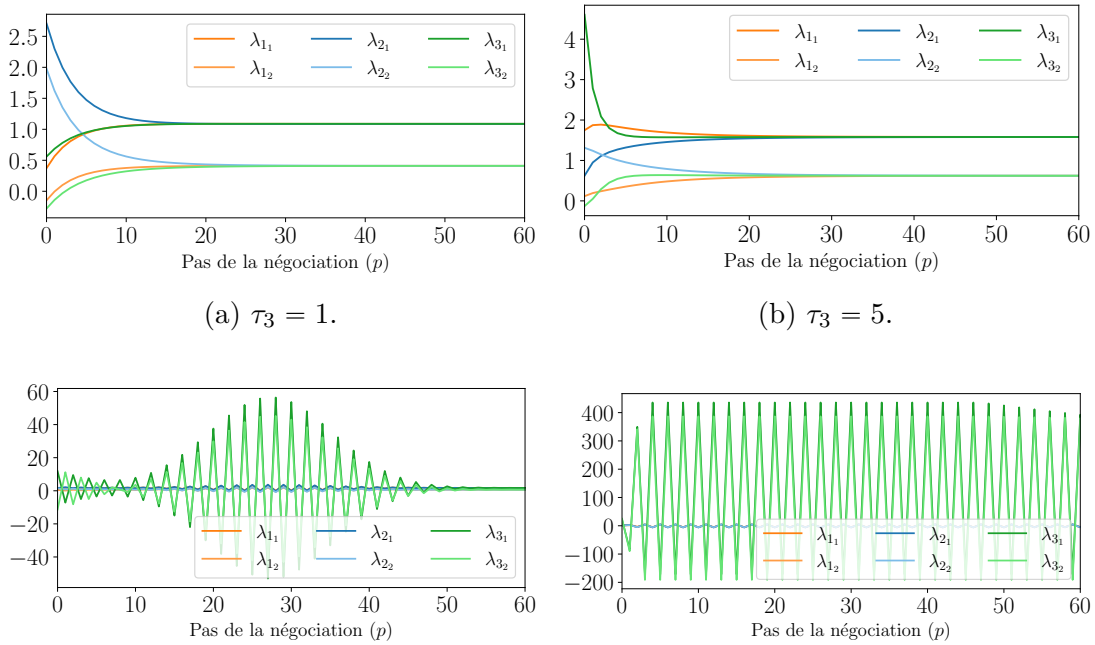
Observe (Fig. A.13b) que J^{acc} a sa valeur la plus petite quen $\tau_3 = 1$, i.e., quand il n'y a pas de triche. Quand τ_3 augmente, les fonctions objectif de l'attaquant (J_3^{acc}) décroît au même moment que celles des autres augmentent. L'opposé se passe pour des valeurs de τ_3 entre 0 et 1. Dans ce cas l'attaquant détériore sa propre performance, ce qui ne justifie pas tel attaque, sauf dans un scénario de détournement.



(a) (b) (detail)

FIGURE A.13 – Accumulate objective functions for different values of τ_3.

Un autre fait peut être vu dans la zone hachurée en rouge de la Fig. A.13a. Les fonctions objectif trouvent l'équilibre mais recommencent à augmenter. Cela est causé par un effondrement de la négociation et les agents n'entrent plus en consensus. On présente la négociation de $\boldsymbol{\lambda}_i$ pour 4 valeurs différentes de τ_3 (Figs. A.14).



(a) $\tau_3 = 1$. (b) $\tau_3 = 5$.

(c) $\tau_3 = 20$. (d) $\tau_3 = 40$.

FIGURE A.14 – Evolution of $\boldsymbol{\lambda}_i$ during negotiation for $k = 5$ with different values of τ_3.

Comme on voit, quand τ_3 augmente, la négociation présente un comportement oscillatoire atténué. L'attenuation diminue si τ_3 augmente, jusqu'à la négociation oscille et aucun consensus soit achévé, cassant la dMPC et peut être aussi le système.

A.7 Commande Prédictive résiliente pour systèmes dépourvus

A.7.1 Systèmes dépourvus sous attaque

Différemment de [Vel+18 ; Mae+21], notre solution pour la dMPC sécurisée est résiliente basée sur la détection active utilisant des méthodes d'apprentissage et des connaissances analytiques du problème. which propose robust solutions, we propose a resilient dMPC based on a hybrid of analytical/learning active detection method. Alors, on commence par analyser le système sous attaque.

A.7.2 Systèmes dépourvus

Premièrement, on rappelle le problème équivalent de la MPC en forme monolitique (A.14), reproduit ici par convenience,

$$\begin{array}{ll} \underset{\boldsymbol{U}[k]}{\text{minimiser}} & \frac{1}{2}\left\|\boldsymbol{U}[k]\right\|_H^2 + \boldsymbol{f}[k]^T\boldsymbol{U}[k] \\ \text{sous} & \bar{\Gamma}\boldsymbol{U}[k] \preceq \boldsymbol{U}_{\max} \end{array}, \tag{A.14}$$

et on rappelle les problèmes locaux (A.19a) de la décomposition primale,

$$\bar{J}_i^\star(\boldsymbol{\theta}_i[k])[k] = \begin{array}{ll} \underset{\boldsymbol{U}_i[k]}{\text{minimiser}} & \frac{1}{2}\left\|\boldsymbol{U}_i[k]\right\|_{H_i}^2 + \boldsymbol{f}_i[k]^T\boldsymbol{U}_i[k] \\ \text{sous} & \bar{\Gamma}_i\boldsymbol{U}_i[k] \preceq \boldsymbol{\theta}_i[k] : \boldsymbol{\lambda}_i[k] \end{array}. \tag{A.19a}$$

Les versions sans contraintes de ces problèmes sont

$$\underset{\boldsymbol{U}[k]}{\text{minimiser}} \quad \frac{1}{2}\left\|\boldsymbol{U}[k]\right\|_H^2 + \boldsymbol{f}[k]^T\boldsymbol{U}[k] \quad , \tag{A.23}$$

$$\underset{\boldsymbol{U}_i[k]}{\text{minimiser}} \quad \frac{1}{2}\left\|\boldsymbol{U}_i[k]\right\|_{H_i}^2 + \boldsymbol{f}_i[k]^T\boldsymbol{U}_i[k], \tag{A.24}$$

et ses solutions sont analytiques [BV04]

$$\mathring{\boldsymbol{U}}^\star = -H^{-1}\boldsymbol{f}[k], \tag{A.25}$$

$$\mathring{\boldsymbol{U}}_i^\star[k] = -H_i^{-1}\boldsymbol{f}_i[k]. \tag{A.26}$$

On appelle un système dépourvu quand la solution sans contrainte $\mathring{\boldsymbol{U}}^\star$ se situe dehors les frontières du polytope formé par les contraintes pour tout k, i.e.,

$$\bar{\Gamma}\mathring{\boldsymbol{U}}^\star \succ \boldsymbol{U}_{\max}, \forall k. \tag{A.27}$$

De plus, dans cette section, nous supposons que pour tous les sous-systèmes, leur

solutions sans contraintes $\mathring{\boldsymbol{U}}_i^\star[k]$ ne respectent pas les contraintes

$$\bar{\Gamma}_i \mathring{\boldsymbol{U}}_i^\star[k] > \boldsymbol{\theta}_i[k], \forall i \in \mathcal{M}, \forall k, \tag{A.28}$$

i.e., même si tous les ressources sont allouées pour un seul sous-système, il n'est pas capable d'achever ses besoin.

Ces hypothèses signifient que la solution optimal de chaque problème nécessiterait beaucoup plus de ressources que le disponible ($\boldsymbol{U}_{\max}$). Alors, les solutions de ces QP sont trouvées par la projection pondérée [8] des solutions sans contraintes sur le polytope. Normalement, le résultat se donne au périmètre du polytope.

En supposant que les contraintes ont au maximum la même quantité de lignes que colonnes [9], la solution peut être la même que pour un problème avec des contraintes d'égalité [10]

$$\begin{array}{cc} \underset{\boldsymbol{U}[k]}{\text{minimiser}} & \frac{1}{2}\left\|\boldsymbol{U}[k]\right\|_H^2 + \boldsymbol{f}[k]^T\boldsymbol{U}[k] \\ \text{sous} & \bar{\Gamma}\boldsymbol{U}[k] = \boldsymbol{U}_{\max} \end{array} . \tag{A.29}$$

Si on utilise la décomposition primale on a

$$\bar{J}_i^\star(\boldsymbol{\theta}_i[k])[k] = \begin{array}{cc} \underset{\boldsymbol{U}_i[k]}{\text{minimize}} & \frac{1}{2}\left\|\boldsymbol{U}_i[k]\right\|_{H_i}^2 + \boldsymbol{f}_i[k]^T\boldsymbol{U}_i[k] \\ \text{subject to} & \bar{\Gamma}_i\boldsymbol{U}_i[k] = \boldsymbol{\theta}_i[k] : \boldsymbol{\lambda}_i[k] \end{array} \tag{A.30}$$

comme problèmes locaux et pour la mise à jour on développe la projection pour une intersection de hyperplans résultant

$$\boldsymbol{\theta}_i^{(p+1)} = \boldsymbol{\theta}_i^{(p)} + \rho^{(p)}\left(\boldsymbol{\lambda}_i^{(p)} - \frac{1}{M}\sum_{j=1}^{M}\boldsymbol{\lambda}_j^{(p)}\right), \forall i \in \mathcal{M}. \tag{A.31}$$

Ces équations seront utilisées pour une brève analyse du système sous attaque.

Remarque 1.4

On utilise des systèmes dépourvus, car quand en manque de ressources, les agents doivent faire le compromis. Ça instigue la compétition entre eux et éventuellement des comportements égoïstes. Si ses besoins étaient achévés, il n'aurait pas de motivation pour une triche, tous les agents seraient satisfaits.

8. Un problème QP peut être vu comme une projection pondérée, dans laquelle la mésure de l'ensemble est pondérée par H et le point original est décalé utilisant $\boldsymbol{f}_i[k]$

9. Cette hypothèse est acceptable une fois que la plupart des contraintes de puissance sont du type somme des puissances, comme $\sum_{i\in\mathcal{M}} \boldsymbol{u}_i[k] \leqslant \boldsymbol{u}_{\max}$.

10. Dans notre cas, les contraintes forment un cône, il est facilement prouvé géométriquement que le point projeté est l'apex du cône.

A.7.3 Analysises des problèmes locaux

Comme les problèmes (A.30) sont des QP avec des contraintes d'égalité, on peut trouver une solution analytique [BV04].

En se basant sur la dualité Lagrangienne on est capable de trouver aussi une solution analytique pour $\boldsymbol{\lambda}_i[k]$. En calculant le Lagrangien et quelques gradients pour achever les conditions d'optimalité de Karush-Kuhn-Tucker (KKT) on a

$$\boldsymbol{\lambda}_i[k] = -P_i\boldsymbol{\theta}_i[k] - \boldsymbol{s}_i[k], \tag{A.32}$$

où $P_i = (\bar{\Gamma}_i H_i^{-1} \bar{\Gamma}_i^T)^{-1}$ et $\boldsymbol{s}_i[k] = P_i \bar{\Gamma}_i H_i^{-1} \boldsymbol{f}_i[k]$.

Comme attendu, $\boldsymbol{\lambda}_i[k]$ depend de $\boldsymbol{\theta}_i[k]$. De plus, à cause de la forme quadratique, la solution est affine.

Remarque 1.5

On peut remarquer, comme dit avant la solution intègre les information de la fonction objectif (présence de H_i et $\boldsymbol{f}_i[k]$), et aussi des constraintes (présence de $\bar{\Gamma}_i$ et $\boldsymbol{\theta}_i[k]$).

A.7.4 Analyse de la négociation

Utilisant (A.31) et (A.32), on peut observer les dynamiques de $\boldsymbol{\theta}_i$ et $\boldsymbol{\lambda}_i$ pendant la négociation. Cette analyse donne des idées pour une méthode future de détection et mitigation.

Dynamique de $\boldsymbol{\lambda}_i$

Si on substitue (A.31) en (A.32), on a pour chaque sous-système

$$\boldsymbol{\lambda}_i^{(p+1)} = -P_i \left(\boldsymbol{\theta}_i^{(p)} + \rho^{(p)} \left(\boldsymbol{\lambda}_i^{(p)} - \frac{1}{M} \sum_{j=1}^{M} \boldsymbol{\lambda}_j^{(p)} \right) \right) - \boldsymbol{s}_i[k]. \tag{A.33}$$

En forme matricielle on a la dynamique complète

$$\boldsymbol{\lambda}^{(p+1)} = \mathcal{A}_\lambda \boldsymbol{\lambda}^{(p)}, \tag{A.34}$$

où

$$\mathcal{A}_\lambda = \begin{bmatrix} I - \frac{M-1}{M}\rho^{(p)}P_1 & \frac{1}{M}\rho^{(p)}P_1 & \dots & \frac{1}{M}\rho^{(p)}P_1 \\ \frac{1}{M}\rho^{(p)}P_2 & I - \frac{M-1}{M}\rho^{(p)}P_2 & \dots & \frac{1}{M}\rho^{(p)}P_2 \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{M}\rho^{(p)}P_M & \frac{1}{M}\rho^{(p)}P_M & \dots & I - \frac{M-1}{M}\rho^{(p)}P_M \end{bmatrix}. \tag{A.35}$$

On voit que c'est un système à temps discret, en anglais Discrete Time (DT), homogène.

Le système varie avec le temps, à cause de ρ, mais on peut le choisir de façon qu'il disparaisse quand $p \to \infty$. Alors, on peu utiliser le critère de Lyapunov [Hes09, §8.6], i.e., les valeurs propres de $\mathcal{A}_\lambda$ doivent être dans le cercle unitaire pour que le système soit stable. Pour des systèmes petits, on peut utiliser le critère de Jury [Jur62] pour vérifier ça.

Analisant $\mathcal{A}_\lambda$ encore, on voit que la somme des ses colonnes est égale a un [11], signifiant que 1 est une valeur propre par la droite $\mathcal{A}_\lambda$:

$$\mathcal{A}_\lambda \mathbf{1} = \mathbf{1}$$

Alors, si on suppose que les autres valeurs propres sont dedans le cercle unitaire, le système converge à un état $\mathbf{1}c$, i.e., où tous les $\boldsymbol{\lambda}_i[k]$ sont égaux [GS10 ; XBK07]. Par contre, la somme des lignes n'est pas égal, alors la somme des états n'est pa constante, où comme dit dans la communauté des algorithmes de consensus, la masse n'est pas conservée. De cette façon, on ne pourrait pas utiliser des stratégies de consensus moyen, sauf si tous les systèmes avaient le même P_i, ce qui rendrait le problème extrêmement pas intéressent (la solution serait de diviser les ressources également entre les sous-systèmes).

Si le système est sous attaque, on utilise le modèle (A.21),

$$\tilde{\boldsymbol{\lambda}}_i[k] = T_i[k]\boldsymbol{\lambda}_i[k]. \tag{A.21}$$

Comme n'importe quel agent peut être l'attaquant, on suppose tous les agents attaquent simultanément le système résultant en :

$$\tilde{\boldsymbol{\lambda}}^{(p+1)} = \tilde{\mathcal{A}}_\lambda \tilde{\boldsymbol{\lambda}}^{(p)}, \tag{A.36}$$

où

$$\tilde{\mathcal{A}}_\lambda = \begin{bmatrix} I - \frac{M-1}{M}\rho^{(p)}T_1[k]P_1 & \frac{1}{M}\rho^{(p)}T_1[k]P_1 & \dots & \frac{1}{M}\rho^{(p)}T_1[k]P_1 \\ \frac{1}{M}\rho^{(p)}T_2[k]P_2 & I - \frac{M-1}{M}\rho^{(p)}T_2[k]P_2 & \dots & \frac{1}{M}\rho^{(p)}T_2[k]P_2 \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{M}\rho^{(p)}T_M[k]P_M & \frac{1}{M}\rho^{(p)}T_M[k]P_M & \dots & I - \frac{M-1}{M}\rho^{(p)}T_M[k]P_M \end{bmatrix}, \tag{A.37}$$

Comme on a montré en §A.6.2, les matrices de triche $T_i[k]$ peuvent change les valeurs propres de la matrice, éventuellement faisant le système osciller et ne pas converger aux valeurs optimales.

11. $I + \frac{M-1}{M}\rho^{(p)}P_i + \sum_{j=1}^{M-1}\frac{1}{M}\rho^{(p)}P_i = I$

Dynamique de $\boldsymbol{\theta}_i$

Similairement, si on fait le contraire et substitue (A.32) en (A.31), ça résulte en

$$\boldsymbol{\theta}_i^{(p+1)} = \boldsymbol{\theta}_i^{(p)} + \rho^{(p)} \left((-P_i \boldsymbol{\theta}_i^{(p)} - \boldsymbol{s}_i[k]) - \frac{1}{M} \sum_{j \in \mathcal{M}} (-P_j \boldsymbol{\theta}_j^{(p)} - \boldsymbol{s}_j[k]) \right), \forall i \in \mathcal{M}. \tag{A.38}$$

ou en forme matricielle

$$\boldsymbol{\theta}^{(p+1)} = \mathcal{A}_\theta \boldsymbol{\theta}^{(p)} + \boldsymbol{B}_\theta[k], \tag{A.39}$$

où

$$\mathcal{A}_\theta = \begin{bmatrix} I - \frac{M-1}{M}\rho^{(p)} P_1 & \frac{1}{M}\rho^{(p)} P_2 & \dots & \frac{1}{M}\rho^{(p)} P_M \\ \frac{1}{M}\rho^{(p)} P_1 & I - \frac{M-1}{M}\rho^{(p)} P_2 & \dots & \frac{1}{M}\rho^{(p)} P_M \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{M}\rho^{(p)} P_1 & \frac{1}{M}\rho^{(p)} P_2 & \dots & I - \frac{M-1}{M}\rho^{(p)} P_M \end{bmatrix} \tag{A.40}$$

$$\boldsymbol{B}_\theta[k] = \begin{bmatrix} -\frac{M-1}{M}\rho^{(p)} \boldsymbol{s}_1[k] + \frac{1}{M}\rho^{(p)} \boldsymbol{s}_2[k] \cdots - \frac{1}{M}\rho^{(p)} \boldsymbol{s}_M[k] \\ \frac{1}{M}\rho^{(p)} \boldsymbol{s}_1[k] - \frac{M-1}{M}\rho^{(p)} \boldsymbol{s}_2[k] \cdots - \frac{1}{M}\rho^{(p)} \boldsymbol{s}_M[k] \\ \vdots \\ \frac{1}{M}\rho^{(p)} \boldsymbol{s}_1[k] + \frac{1}{M}\rho^{(p)} \boldsymbol{s}_2[k] \cdots - \frac{M-1}{M}\rho^{(p)} \boldsymbol{s}_M[k] \end{bmatrix}, \tag{A.41}$$

qui aussi est un système DT, mais avec une entrée forcée ($\boldsymbol{B}_\theta$).

On peut analyser la stabilité de $\mathcal{A}_\theta$ exactement comme fait avec $\mathcal{A}_\lambda$, c'est-à-dire, les valeurs propres doivent être dans le cercle unitaire.

On analyse aussi la somme de ses lignes et colonnes. La somme des colonnes est égal à 1. Alors, 1 est une de ces valeurs propres par la gauche :

$$\mathbf{1}^T \mathcal{A}_\theta = \mathbf{1}^T \tag{A.42}$$

Comme espéré, le système conserve la masse, i.e., les θ_i respectent la contrainte d'égalité globale. De cette façon, la dépendence linéaire entre les θ_i est explicite. Si on analyse les colonnes, on voit que la somme n'est pas 1. Alors, comme nous avons constaté, le système converge mais pas nécessairement à un état où tous les θ_i sont égaux. Cela serait le cas où les P_i sont égaux.

Comme on voit, les valeurs propres dependent just de P_i et $\rho^{(p)}$. Comme les P_i sont des paramètres du système, le designer ne peut que choisir l'évolution de $\rho^{(p)}$.

Une fois le $\rho^{(p)}$ choisi, la matrice $\mathcal{A}_\theta$ évolue toujours de la même manière. Alors, s'il y a un changement de cette évolution, ça veut dire que le système présente un comportement anormale.

Comme on peut observer, une perturbation en P_i (changement de la fonction objectif ou contraintes) change les valeurs de $\mathcal{A}_\theta$, et par conséquence sa dynamique. Ce changement peut déstabiliser le système si les valeurs propres sont décalées dehors le cercle unitaire,

comme déjà montré en §A.6.2. Par contre, les perturbations en $\boldsymbol{s}_i[k]$ (modification de la function objectif, contraintes ou état/référence) peuvent change l'état en régime permanent (changement en $\boldsymbol{B}_\theta[k]$).

Si on suppose que le système est sous attaque et tous les agents sont suspects on a la dynamique

$$\boldsymbol{\theta}^{(p+1)} = \tilde{\mathcal{A}}_\theta[k]\boldsymbol{\theta}^{(p)} + \tilde{\boldsymbol{B}}_\theta[k] \tag{A.43}$$

où

$$\tilde{\mathcal{A}}_\theta[k] = \begin{bmatrix} I - \frac{M-1}{M}\rho^{(p)}T_1[k]P_1 & \frac{1}{M}\rho^{(p)}T_2[k]P_2 & \dots & \frac{1}{M}\rho^{(p)}T_M[k]P_M \\ \frac{1}{M}\rho^{(p)}T_1[k]P_1 & I - \frac{M-1}{M}\rho^{(p)}T_2[k]P_2 & \dots & \frac{1}{M}\rho^{(p)}T_M[k]P_M \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{M}\rho^{(p)}T_1[k]P_1 & \frac{1}{M}\rho^{(p)}T_2[k]P_2 & \dots & I - \frac{M-1}{M}\rho^{(p)}T_M[k]P_M \end{bmatrix} \tag{A.44}$$

$$\tilde{\boldsymbol{B}}_\theta[k] = \begin{bmatrix} -\frac{M-1}{M}\rho^{(p)}T_1[k]\boldsymbol{s}_1[k] + \frac{1}{M}\rho^{(p)}T_2[k]\boldsymbol{s}_2[k] \cdots - \frac{1}{M}\rho^{(p)}T_M[k]\boldsymbol{s}_M[k] \\ \frac{1}{M}\rho^{(p)}T_1[k]\boldsymbol{s}_1[k] - \frac{M-1}{M}\rho^{(p)}T_2[k]\boldsymbol{s}_2[k] \cdots - \frac{1}{M}\rho^{(p)}T_M[k]\boldsymbol{s}_M[k] \\ \vdots \\ \frac{1}{M}\rho^{(p)}T_1[k]\boldsymbol{s}_1[k] + \frac{1}{M}\rho^{(p)}T_2[k]\boldsymbol{s}_2[k] \cdots - \frac{M-1}{M}\rho^{(p)}T_M[k]\boldsymbol{s}_M[k] \end{bmatrix} \tag{A.45}$$

Comme dans le cas avec $\mathcal{A}_\theta$, en ajustant les $T_i[k]$, un attaquant peut changer les valeurs propres de $\mathcal{A}_\lambda$. Les matrices de triche $T_i[k]$ peuvent être aussi ajusté pour pousser la négociation à un nouveau point, mais comme on voit, ça modifie aussi la dynamique do système. Alors, les ataquants doivent faire le compromis entre leur gourmandise et de ne pas casser la négociation, ce qu'est pire pour tous les agents, incluant les attaquants.

Quelqu'un peut remarquer qu'on pourrait créer un détecteur de comportement anormale en supervisant la matrice $\mathcal{A}_\theta$. Une fois qu'il y a un changement d'une de ses valeurs propres, un comportement anormale est détecté.

Malheureusement, cette méthode nous donne que la détection, mais pas forcément quel agent à causé l'anomalie, ce que serait utile pour un algorithm de mitigation. D'ailleurs, pour estimer les éléments de $\mathcal{A}_\theta$, il nous serait nécessaire estimer $M^2(c)^2$ éléments [12]. Exploitant la structure connue de la matrice et des blocs, on pourrait réduire le total au moins $2M(\frac{c+c^2}{2})$ éléments [13].

En suite on verra un mécanisme capable de pas seulement détecter les attaques, mais aussi de identifier l'attaquant, et ça en utilisant la moitié des éléments cités.

12. M^2 blocs de taille c
13. (M diagonal + M hors diagonal) blocs symmetriques de taille c (seulement triangle supérieur)

A.7.5 Détection

Utilisant (A.21) on peut récrire (6.19) comme

$$\tilde{\boldsymbol{\lambda}}_i[k] = -\tilde{P}_i[k]\boldsymbol{\theta}_i[k] - \tilde{\boldsymbol{s}}_i[k], \tag{A.46}$$

où $\tilde{P}_i[k] = T_i[k]P_i$ et $\tilde{\boldsymbol{s}}_i[k] = T_i[k]\boldsymbol{s}_i[k]$.

Une fois que P_i doit est constant, n'importe quel chagement entre deux négociations est une conséquence d'un comportement anormal, dans ce cas causé par $T_i[k]$.

Utilisant la relation entre $\boldsymbol{\theta}_i[k]$ et $\boldsymbol{\lambda}_i[k]$, on peut superviser les échanges entre coordinateur et agents, estimant $\widehat{\tilde{P}}_i[k]$ et $\widehat{\tilde{\boldsymbol{s}}}_i[k]$ tels que

$$\tilde{\boldsymbol{\lambda}}_i[k] = -\tilde{P}_i[k]\boldsymbol{\theta}_i[k] - \tilde{\boldsymbol{s}}_i[k], \tag{A.47}$$

Si on a accès à la valeur nominale de P_i [14], notée $\bar{P}_i$, on peut créer une loi de détection pour chaque agent i. En choisissant une borne arbitraire ϵ_{P_i}, on peut définir l'erreur

$$E_i[k] = \left\| \widehat{\tilde{P}}_i[k] - \bar{P}_i \right\|_F, \tag{A.48}$$

et la fonction de détection d_i :

$$d_i = \mathbb{1}_{\{E_i[k] \geqslant \epsilon_{P_i}\}}, \tag{A.49}$$

qui détecte un attaque quand la borne est surpassée, i.e., $d_i = 1$.

Remarque 1.6

Comme montré en Remarque 1.3, le choix de ϵ_{P_i} peut influencer le taux de faux positifs. La borne doit être supérieure à l'erreur d'estimation pour un scénario sans attaque. La propagation des érreurs n'est pas dans le cadre d'étude de cette thèse.

Remarque 1.7

Si l'estimation ne converge pas, ça veut dire que les rélations ne sont plus linéaires, et alors on pourrait comme ça détecter un comportement anormal.

Une brève discussion pour l'estimation de $\widehat{\tilde{P}}_i[k]$ suit.

A.7.6 Estimation

On choisie la méthode des moindres carrés récursive, en anglais Recursive Least Squares (RLS) [ÅW89], pour estimer les paramètres.

14. Donnée ou estimée à partir de données historiques fiables.

Dans le RLS, pour estimer un paramètre $\boldsymbol{\nu}_i$, on utilise un modèle de régression avec entrées et sorties (B_i et $\boldsymbol{\lambda}_i$) qui nous possibilitent de calculer un résidu ϵ. Ce résidue, allié à un gain Φ, est utilisé pour mettre à jour les estimations de $\boldsymbol{\nu}_i$. Comme la valeur exacte n'est pas connue on utilise une initialisation et un terme d'oubli ϕ pour mettre à jour le gain Φ, et naturellement comme dit le nome, oubliant les événements du passé.

On transforme (A.46) pour utiliser un modèle comme

$$B_i \boldsymbol{\nu}_i = \boldsymbol{\lambda}_i, \tag{A.50}$$

où on vectorise $\widehat{\tilde{P}}_i[k]$ and $\widehat{\tilde{\boldsymbol{s}}}_i[k]$ en $\boldsymbol{\nu}_i$ et change la structure de $\boldsymbol{\theta}_i$ pour créer B_i et $\boldsymbol{\lambda}_i$

$$\boldsymbol{\nu}_i = \begin{bmatrix} \mathrm{vec}(\widehat{\tilde{P}}_i[k]) \\ \widehat{\tilde{\boldsymbol{s}}}_i[k] \end{bmatrix}, \tag{A.51}$$

$$B_i = \begin{bmatrix} I_c \otimes \boldsymbol{\theta}_i^{\mathrm{T}} & I_c \end{bmatrix}. \tag{A.52}$$

Si on veut utiliser le structure symétrique de $\widehat{\tilde{P}}_i[k]$ and on peut estimer juste le triangle supérieur, en réduisant le total d'éléments à $\frac{c^2+3c}{2}$.

On restructure $\boldsymbol{\nu}_i$

$$\boldsymbol{\nu}_i = \begin{bmatrix} \widehat{\tilde{P}}_i[k]^{\mathrm{T}}_{(1,1:c)} \\ \widehat{\tilde{P}}_i[k]^{\mathrm{T}}_{(2,2:c)} \\ \vdots \\ \widehat{\tilde{P}}_i[k]^{\mathrm{T}}_{(c,c)} \\ \widehat{\tilde{\boldsymbol{s}}}_i[k] \end{bmatrix}. \tag{A.53}$$

Cependant, pour B_i, on utilise un algorithme de construction (Algorithme A.2) :

$$B_i = \begin{bmatrix} \texttt{structDataSym}(\boldsymbol{\theta}_i) & I_c \end{bmatrix}. \tag{A.54}$$

Algorithm A.2: Restructuration de forme symétrique.

```
structDataSym (x)
    s ← #x
    si s == 1 alors
        Y ← x
    sinon
        Y ← [ [ [0_{s-1}   x_(1) I_{s-1}] ]   0^T_{(s^2-s)/2}
              [ structDataSym(x_(2:s))    ] ]
```

Pour mieux comprendre on donne un example. Si $\boldsymbol{\theta}_i$ a $c = 3$ éléments, en appliquant

`structDataSym`, ça résulte en

$$\texttt{structDataSym}(\theta_i) = \begin{bmatrix} \theta_{i(1)} & \theta_{i(2)} & \theta_{i(3)} & 0 & 0 & 0 \\ 0 & \theta_{i(1)} & 0 & \theta_{i(2)} & \theta_{i(3)} & 0 \\ 0 & 0 & \theta_{i(1)} & 0 & \theta_{i(2)} & \theta_{i(3)} \end{bmatrix}. \tag{A.55}$$

Remarque 1.8

On ne nécessite pas de $\boldsymbol{s}_i[k]$ pour la estimation, on l'estime quand même car on l'utilisera pour la mitigation.

On décrit en Algorithme A.3 comme calculer une estimation de $\boldsymbol{\nu}_i$.

Algorithm A.3: Update of RLS to estimate $\hat{\tilde{P}}_i[k]$ and $\hat{\tilde{\boldsymbol{s}}}_i[k]$ simultaneously.

Entrées: Pas h, Oubli ϕ, Estimation ${\boldsymbol{\nu}_i}^{(h)}$, Gain $\Phi^{(h)}$, Entrée $\boldsymbol{\theta}_i^{(h)}$, Sortie ${\boldsymbol{\lambda}_i}^{(h)}$
Sorties: Nouvelle estimation ${\boldsymbol{\nu}_i}^{(h+1)}$, Nouveau Gain $\Phi^{(h+1)}$
$B_i \leftarrow \left[\texttt{structDataSym}(\boldsymbol{\theta}_i^{(h)}), I_c\right]$
Calculate residual $\epsilon : \epsilon \leftarrow \boldsymbol{\lambda}_i - B_i\boldsymbol{\nu}_i$
Update gain $\Phi : \Phi^{(h+1)} \leftarrow \phi^{-1}\Phi^{(h)} - \phi^{-2}\Phi^{(h)}{B_i}^{\mathrm{T}}(I_c + \phi^{-1}B_i\Phi^{(h)}{B_i}^{\mathrm{T}})^{-1}B_i\Phi^{(h)}$
Calculate new estimate $\boldsymbol{\nu}_i : {\boldsymbol{\nu}_i}^{(h+1)} \leftarrow {\boldsymbol{\nu}_i}^{(h)} + \Phi^{(h+1)}{B_i}^{\mathrm{T}}\epsilon$

On pourrait essayer d'utiliser l'estimation pendant la négociation, mais l'agorithme ne marcherait pas. Le fait que $\boldsymbol{\lambda}_i$ dépend de $\boldsymbol{\theta}_i$ et vice-versa fait que la négociation se comporte comme un système DT (comme vu en §A.7.4). Cette dépendance diminue l'excitation de l'algorithme et comme vu en [ÅW89], ça fait que la solution devienne pas unique, une des matrices devienne singulière. Une façon d'augmenter artificiellement l'excitation est de déconecter les $\boldsymbol{\theta}_i$ de la négociation et utiliser un autre type d'entrée.

On a opté d'utiliser un bruit comme entrée une foi que comme vu en [ÅW89, §3.4], le signal aléatoire a excitation persistent de n'importe quel ordre.

A.7.7 Technique de détection complète

La détection peut être décrite enfin par l'Algorithme A.4.

Une fois les attaquants identifiés, on a besoin de la mitigation pour réduire les effets.

Algorithm A.4: Algorithme de détection dans un pas k (en parallèle si possible).

Entrées: $\bar{P}_i$ Nominal, $h_{\max}$, Borne $\epsilon_{\boldsymbol{\nu}_i}$, Borne ϵ_{P_i},
Sorties: Détection d_i, Estimations de $\widehat{\bar{P}}_i[k]$ et $\widehat{\bar{\boldsymbol{s}}}_i[k]$
$h \leftarrow 0$
Initialise ${\boldsymbol{\nu}_i}^{(p)}$
répéter
  Coordinateur envoie $\boldsymbol{\theta}_i^{(h)}$ aléatoire à agent i
  Sous-système résout (A.19a), et envoie $\boldsymbol{\lambda}_i^\star[k](\boldsymbol{\theta}_i^{(p)})$ en retour
  Coordinateur met à jour l'estimation de $\boldsymbol{\nu}_i$ avec Algorithme 6.2
  si $[h \geqslant h_{\max}]$ **alors**
    Coordinateut fixe d_i : $d_i \leftarrow 1$
    break
  $h \leftarrow h + 1$
jusqu'à $\left\|{\boldsymbol{\nu}_i}^{(h+1)} - {\boldsymbol{\nu}_i}^{(h)}\right\| \leqslant \epsilon_{\boldsymbol{\nu}_i}$
Reconstruit $\widehat{\bar{P}}_i[k]$ et $\widehat{\bar{\boldsymbol{s}}}_i[k]$ à partir de $\boldsymbol{\nu}_i$
$E_i[k] \leftarrow \left\|\widehat{\bar{P}}_i[k] - \bar{P}_i\right\|_F$
Coordinateur calcule d_i : $d_i \leftarrow d_i \vee \mathbb{1}_{\{E_i[k] \geqslant \epsilon_{P_i}\}}$

A.7.8 Mitigation

Pour la mitigation on a quelques possibilités d'objectifs. On donne un petit exemple.

Exemple Qualitatif

Imaginons un système simple de deux sous-systèmes Linear Time-Invariant Discrete Time (LTIDT) d'une dimension, contraints par l'entrée, controllés par la dMPC presentée dans ce travail, problèmes du type QP et toute autre condition imposé dans cette section. On utilise un horizon de prédiction $N = 1$ and contraintes locales $\boldsymbol{u}_i \geqslant \mathbf{0}$ pour faciliter la visualisation.

Si on trace les courbes de niveaux du système (Fig. A.15), on voit le minimum sans contraintes projeté (projection pondérée) sur le hyperplan. La projection résulte la solution sous contraintes (point bleu).

Une fois qu'un agent attaque le système (A.21), les courbes sont déformées (Fig. A.16). Cette distortion change la projection, resultant en une nouvelle solution (en vert).

Si l'attaquant est détecté, une option est de mitiger la dérive par substituant le $\tilde{\boldsymbol{\lambda}}_i$ réçu par $\mathbf{0}$. Cela peut être interpreté comme un coordinateur punissant l'attaquant, en supposant que l'attaquant est satisfait $\boldsymbol{\lambda}_i = \mathbf{0}$, indépendamment de l'allocation faite. Cette négligence vers l'attaquant résulte en lui recevant moins et moins de ressources. À la limite, ça serait comme si l'attaquant ne faisait plus part de la négociation, en étant débranché, comme en [Vel+18 ; Mae+21].

Si on le fait (Fig. 6.3) on voit les nouvelles courbes et la nouvelle solution (en orange).

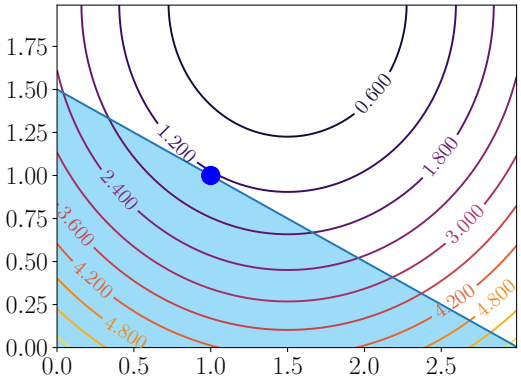


FIGURE A.15 – Minimum Original.

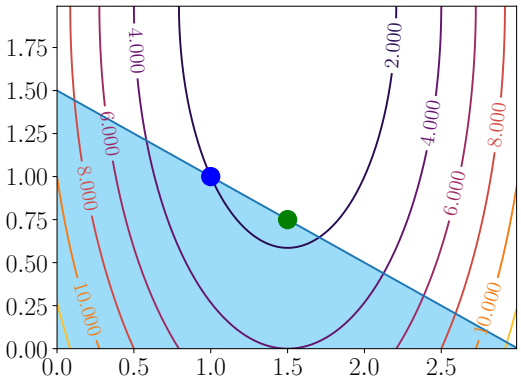


FIGURE A.16 – Minimum après attaque.

Même si l'idée de punir l'attaquant semble attirante, comme nous controllons un CPS ça peut affecter la qualité de service, en anglais Quality of Service (QoS), et par conséquence les personnes.

Une solution serait d'allouer des ressources suffisantes pour un niveau minimal de QoS, mais ça peut être un peu difficile à cause de sa subjectivité.

Nous proposons une autre solution, essayer de récuperer le comportement normal, tant que quelques hypothèses soient garanties. Cela permet de passer des courbes du système attaqué (Fig. A.16) à des autres, comme montré en Fig. A.18, où le nouveau minimum s'encontre dans une voisinage acceptable de l'original.

Récuperant comportement nominal

Comme vu, si $\boldsymbol{\lambda}_i = \mathbf{0}$, ça veut dire que l'allocation de l'agent lui satisfait. Alors, on n'espère pas que l'attaquant va tricher pour dire qu'il est satisfait quand il n'y est pas.

De cette façon, on assume que

$$\left[\tilde{\boldsymbol{\lambda}}_i = \mathbf{0}\right] \Leftrightarrow \left[\boldsymbol{\lambda}_i = \mathbf{0}\right], \tag{A.56}$$

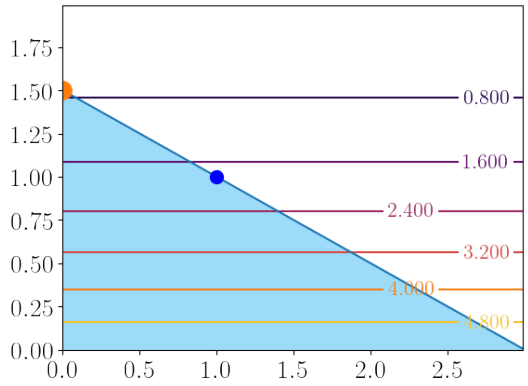


FIGURE A.17 – Nouveau optimal quand négligent l'attaquant.

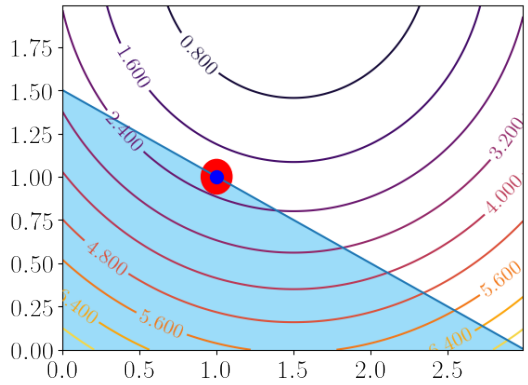


FIGURE A.18 – Valeur optimale après récuperer comportement original.

qui signifie que

$$\tilde{\boldsymbol{\lambda}}_i = T_i[k]\boldsymbol{\lambda}_i = \mathbf{0}, \text{ if and only if } \boldsymbol{\lambda}_i = \mathbf{0}, \tag{A.57}$$

impliquant que $T_i[k]$ est **inversible**.

En observant (**??**), on suppose aussi que le choix de $T_i[k]$ ne change pas le structure de P_i, i.e., $\tilde{P}_i[k]$ est aussi symétrique.

Si on a l'estimation $\hat{\tilde{P}}_i[k]$ et la valeur nominale $\bar{P}_i$, avec ces hypothèses on peut estimer l'inverse de $T_i[k]$:

$$\widehat{T_i[k]^{-1}} = \bar{P}_i \hat{\tilde{P}}_i[k]^{-1}. \tag{A.58}$$

Et avec cette estimation on peut reconstruire $\boldsymbol{\lambda}_i$:

$$\overset{\text{rec}}{\boldsymbol{\lambda}}_i = -\bar{P}_i\boldsymbol{\theta}_i - \widehat{T_i[k]^{-1}}\hat{\tilde{\mathbf{s}}}_i[k] \tag{A.59}$$

Alors, on peut moduler la négociation en choisissant la version de $\boldsymbol{\lambda}_i$ qu'on veut, i.e.,

$$\overset{\text{mod}}{\boldsymbol{\lambda}}_i = \begin{cases} \tilde{\boldsymbol{\lambda}}_i, & \text{if } d_i = 0 \\ \overset{\text{rec}}{\boldsymbol{\lambda}}_i, & \text{if } d_i = 1 \end{cases} \tag{A.60}$$

Remarque 1.9

Observe que si l'estimation ne converge pas on ne peut pas réconstruire $\boldsymbol{\lambda}_i[k]$. Une solution serait ignorer l'attaquant comme montré. Par contre, on ignore les cas où l'estimation ne converge pas.

A.7.9 La dMPC résiliente pour systèmes dépourvus

En mettant ensemble les méthodes de détection et mitigation présentées, on peut finalement avoir notre dMPC résiliente basée sur la décomposition primale pour systèmes dépourvus, ou en anglais Resilient Primal Decomposition-based dMPC for deprived systems (RPdMPC-DS). On peut observer que les mécanismes présentés peuvent être vus comme un superviseur ajouté pour chaque agent (Fig. A.19). Algorithme 6.4 systematize la méthode.

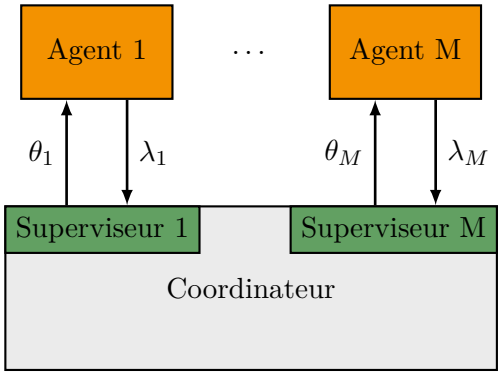


FIGURE A.19 – Échange entre agents et coordinateur dans la RPdMPC-DS.

Algorithm A.5: La dMPC Résiliente basée sur la décomposition primale pour systèmes dépourvus.

Phase de Détection:

Coordinateur compute d_i et estimations de $\hat{\tilde{P}}_i[k]$ avec Algorithme A.4

Phase de Négociation:

$p \leftarrow 0$

Coordinateur initialise $\boldsymbol{\theta}^{(p)}$ et les envoie aux sous-systèmes

répéter

Sous-systèmes résolvent (A.19a), et envoient $\boldsymbol{\lambda}_i^\star(\boldsymbol{\theta}^{(p)})$

Coordinateur m-à-j les allocations (A.31) avec versions adéquates de $\boldsymbol{\lambda}_i$: $\boldsymbol{\lambda}_i^\star(\boldsymbol{\theta}^{(p)})$, si $d_i = 0$ et $\overset{\text{rec}}{\boldsymbol{\lambda}}_i$, si $d_i = 1$ (A.59)

$p \leftarrow p + 1$

jusqu'à $\left[\left\|\boldsymbol{\theta}[k]^{(p)} - \boldsymbol{\theta}[k]^{(p-1)}\right\| \leqslant \epsilon\right] \vee [p \geqslant p_{\max}]$

Agents appliquent la dernière commande $\boldsymbol{u}_i^\star[0|k]$ calculée

$k \leftarrow k + 1$

A.7.10 Expériment Numérique

Étude de cas

Notre étude de cas est un simple DHN composé par $M = 4$ petites maisons (appelées I, II, II and IV). Chacune a un état $\boldsymbol{x}_i$, et une référence de temperature différente $\boldsymbol{w}_i(t)$ (besoins diveses). Les maisons sont equipées de convecteurs, et la puissance total de chauffage de chaque maison est l'entrée $\boldsymbol{u}_i(t)$. Comme il n'y a pas de dispositifs pour rafraîchir l'ambient, les entrées sont positives, i.e., $\boldsymbol{u}_i[k] \succeq \mathbf{0}$. En Fig. 6.6 les maisons consomment la puissance d'un fournisseur.

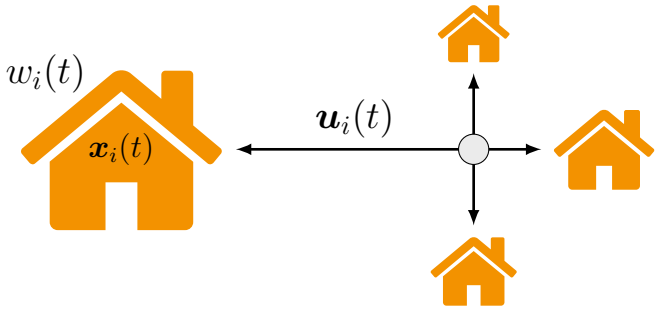


FIGURE A.20 – District avec 4 maisons.

On modèle des maisons utilisant des modèles 3R-2C monozone ave des fenêtres [GDU02] (Fig. A.21). Les paramètres sont décrits sur Tab. A.2 et A.3.

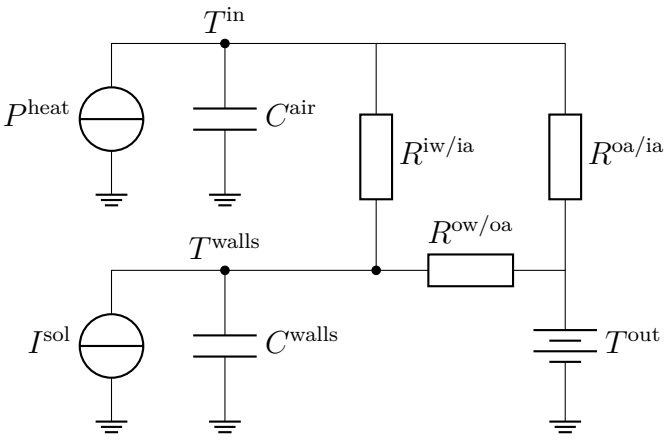


FIGURE A.21 – Modèle thermique 3R-2C d'une maison.

TABLE A.2 – Paramètres du Modèle

Symbole	Signification
C_i^{air}	Capacité termique de l'air intérieur
C_i^{walls}	Capacité termique des parois externes
$R_i^{\text{iw/ia}}$	Resist. entre l'air et parois internes
$R_i^{\text{ow/oa}}$	Resist. entre l'air et parois externes
$R_i^{\text{oa/ia}}$	Resist. entre l'air intérieur et extérieur (à travers des fenêtres)
P^{heat}	Puissance de chauffage des convecteurs
I^{sol}	Puissance irradiée par le Soleil
T^{in}	Temperature moyenne de l'air à l'intérieur
T^{walls}	Temperature moyenne des parois internes

Le circuit peut être traduit par le modèle d'espace d'état

$$\begin{array}{rcl} \dot{\boldsymbol{x}}_i(t) & = & A_{ci}\boldsymbol{x}_i(t) \quad + \quad B_{ci}\boldsymbol{u}_i(t) + D_{ci}\boldsymbol{d}_i(t) \\ \boldsymbol{y}_i(t) & = & C_{ci}\boldsymbol{x}_i(t) \end{array}, \tag{A.61}$$

où

$$A_{\text{c}_i} = \begin{bmatrix} -\frac{1}{C_i^{\text{walls}}R_i^{\text{oa/ia}}} - \frac{1}{C_i^{\text{walls}}R_i^{\text{iw/ia}}} & \frac{1}{C_i^{\text{walls}}R_i^{\text{iw/ia}}} \\ \frac{1}{C_i^{\text{air}}R_i^{\text{iw/ia}}} & -\frac{1}{C_i^{\text{air}}R^{\text{ow/oa}}o_i} - \frac{1}{C_i^{\text{air}}R_i^{\text{iw/ia}}} \end{bmatrix} \quad B_{\text{c}_i} = \begin{bmatrix} \frac{1}{C_i^{\text{air}}} \\ 0 \end{bmatrix}^{\text{T}}$$

$$D_{\text{c}_i} = \begin{bmatrix} 0 & \frac{1}{C_i^{\text{air}}R_i^{\text{ow/oa}}} \\ \frac{1}{C_i^{\text{air}}} & \frac{1}{C_i^{\text{air}}R^{\text{ow/oa}}} \end{bmatrix} \qquad C_{\text{c}_i} = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{A.62}$$

avec $\boldsymbol{x}_i(t) = \left[T_i^{\text{in}}(t)\ T_i^{\text{walls}}(t)\right]^{\text{T}}$, $\boldsymbol{u}_i(t) = P_i^{\text{heat}}$, $\boldsymbol{d}_i(t) = \left[I^{\text{sol}}(t)\ T^{\text{out}}(t)\right]^{\text{T}}$.

Les entrées $\boldsymbol{u}_i(t)$ sont contraintes par une puissance de chauffage maximale disponibilisée par le founisseur, i.e.,

$$\sum_{i\in\mathcal{M}} \left[\boldsymbol{u}_i(t)\right] \preceq \boldsymbol{u}_{\max} = 40\text{kW}. \tag{A.63}$$

TABLE A.3 – Paramètres pour chaque maison

Élément	I	II	III	IV	Unit
C^{walls}	8	7	9	6	10^4J/K
C^{air}	5.0	4.0	4.5	4.7	10^4J/K
$R^{\text{oa/ia}}$	5	6	4	5	10^{-3}K/W
$R^{\text{iw/ia}}$	2.5	2.3	2.0	2.2	10^{-4}K/W
$R^{\text{ow/oa}}$	0.5	1.0	0.8	0.9	10^{-4}K/W

Remarque 1.10

On ignore les perturbation causées par le soleil et l'éxterieur, i.e., $\boldsymbol{d}_i(t) = \mathbf{0}$.

Comme on utilise la MPC, on discretise le système avec un bloqueur d'ordre zéro et $T_s = 0.25$h comme pas d'échantillonnage, resultant en les 3-tuples (A_i, B_i, C_i).

Appliquant le RPdMPC-DS

D'abord, on suppose que le système est dépourvu, c'est à dire, ayant des états initiaux $\boldsymbol{x}_i[0]$, les références $\boldsymbol{w}_i[k]$ ne peuvent pas être achevées.

On utilise par exemple, les états $\boldsymbol{x}_{\mathrm{I}}[0] = [18.3 \quad 3.]^{\mathrm{T}}$, $\boldsymbol{x}_{\mathrm{II}}[0] = [19.6 \quad 5.9]^{\mathrm{T}}$, $\boldsymbol{x}_{\mathrm{III}}[0] = [18.4 \quad 5.3]^{\mathrm{T}}$, and $\boldsymbol{x}_{\mathrm{IV}}[0] = [17.4 \quad 5.3]^{\mathrm{T}}$, et références $w_{\mathrm{I}}[0] = 20$, $w_{\mathrm{II}}[0] = 20$, $w_{\mathrm{III}}[0] = 20$, and $w_{\mathrm{IV}}[0] = 20$. Utilisant les gains $Q_i = 10I$, et $R_i = I_2$, et en choisissant de contrôler le système avec un horizon de prédiction $N = 4$, on structure le problème de la dMPC et le décompose avec la décomposition primale, avec des problèmes locaux

$$\begin{aligned} \underset{\boldsymbol{U}_i[k]}{\text{minimize}} \quad & J_i = \tfrac{1}{2} \left\| \boldsymbol{U}_i[k] \right\|_{H_i}^2 + \boldsymbol{f}_i[k]^T \boldsymbol{U}_i[k] \\ \text{subject to} \quad & \bar{\Gamma}_i \boldsymbol{U}_i[k] = \boldsymbol{\theta}_i[k] : \boldsymbol{\lambda}_i[k] \\ & \boldsymbol{U}_i[k] \geq \mathbf{0} \end{aligned} \tag{A.64}$$

et équations de mise à jour (A.31).

Résultats

On teste l'algorithme avec une simulation dans une période de 5 heures, i.e., $k \in \{0, 5/T_s\}$, dans quelques scénarios différents :

1. Comportement Nominal (dénoté N)
2. Agent I attaque le système commandé par dMPC standard (dénoté S)
3. Agent I attaque le système commandé par RPdMPC-DS (dénoté C)

L'attaque choisi pour les scénarios 2 et 3, sont

$$T_I = \begin{cases} \left[\begin{smallmatrix} 4 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 4 \end{smallmatrix}\right], & \text{si } k > 5 \\ I_c, & \text{sinon} \end{cases}. \tag{A.65}$$

Dans Fig. A.22, on compare la temperature de la maison I ($\boldsymbol{y}_I[k]$) avec sa référence $\boldsymbol{w}_I[k]$, et après la variable de détection $E_I(k)$ avec sa borne $\epsilon_P = 10^{-4}$.

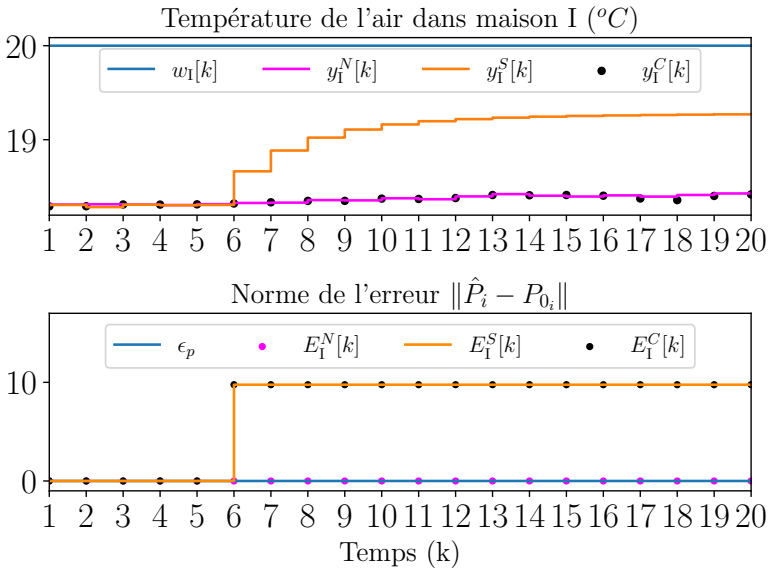


FIGURE A.22 – Température de la maison I et la variable $E_I(k)$ pour les 3 scénarios.

Comme attendu, les variables de détection repose sous la borne $\epsilon_P = 10^{-4}$ quand on est dans le scénario nominal. Par contre, pour les scénarios S et C, où l'agent attaque le système, la variable dépasse la borne, indiquant la triche. En scénario S, on observe l'erreur de suivi $w_I - y_I$ se réduire quand l'attaque est activé, suggérant que l'attaquant pousse la négocitation pour recevoir plus de ressources. D'ailleurs, les autres agents reçoivent moins de ressources, et leur erreur de suivi augmente (Fig. A.23).

Quand la RPdMPC-DS est activée en scénario C, les temperatures récupèrent un comportement proche du nominal, comme on voit dans Fig. A.22 et A.23.

On peut aussi comparer les fonctions objectif locales et global J_i^{acc} et J^{acc} dans Tab. A.4. Comme attendue, la fonction decroît pour l'attaqueur si la dMPC n'est pas sécurisée. Cependant une fois que la RPdMPC-DS est activé les objectifs reviennent à une valeur proche de l'originale.

TABLE A.4 – Fonctions objectif J_i (% erreur)

Agent	Scénario N	Scénario S	Scénario C
I	299.5	190.8 (−36.3)	301.0 (0.0)
II	192.4	234.1 (21.7)	191.4 (−0.5)
III	305.9	359.1 (17.4)	305.9 (−0.0)
IV	297.5	349.9 (17.6)	297.2 (−0.1)
Global	1095.3	1133.9 (3.5)	1095.5 (0.0)

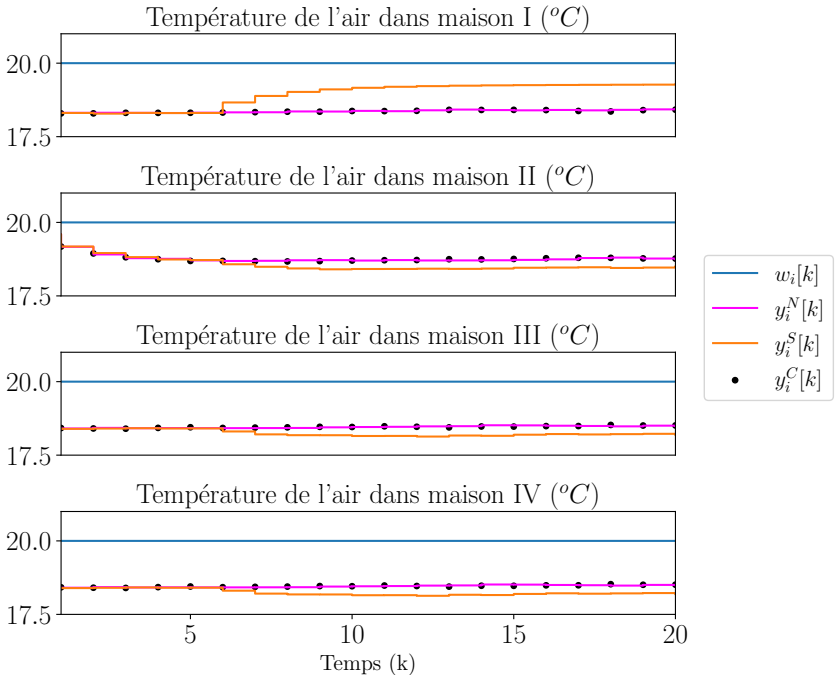


FIGURE A.23 – Temperature de l'air dans toutes les maisons pour les 3 scénarios.

A.8 Commande Prédictive résiliente sur pénurie artificielle

A.8.1 Relaxant le problème

On reprend le problème (A.14) et on ajoute un ensemble de contraintes décomposable $\mathcal{U}$:

$$\begin{array}{ll} \underset{\boldsymbol{U}[k]}{\text{minimize}} & \frac{1}{2}\left\|\boldsymbol{U}[k]\right\|_H^2 + \boldsymbol{f}[k]^T\boldsymbol{U}[k] \\ \text{subject to} & \bar{\Gamma}\boldsymbol{U}[k] \preceq \boldsymbol{U}_{\max} \\ & \boldsymbol{U}[k] \in \mathcal{U} \end{array} . \tag{A.66}$$

On décompose le problème avec la décomposition primale et on a les problèmes locaux :

$$\bar{J}_i^\star(\boldsymbol{\theta}_i[k])[k] = \begin{array}{ll} \underset{\boldsymbol{U}_i[k]}{\text{minimize}} & \frac{1}{2}\left\|\boldsymbol{U}_i[k]\right\|_{H_i}^2 + \boldsymbol{f}_i[k]^T\boldsymbol{U}_i[k] \\ \text{subject to} & \bar{\Gamma}_i\boldsymbol{U}_i[k] \preceq \boldsymbol{\theta}_i[k] : \boldsymbol{\lambda}_i[k] \\ & \boldsymbol{U}_i[k] \in \mathcal{U}_i \end{array} , \tag{A.67}$$

et on résout avec le sous gradient projeté :

$$\boldsymbol{\theta}[k]^{(p+1)} = \text{Proj}^{\mathcal{S}}(\boldsymbol{\theta}[k]^{(p)} + \rho^{(p)}\boldsymbol{\lambda}[k]^{(p)}), \tag{A.20}$$

mais maintenant l'ensemble de la projection est $\mathcal{S} = \{\boldsymbol{\theta}[k] \mid I_c^M\boldsymbol{\theta}[k] \preceq \boldsymbol{U}_{\max}\}$.

Dans cette section, on assume encore que la contrainte originale (A.2) a au maximum la même quantité de ligne que des colonnes et forment des cônes. Par contre, on relaxe une hypothèse faite en §A.7.2. Les systèmes ne sont plus nécessairement dépourvus, i.e., les solutions sans contraintes $\mathring{\boldsymbol{U}}^\star = -H^{-1}\boldsymbol{f}[k]$ et $\mathring{\boldsymbol{U}}_i^\star[k] = -H_i^{-1}\boldsymbol{f}_i[k]$ peuvent ou pas respecter les contraintes.

Comme on verra, cette relaxation fait le problème plus générale mais au même moment exponentielment plus complexe.

A.8.2 L'impacte sur la solution des problèmes locaux

On peut faire la méme analyse que dans la dernière section.

Même si on a des problèmes QP avec des contraintes d'inégalité, on peut encore utiliser la même méthode Lagrangienne pour trouver une solution analytique.

Par contre, les variables duales qui correspondent aux contraintes d'inégalité doivent être positives [BV04]. En utilisant les conditions de complémentarité, il est possible de vérifier que $\boldsymbol{\lambda}_i[k]$ est 0 (ou différent de 0) dépendant du statuts de chaque contrainte pour la solution sans contrainte $\mathring{\boldsymbol{U}}_i^\star[k]$.

Alors, la solution complète de $\boldsymbol{\lambda}_i[k]$ dépendra de la permutation des statuts des contraintes. Et si on a n_{ineq} contraintes, on peut avoir potentiellement $2^{n_{\text{ineq}}}$ permutations.

Remarque 1.11

On dit potentiellement, car ça peut dépendre de la forme des contraintes, comme on verra en §A.8.2.

On voit que la solution de (A.67) dépend du statut des contraintes, qui dépendent de la valeur de $\boldsymbol{\theta}_i[k]$. De cette façon, on remarque que l'ensemble des contraintes divise l'espace de $\boldsymbol{\theta}_i[k]$ en plusieurs régions dénotées $\mathcal{R}^n_{\boldsymbol{\lambda}_i}$, ou chaque région répresente la combinaison de contraintes actives ou pas pour $\mathring{\boldsymbol{U}}^\star_i[k]$.

Les régions $\mathcal{R}^n_{\boldsymbol{\lambda}_i}$ sont des polytopes définis par $\mathcal{R}^n_{\boldsymbol{\lambda}_i} = \{\boldsymbol{x} \mid G^n[k]\boldsymbol{x} \leq b^n[k]\}$ où les hyperplans peuvent varier avec le temps k. Comme remarqué en [Bem+02], les polytopes ne se chevauchent pas, i.e., $\mathcal{R}^i_{\boldsymbol{\lambda}_i} \cap \mathcal{R}^j_{\boldsymbol{\lambda}_i} = \varnothing, \forall i \neq j$. Alors, les régions forment une partition de l'espace de $\boldsymbol{\theta}_i$, i.e., si on a $\boldsymbol{\theta}_i[k] \in \mathcal{Y} \subseteq \mathbb{R}^c$, alors $\bigcup_{i \in \{0:n_{\text{ineq}}-1\}} \mathcal{R}^i = \mathcal{Y}$.

Remarque 1.12

La détermination des polytopes n'est pas dans le scope de ce travail, mais peut être trouvé par des méthodes en [LB19, §4.1.3.2].

Par exemple si on 2 contraintes l'espace de $\boldsymbol{\theta}_i$ pourrait être particioné comme en Fig. A.24 (on signalise si les valeurs des variables duales sont égaux à zéro ou pas).

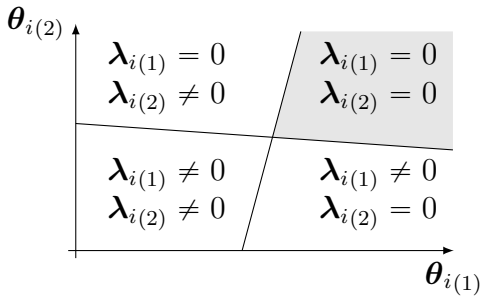


FIGURE A.24 – Deux contraintes qui partionnent l'espace de $\boldsymbol{\theta}_i$.

Pour n'importe quel $\boldsymbol{\theta}_i$ dans l'aire grise (l'intersection des hyperplans), la solution de (A.67) est sans contrainte, i.e., les contraintes seront inactives. Et alors, par définition, le $\boldsymbol{\lambda}_i = \boldsymbol{0}$. Pour les autres cas, au moins une des contraintes sera actives, et on peut enlever les inactives. Si toutes les contraintes sont actives, le résultat est le même que pour le systèmes dépourvus (A.32) :

$$\boldsymbol{\lambda}_i[k] = -P_i\boldsymbol{\theta}_i[k] - \boldsymbol{s}_i[k], \tag{A.32}$$

Si on enleve un ensemble de contraintes, on peut résoudre un problème similaire mais avec un nombre réduit de variables. Par exemple, si on suppose que la première contrainte est inactive. On fait que $\boldsymbol{\lambda}_{i(1)} = 0$ et on le met à côté, on enlève les lignes correspondant en $\bar{\Gamma}_i$ et $\boldsymbol{\theta}_i$, et après on résout pour les éléments restant de $\boldsymbol{\lambda}_i$. Le résultat est

$$\boldsymbol{\lambda}_{i(2:\text{end})}[k] = -\overset{2:c}{P}_i \boldsymbol{\theta}_{i(2:\text{end})}[k] - \overset{2:c}{\boldsymbol{s}}_i[k], \tag{A.68}$$

où $\overset{2:c}{P}_i = (\bar{\Gamma}_{i(2:\text{end},\star)} H_i^{-1} (\bar{\Gamma}_{i(2:\text{end},\star)})^T)^{-1}$ et $\overset{2:c}{\boldsymbol{s}}_i[k] = \overset{2:c}{P}_i \bar{\Gamma}_{i(2:\text{end},\star)} H_i^{-1} \boldsymbol{f}_i[k]$.

On peut faire pareil pour toute autre partition, c'est à dire, toutes les permutations possibles de contraintes actives et inactives. Ce résultat est la base pour calculer la solution du problème primal utilisé en [Bem+02 ; AB09], qui est la base de la commande MPC explicite.

Cependant, les équations auront de tailles d'éléments différentes (autant d'éléments que de contraintes actives). Pour égaliser la taille des équations, on récupere les éléments de $\boldsymbol{\lambda}_i$ mises à zéro et mises à côté. Si on réprend l'exemple, (A.68) serait récrite comme

$$\boldsymbol{\lambda}_i[k] = \begin{bmatrix} 0 \\ \boldsymbol{\lambda}_{i(2:\text{end})}[k] \end{bmatrix} = - \begin{bmatrix} 0 & 0 \\ 0 & \overset{2:c}{P}_i \end{bmatrix} \boldsymbol{\theta}_i[k] - \begin{bmatrix} 0 \\ \overset{2:c}{\boldsymbol{s}}_i[k] \end{bmatrix} \tag{A.69}$$

Afin de faciliter, on utilise une représentation binaire tel que si on a n_{ineq} contraintes, on utilise n_{ineq} chiffres binaires pour les représenter. On marque 1 si la contrainte est inactive et 0 sinon.

Dans la Fig. A.24, $00_{(2)}$ représente le quadrant inférieur gauche, $01_{(2)}$ et $10_{(2)}$ les quadrant inférieur droit et supérieur gauche, respectivement, et $11_{(2)}$ l'aire en gris. On utilise la base 10 pour codifier le chiffre. Alors (A.69) est récrit comme

$$\boldsymbol{\lambda}_i[k] = -P_i^{(2^{n_{\text{ineq}}-1})} \boldsymbol{\theta}_i[k] - \boldsymbol{s}_i[k]^{(2^{n_{\text{ineq}}-1})}, \tag{A.70}$$

ou l'index $(2^{n_{\text{ineq}}-1})$ indique que juste la première contrainte est inactive.

On peut alors, écrire la solution complète comme une fonction affine par morceaux, ou en anglais Piecewise Affine (PWA) function :

$$\boldsymbol{\lambda}_i[k] = \begin{cases} -P_i^{(0)} \boldsymbol{\theta}_i[k] - \boldsymbol{s}_i^{(0)}[k], & \text{si } \boldsymbol{\theta}_i[k] \in \mathcal{R}_{\boldsymbol{\lambda}_i}^0 \\ \vdots & \vdots \\ -P_i^{(2^{n_{\text{ineq}}}-1)} \boldsymbol{\theta}_i[k] - \boldsymbol{s}_i^{(2^{n_{\text{ineq}}}-1)}[k], & \text{si } \boldsymbol{\theta}_i[k] \in \mathcal{R}_{\boldsymbol{\lambda}_i}^{2^{n_{\text{ineq}}}-1} \end{cases}. \tag{A.71}$$

Avec cette équation on peut envisager le changement causé par la relaxation, où on avait juse une équation liant $\boldsymbol{\lambda}_i$ à $\boldsymbol{\theta}_i$, maintenant on a potentiellement $2^{n_{\text{ineq}}}$ équations différentes.

Remarque 1.13

Comme attendu, par définition, $P_i^{(n)}$ et $\boldsymbol{s}_i^{(n)}[k]$ sont de plus en plus creuses. Quand n augmente, plus en plus contraintes deviennent inactive, et plus de blocs sont mises à zéro et finalement on a $P_i^{(2^{n_{\text{ineq}}}-1)} = 0_c$ et $\boldsymbol{s}_i^{(2^{n_{\text{ineq}}}-1)}[k] = \mathbf{0}_c$.

Considérations sur le statut des contraintes

Il est important de constater que, comme dit, pour quelques ensembles de contraintes le numéro de permutations n'est pas forcément $2^{n_{\text{ineq}}}$.

On donne quelques examples. Il y a des cas où il n'existe pas une région où toutes les contraintes sont inactives, i.e., l'intersection des demi-espaces est nulle (voir Fig. A.25). Ces problèmes sont appelés infaisable, et alors sont ignorés dans ce travail.

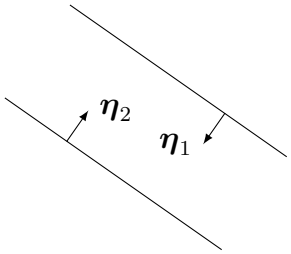


FIGURE A.25 – Ensemble de contraintes avec intersection nulle.

Dans autres cas, il n'y a pas de région où toutes les contraintes sont inactives.

Par exemple, pour des contraintes avec vecteurs normaux $\boldsymbol{\eta}_i$, si l'angle des demi-espaces $\langle_{\boldsymbol{\eta}_i}^{\boldsymbol{\eta}_j}$ est 180^o et l'intersection n'est pas nulle (voir Fig. A.26), au moins une des contraintes sera toujour active.

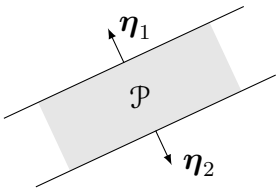


FIGURE A.26 – Deux contraintes où $\langle_{\boldsymbol{\eta}_1}^{\boldsymbol{\eta}_2} = 180^o$.

Encore autre exemple c'est quand les contraintes forment un polyèdre. On présent l'exemple minimal, le simplex en $\mathbb{R}^2$ (voir Fig. A.27). Dans ce cas, on aura toujours au moins 1 contrainte inactive.

Comme on suppose d'avoir au plus la même quantité de contraintes que de dimensions, il n'est possible de créer de polyèdres [15]. Par contre, pour dimensions supérieures, les prismes doivent aussi être pris en compte.

15. Pour $\mathbb{R}^n$, le simplex a $n+1$ faces

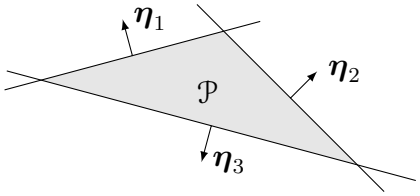


FIGURE A.27 – Un polyèdre de 3 faces.

A.8.3 L'impact sur la négociation

On utilise la même techniques qu'en §A.7.4, de substituer une équation dans l'autre. Ainsi on contemplera la vraie complexité du problème.

La première difficulté est la méthode du sous gradient projeté. À cause de $\mathcal{S}$ être formé par l'intersection de demi-espaces, la solution est trouvée par la défintion de la projection Euclidienne, qui est aussi un problème QP

$$\text{Proj}^{\mathcal{S}}(\boldsymbol{\theta}[k]^{(p)} + \rho^{(p)}\boldsymbol{\lambda}[k]^{(p)}) = \underset{\boldsymbol{x}}{\text{argmin}} \left\{ \begin{array}{ll} \underset{\boldsymbol{U}[k]}{\text{minimiser}} & \left\| \boldsymbol{x} - \boldsymbol{\theta}[k]^{(p)} + \rho^{(p)}\boldsymbol{\lambda}[k]^{(p)} \right\| \\ \text{sous} & I_c^M \boldsymbol{x} \preceq \boldsymbol{U}_{\max} : \boldsymbol{\mu} \end{array} \right\}, \quad \text{(A.72)}$$

avec sa propre variable duale $\boldsymbol{\mu}$. Pour faciliter la notation on utilise $\boldsymbol{x}_0 = \boldsymbol{\theta}[k]^{(p)} + \rho^{(p)}\boldsymbol{\lambda}[k]^{(p)}$.

La solution est faite comme en §A.8.3. Et par conséquence la solution a la même forme PWA que (A.71) :

$$\boldsymbol{\theta}[k]^{(p+1)} = \begin{cases} \boldsymbol{x}_0 + I_c^{M(0)} \left[-P_{\boldsymbol{\mu}}^{(0)} \boldsymbol{U}_{max} + \boldsymbol{s}_{\boldsymbol{\mu}}^{(0)}[k] \right], & \text{si } \boldsymbol{x}_0 \in \mathcal{R}_{\boldsymbol{\mu}}^0 \\ \quad \vdots & \quad \vdots \\ \boldsymbol{x}_0, & \text{si } \boldsymbol{x}_0 \in \mathcal{R}_{\boldsymbol{\mu}}^{2^c-1} \end{cases}, \quad \text{(A.73)}$$

avec des régions $\mathcal{R}_{\boldsymbol{\mu}}^n = \{\boldsymbol{x} \mid G_{\boldsymbol{\mu}}^n[k]\boldsymbol{x} \preceq \boldsymbol{U}_{\max}^n[k]\}$ [16] qui ne se chevauchent pas, formant une particion de l'espace $\boldsymbol{x}_0$. Les élément non creux (codés par les 0 dans la représentation binaire de (n)) de $P_{\boldsymbol{\mu}}^{(n)}$ sont $(I_c^{M(n)} I_c^{M(n)^{\mathrm{T}}})^{-1}$, et ces de $\boldsymbol{s}_{\boldsymbol{\mu}}^{(n)}[k]$ sont $(I_c^{M(n)} I_c^{M(n)^{\mathrm{T}}})^{-1} I_c^{M(n)} \boldsymbol{x}_0$, $I_c^{M(n)}$ est construite par l'enlèvement des contraintes comme codées par (n).

Pour $n = 0$, la solution est la même que pour le cas d'égalité (A.31) ($\# I_c^M = c \times Mc$). Quand $n = 2^c - 1$ (toutes contraintes sont actives), la solution est la même du cas sans contraintes, une fois que $\boldsymbol{\mu} = \boldsymbol{0}$ (voir Remarque 1.13).

Alors, dépendant de dans quel partition $\boldsymbol{\theta}[k]^{(p)} + \rho^{(p)}\boldsymbol{\lambda}[k]^{(p)}$ est, $\boldsymbol{\theta}_i[k]^{(p+1)}$ peut avoir une valeur différent, alors on peut avoir au maximum 2^c possibilities. Et $\boldsymbol{\theta}[k]^{(p)} + \rho^{(p)}\boldsymbol{\lambda}[k]^{(p)}$ depend de $\boldsymbol{\lambda}_i[k]$, qui peut avoir $2^{n_{\text{ineq}}}$ modes différents.

Si en §A.7.4, quand on a fait la substitution nous avons trouvé l'expressions des système DT, si on substitue lésultats en (A.73) on aura des systèmes DT hybrides. Un avec une

16. Also not defined in this work

entrée constante pour $\boldsymbol{\theta}_i[k]$, et un autre homogène pour $\boldsymbol{\lambda}_i[k]$.

Si on conclue le calcules on peut estimer une borne supérieure du nombre total de permutation $\bar{\epsilon}$. Ça totalise

$$\bar{\epsilon} = \overbrace{2^c}^{\text{régions de la projection}} \times \underbrace{2^{n_{\text{ineq}}} \times \cdots \times 2^{n_{\text{ineq}}}}_{M \times \text{régions pour chaque } \boldsymbol{\lambda}_i} = 2^{c+Mn_{\text{ineq}}}. \tag{A.74}$$

Remarque 1.14

Problablement quelques contraintes ne sont pas possible, mais encore on voit la croissance exponentiel de nombre de modes du système hybride. Le calcul exact n'est pas dans le scope de ce travail.

Pour le cas dépourvu de la section antérieure, on utilise le cas (0), où $\boldsymbol{\theta}[k]^{(p)} + \rho^{(p)}\boldsymbol{\lambda}[k]^{(p)}$ est projeté dans l'intersection des tout les demi-espaces. Et du l'autre hypothèse ($\bar{\Gamma}_i \mathring{\boldsymbol{U}}_i^{\star}[k] \succ \boldsymbol{\theta}_i[k], \forall i \in \mathcal{M}, \forall k$), on fixe tous les $\boldsymbol{\lambda}_i$ à la première solution de (A.71). Les choix consécutifs d'hypothèse réduit le nombre total de modes à un, presenté par (A.34) et (A.39) :

$$\boldsymbol{\lambda}^{(p+1)} = \mathcal{A}_\lambda \boldsymbol{\lambda}^{(p)}, \tag{A.34}$$

$$\boldsymbol{\theta}^{(p+1)} = \mathcal{A}_\theta \boldsymbol{\theta}^{(p)} + \boldsymbol{B}_\theta[k] \tag{A.39}$$

Si on concatene les répresentations pour la projection et pour chaque problème local, on aura un numéro avec $c + Mn_{\text{ineq}}$ chiffres. Et alors, la solution en (A.34) et (A.39) peut être vu comme le cas spécifique de :

$$\boldsymbol{\lambda}[k]^{(p+1)} = \begin{cases} \mathcal{A}_\lambda^0 \boldsymbol{\lambda}[k]^{(p)}, & \text{si } \boldsymbol{\theta} \in \mathcal{R}_{\boldsymbol{\lambda}}^0 \\ \vdots & \vdots \\ \mathcal{A}_\lambda^{N_{\boldsymbol{\lambda}}} \boldsymbol{\lambda}[k]^{(p)}, & \text{si } \boldsymbol{\theta} \in \mathcal{R}_{\boldsymbol{\lambda}}^{N_{\boldsymbol{\lambda}}} \end{cases} \tag{A.75}$$

$$\boldsymbol{\theta}[k]^{(p+1)} = \begin{cases} \mathcal{A}_\theta^0 \boldsymbol{\theta}[k]^{(p)} + \boldsymbol{B}_\theta^0[k], & \text{si } \boldsymbol{\theta} \in \mathcal{R}_{\boldsymbol{\theta}}^0 \\ \vdots & \vdots \\ \mathcal{A}_\theta^{N_{\boldsymbol{\theta}}} \boldsymbol{\theta}[k]^{(p)} + \boldsymbol{B}_\theta^{N_{\boldsymbol{\theta}}}[k], & \text{si } \boldsymbol{\theta} \in \mathcal{R}_{\boldsymbol{\theta}}^{N_{\boldsymbol{\theta}}} \end{cases} \tag{A.76}$$

where $N_{\boldsymbol{\lambda}} = N_{\boldsymbol{\theta}} = 2^{c+Mn_{\text{ineq}}} - 1$. Les régions décrites par $\mathcal{R}_{\boldsymbol{\lambda}}^n = \{\boldsymbol{x} \mid G_{\boldsymbol{\lambda}}^n[k]\boldsymbol{\theta}[k]^{(p)} \preceq \boldsymbol{b}_{\boldsymbol{\lambda}}^n[k]\}$, et $\mathcal{R}_{\boldsymbol{\theta}}^n = \{\boldsymbol{x} \mid G_{\boldsymbol{\theta}}^n[k]\boldsymbol{\theta}[k]^{(p)} \preceq \boldsymbol{b}_{\boldsymbol{\theta}}^n[k]\}$ coincident, et sont formées par la superposition des régions de $\mathcal{R}_{\boldsymbol{\lambda}_i}^n$ en (A.73) et $\mathcal{R}_{\boldsymbol{\mu}}^n$ en (A.71).

Si on applique l'attaque (A.21), on a des résultats similaires

$$\tilde{\boldsymbol{\lambda}}[k]^{(p+1)} = \begin{cases} \tilde{\mathcal{A}}_{\lambda}^{0}\tilde{\boldsymbol{\lambda}}[k]^{(p)}, & \text{si } \boldsymbol{x}_0 \in \mathcal{R}_{\boldsymbol{\lambda}}^{0} \\ \vdots & \vdots \\ \tilde{\mathcal{A}}_{\lambda}^{N_{\boldsymbol{\lambda}}}\tilde{\boldsymbol{\lambda}}[k]^{(p)}, & \text{si } \boldsymbol{x}_0 \in \mathcal{R}_{\boldsymbol{\lambda}}^{N_{\boldsymbol{\lambda}}} \end{cases} \tag{A.77}$$

$$\boldsymbol{\theta}[k]^{(p+1)} = \begin{cases} \tilde{\mathcal{A}}_{\theta}^{0}\boldsymbol{\theta}[k]^{(p)} + \tilde{\boldsymbol{B}}_{\theta}^{0}[k], & \text{si } \boldsymbol{\theta} \in \mathcal{R}_{\boldsymbol{\theta}}^{0} \\ \vdots & \vdots \\ \tilde{\mathcal{A}}_{\theta}^{N_{\boldsymbol{\theta}}}\boldsymbol{\theta}[k]^{(p)} + \tilde{\boldsymbol{B}}_{\theta}^{N_{\boldsymbol{\theta}}}[k], & \text{si } \boldsymbol{\theta} \in \mathcal{R}_{\boldsymbol{\theta}}^{N_{\boldsymbol{\theta}}} \end{cases} \tag{A.78}$$

On peut voir que comme l'attaque multiplie $\boldsymbol{\lambda}_i$ par $T_i[k]$, les partitions ne changent pas. Similairement on peut comparer les résultats de la dernière section et voir que $\tilde{\mathcal{A}}_{\boldsymbol{\lambda}}^{0} = \tilde{\mathcal{A}}_{\boldsymbol{\lambda}}$, $\tilde{\mathcal{A}}_{\boldsymbol{\theta}}^{0} = \tilde{\mathcal{A}}_{\boldsymbol{\theta}}$, $\tilde{\boldsymbol{B}}_{\boldsymbol{\theta}}^{0}[k] = \tilde{\boldsymbol{B}}_{\boldsymbol{\theta}}[k]$, et toute l'analyse faite se tient.

Remarque 1.15

Un fait intéressant d'observer est que pour des cas où $\boldsymbol{\lambda}_i$ est déjà $\mathbf{0}$, la multiplication de T_i n'affecte pas sa valeur. Un agent satisfait, n'a pas de motivation de prendre plus de ressources, une fois que les ressources allouées pour lui sont suffisantes pour achever son objectif. Alors, dans le cas le plus extrême, plus spécifiquement le $N_{\boldsymbol{\theta}}$-ème mode, on voit que même si tous les agents trichent simultanément, il n'aura pas d'impact sur le système, une fois qu'il seront déjà satisfaits, $\boldsymbol{\lambda}_i = \mathbf{0}\ \forall i \in \mathcal{M}$, resultant en $\tilde{\mathcal{A}}_{\boldsymbol{\lambda}}^{N_{\boldsymbol{\lambda}}} = \mathcal{A}_{\boldsymbol{\lambda}}^{N_{\boldsymbol{\lambda}}}$, $\tilde{\mathcal{A}}_{\boldsymbol{\theta}}^{N_{\boldsymbol{\theta}}} = \mathcal{A}_{\boldsymbol{\theta}}^{N_{\boldsymbol{\theta}}}$, et $\tilde{\boldsymbol{B}}_{\boldsymbol{\theta}}^{N_{\boldsymbol{\theta}}}[k] = \boldsymbol{B}_{\boldsymbol{\theta}}^{N_{\boldsymbol{\theta}}}[k]$.

On pourrait analyser la stabilité d'une manière cas à cas, on regardant quels contraintes sont actives ou pas pour chaque agent et comme la triche influencie leur solutions. Mais comme on a un numéro qui explose exponentielment, on utilise un simple example différent du cas trivial montré dans la dernière section.

On note la région (ou zone) n, ou la n-zone, les polytopes définies par $\mathcal{R}^n = \{\boldsymbol{x} \mid G_{\boldsymbol{\theta}}^{n}[k]\boldsymbol{x} \preceq \boldsymbol{b}_{\boldsymbol{\theta}}^{n}[k]\}$. On prend le exemple avec 2 contraintes en Fig. A.24. Comme on a 2 contraintes, on a 4 zones :

$$\tilde{\boldsymbol{\lambda}}_i[k] = \begin{cases} -\widetilde{P}_i^{(0)}\boldsymbol{\theta}_i[k] - \widetilde{\boldsymbol{s}}_i^{(0)}[k], & \text{si } \boldsymbol{\theta}_i[k] \in \mathcal{R}^0 \\ -\widetilde{P}_i^{(1)}\boldsymbol{\theta}_i[k] - \widetilde{\boldsymbol{s}}_i^{(1)}[k], & \text{si } \boldsymbol{\theta}_i[k] \in \mathcal{R}^1 \\ -\widetilde{P}_i^{(2)}\boldsymbol{\theta}_i[k] - \widetilde{\boldsymbol{s}}_i^{(2)}[k], & \text{si } \boldsymbol{\theta}_i[k] \in \mathcal{R}^2 \\ -\widetilde{P}_i^{(3)}\boldsymbol{\theta}_i[k] - \widetilde{\boldsymbol{s}}_i^{(3)}[k], & \text{si } \boldsymbol{\theta}_i[k] \in \mathcal{R}^3 \end{cases} \tag{A.79}$$

où

$$\widetilde{P}_i^{(0)} = T_i(\bar{\Gamma}_{i(\star,\star)}H_i^{-1}(\bar{\Gamma}_{i(\star,\star)})^T)^{-1} = T_i(\bar{\Gamma}_i H_i^{-1}\bar{\Gamma}_i{}^T)^{-1} \quad (A.80)$$

$$\widetilde{P}_i^{(1)} = T_i\begin{bmatrix}(\bar{\Gamma}_{i(1,*)}H_i^{-1}(\bar{\Gamma}_{i(1,*)})^T)^{-1} & 0\\ 0 & 0\end{bmatrix} = \begin{bmatrix}T_{i(1,1)}(\bar{\Gamma}_{i(1,*)}H_i^{-1}(\bar{\Gamma}_{i(1,*)})^T)^{-1} & 0\\ T_{i(2,1)}(\bar{\Gamma}_{i(1,*)}H_i^{-1}(\bar{\Gamma}_{i(1,*)})^T)^{-1} & 0\end{bmatrix} \quad (A.81)$$

$$\widetilde{P}_i^{(2)} = T_i\begin{bmatrix}0 & 0\\ 0 & (\bar{\Gamma}_{i(2,*)}H_i^{-1}(\bar{\Gamma}_{i(2,*)})^T)^{-1}\end{bmatrix} = \begin{bmatrix}0 & T_{i(1,2)}(\bar{\Gamma}_{i(2,*)}H_i^{-1}(\bar{\Gamma}_{i(2,*)})^T)^{-1}\\ 0 & T_{i(2,2)}(\bar{\Gamma}_{i(2,*)}H_i^{-1}(\bar{\Gamma}_{i(2,*)})^T)^{-1}\end{bmatrix} \quad (A.82)$$

$$\widetilde{P}_i^{(3)} = T_i\begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix} = \begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix} \quad (A.83)$$

$$\widetilde{\boldsymbol{s}}_i^{(0)}[k] = T_i\boldsymbol{s}_i^{(0)}[k] = T_i(\bar{\Gamma}_i H_i^{-1}\bar{\Gamma}_i{}^T)^{-1}\bar{\Gamma}_i H_i^{-1}\boldsymbol{f}_i[k] \quad (A.84)$$

$$\widetilde{\boldsymbol{s}}_i^{(1)}[k] = T_i\boldsymbol{s}_i^{(1)}[k] = \begin{bmatrix}T_{i(1,1)}(\bar{\Gamma}_{i(1,*)}H_i^{-1}(\bar{\Gamma}_{i(1,*)})^T)^{-1}\bar{\Gamma}_{i(1,*)}H_i^{-1}\boldsymbol{f}_i[k]\\ T_{i(2,1)}(\bar{\Gamma}_{i(1,*)}H_i^{-1}(\bar{\Gamma}_{i(1,*)})^T)^{-1}\bar{\Gamma}_{i(1,*)}H_i^{-1}\boldsymbol{f}_i[k]\end{bmatrix} \quad (A.85)$$

$$\widetilde{\boldsymbol{s}}_i^{(2)}[k] = T_i\boldsymbol{s}_i^{(2)}[k] = \begin{bmatrix}T_{i(1,2)}(\bar{\Gamma}_{i(2,*)}H_i^{-1}(\bar{\Gamma}_{i(2,*)})^T)^{-1}\bar{\Gamma}_{i(2,*)}H_i^{-1}\boldsymbol{f}_i[k]\\ T_{i(2,2)}(\bar{\Gamma}_{i(2,*)}H_i^{-1}(\bar{\Gamma}_{i(2,*)})^T)^{-1}\bar{\Gamma}_{i(2,*)}H_i^{-1}\boldsymbol{f}_i[k]\end{bmatrix} \quad (A.86)$$

$$\widetilde{\boldsymbol{s}}_i^{(3)}[k] = T_i\boldsymbol{s}_i^{(3)}[k] = \begin{bmatrix}0\\ 0\end{bmatrix} \quad (A.87)$$

avec

$$\boldsymbol{s}_i^{(0)}[k] = (\bar{\Gamma}_{i(\star,\star)}H_i^{-1}(\bar{\Gamma}_{i(\star,\star)})^T)^{-1}\bar{\Gamma}_{i(*,*)}H_i^{-1}\boldsymbol{f}_i[k] \quad (A.88)$$

$$\boldsymbol{s}_i^{(1)}[k] = \begin{bmatrix}(\bar{\Gamma}_{i(1,*)}H_i^{-1}(\bar{\Gamma}_{i(1,*)})^T)^{-1}\bar{\Gamma}_{i(1,*)}H_i^{-1}\boldsymbol{f}_i[k]\\ 0\end{bmatrix} \quad (A.89)$$

$$\boldsymbol{s}_i^{(2)}[k] = \begin{bmatrix}0\\ (\bar{\Gamma}_{i(2,*)}H_i^{-1}(\bar{\Gamma}_{i(2,*)})^T)^{-1}\bar{\Gamma}_{i(2,*)}H_i^{-1}\boldsymbol{f}_i[k]\end{bmatrix} \quad (A.90)$$

$$\boldsymbol{s}_i^{(3)}[k] = \begin{bmatrix}0\\ 0\end{bmatrix} \quad (A.91)$$

One voir que pour les zones 1 et 2, les éléments non creux de $P_i^{(n)}$ et $\boldsymbol{s}_i^{(n)}[k]$ influencie les autres éléments utilisant les colonnes de T_i équivalentes aux contraintes actives. Par exemple, dans la zone 1, $T_{i(1,1)}$ et $T_{i(2,1)}$ sont utilisés, et pour la zone 2, $T_{i(1,2)}$ et $T_{i(2,2)}$.

Remarque 1.16

Si T_i est diagonale, il n'y a pas l'injection des "éléments actives" dans un "élément inactive".

Alors, juste quelques colonnes de T_i sont propagées.

Si on a $M = 2$ sous-systèmes, un dans la 1-zone et l'autre dans la 2-zone (appellons les agents I et II) et on suppose que la solution de la projection est dans la 0-zone, i.e., $G^0_{\boldsymbol{\mu}}[k](\boldsymbol{\theta}[k]^{(p)} + \rho^{(p)}\boldsymbol{\lambda}[k]^{(p)}) \preceq \boldsymbol{U}^0_{\max}[k]$, on code la permutation comme

$$\underbrace{00}_{\text{zone de la projection}} \overbrace{01}^{\text{zone de l'agent I}} \underbrace{10}_{\text{zone de l'agent II}} = 000110_{(2)} = 6_{(10)}. \tag{A.92}$$

Appliquant la projection, on a comme solution (5.14)

$$\boldsymbol{\theta}^{(p+1)} = \boldsymbol{\theta}^{(p)} + \rho^{(p)}\boldsymbol{\lambda}^{(p)} + {I_c^M}^{\mathrm{T}}/(I_c^M {I_c^M}^{\mathrm{T}}) \left(I_c^M(\boldsymbol{\theta}^{(p)} - \rho^{(p)}\boldsymbol{\lambda}^{(p)}) - \boldsymbol{U}_{\max} \right), \tag{??}$$

qui peut être récrit

$$\boldsymbol{\theta}_i^{(p+1)} = \boldsymbol{\theta}_i^{(p)} + \frac{1}{M}\left(\sum_{j\in\mathcal{M}} \boldsymbol{\theta}_j^{(p)} - \boldsymbol{U}_{\max} \right) + \rho^{(p)} \left(\boldsymbol{\lambda}^{(p)} - \frac{1}{M}\sum_{j\in\mathcal{M}} \boldsymbol{\lambda}_j^{(p)} \right) \tag{A.93}$$

et en utilisant $\tilde{\boldsymbol{\lambda}}_i$ à la place :

$$\begin{aligned} \boldsymbol{\theta}_i^{(p+1)} = \boldsymbol{\theta}_i^{(p)} + \frac{1}{M}\left(\sum_{j\in\mathcal{M}} \boldsymbol{\theta}_j^{(p)} - \boldsymbol{U}_{\max} \right) + \rho^{(p)} \left(-\widetilde{P}_i^{(n(i))}\boldsymbol{\theta}_i^{(p)} - \widetilde{\boldsymbol{s}}_i^{(n(i))}[k] \right) \\ - \frac{\rho^{(p)}}{M} \sum_{j\in\mathcal{M}} \left(-\widetilde{P}_j^{(n(j))}\boldsymbol{\theta}_j^{(p)} - \widetilde{\boldsymbol{s}}_j^{(n(j))}[k] \right) \end{aligned} \tag{A.94}$$

où $n(i)$ indique le code de zone de l'agent i.

On peut finalement récrire comme

$$\boldsymbol{\theta}^{(p+1)} = \tilde{\mathcal{A}}_\theta^{(6)}\boldsymbol{\theta}^{(p)} + \tilde{\boldsymbol{B}}_\theta^{(6)}[k] \tag{A.95}$$

où

$$\tilde{\mathcal{A}}_\theta^{(6)} = \begin{bmatrix} 2I - \frac{1}{2}\rho^{(p)}\widetilde{P_I}^{(1)} & I + \frac{1}{2}\rho^{(p)}\widetilde{P_{II}}^{(2)} \\ I + \frac{1}{2}\rho^{(p)}\widetilde{P_I}^{(1)} & 2I - \frac{1}{2}\rho^{(p)}\widetilde{P_{II}}^{(2)} \end{bmatrix} \tag{A.96}$$

$$\tilde{\boldsymbol{B}}_\theta^{(6)}[k] = \begin{bmatrix} -\frac{1}{2}\rho^{(p)}\widetilde{\boldsymbol{s}_I}^{(1)}[k] + \frac{1}{2}\rho^{(p)}\widetilde{\boldsymbol{s}_{II}}^{(2)}[k] - \frac{\boldsymbol{U}_{\max}}{2} \\ \frac{1}{2}\rho^{(p)}\widetilde{\boldsymbol{s}_I}^{(1)}[k] - \frac{1}{2}\rho^{(p)}\widetilde{\boldsymbol{s}_{II}}^{(2)}[k] - \frac{\boldsymbol{U}_{\max}}{2} \end{bmatrix} \tag{A.97}$$

Comme attendu les colonnes de T_i sont propagées en $\mathcal{A}_{\boldsymbol{\theta}}$, ce qui change sa dynamique. On pourrait fair pareil pour $\boldsymbol{\lambda}_i$, mais on laisse pour le lecteur.

Pour la dynamique générale du système hybride on peut utiliser des méthodes présentées en [BBM17].

Comme on voit, la relaxation de l'hypothèse a complexifié la solution exponentielment. On discutera comment adapter la détection pour surmonter cette complexité.

A.8.4 Mitigation et pénurie artificielle

On commence par la mitigation, car quelques choix facilitent la détection.

On utilise la même méthode de reconstruction de $\boldsymbol{\lambda}_i$. Si on a une estimation de T_i, qui on assume inversible, on peut reconstruire $\boldsymbol{\lambda}_i$

$$\overset{\text{rec}}{\boldsymbol{\lambda}}_i = \widehat{T_i[k]^{-1}}\tilde{\boldsymbol{\lambda}}_i. \tag{A.98}$$

On estime $\widehat{T_i[k]^{-1}}$ comme en (A.58), mais utilisant la seule version de $P_i^{(n)}$ inversible :

$$\widehat{T_i[k]^{-1}} = \bar{P}_i^{(0)}\hat{\tilde{P}}_i^{(0)}[k]^{-1}. \tag{A.99}$$

We cannot use (6.47) anymore (reprised for clarity), since we do not have P_i.

$$\widehat{T_i[k]^{-1}} = \bar{P}_i\hat{\tilde{P}}_i[k]^{-1}. \tag{A.58}$$

Pour ça on a besoin de que $P_i^{(0)}$ existe et qu'on aye accès à la valeur nominal $\bar{P}_i^{(0)}$.

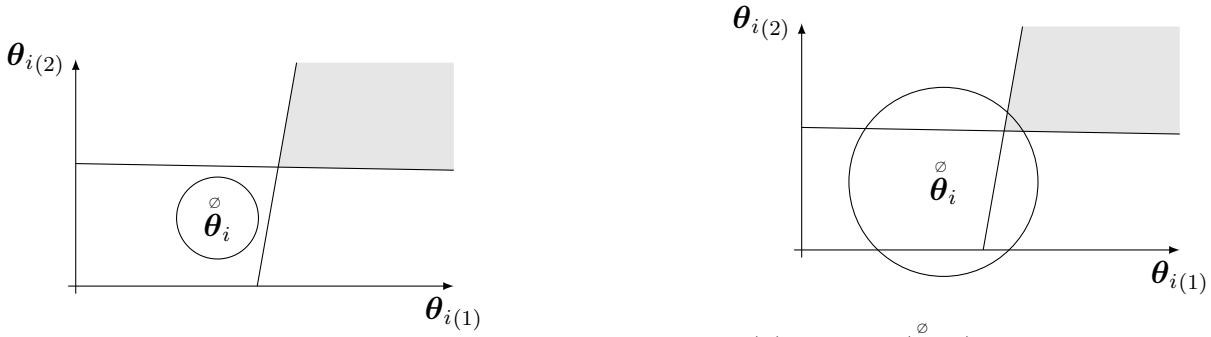
La condition nécessaire pour l'existence de $P_i^{(0)}$ (la 0-zone) est qu'on puisse choisir $\boldsymbol{\theta}_i$ tel que les contraintes soient actives pour la solution sans contraintes $\mathring{\boldsymbol{U}}_i^\star[k]$.

Alors on appelle *pénurie artificielle* la méthode de forcer les contraintes à être actives en choisissant la valeur de $\boldsymbol{\theta}_i$. Une valuer $\overset{\varnothing}{\boldsymbol{\theta}}_i$ qui active les contraintes est appelé le *point de pénurie artificielle*.

En générant des points dans une boule ($\mathcal{B}(\boldsymbol{x}_c, r) = \{\boldsymbol{x} \mid \|\boldsymbol{x} - \boldsymbol{x}_c\| \leqslant r\}$) autour $\overset{\varnothing}{\boldsymbol{\theta}}_i$ (Fig. A.28a), on peut surveiller la communication comme dans la dernière section pour estimer $P_i^{(0)}$.

Comme on ne connait les frontières des polytopes, il est possible de mal choisir un rayon que ne les traversent pas (voir Fig. 7.5b).

Alors, pour l'estimation on a besoin d'une méthode que puisse estimer $P_i^{(0)}$ malgré le choix du rayon. Une méthode qui puisse aussi classifier les zones (vue en §A.8.6).



(a) Boule $\mathcal{B}(\overset{\varnothing}{\boldsymbol{\theta}}_i, r)$.

(b) Boule $\mathcal{B}(\overset{\varnothing}{\boldsymbol{\theta}}_i, r)$ qui traverse les zones.

FIGURE A.28 – Effets de la génération avec différent rayons r.

A.8.5 Détection

Comme vu en (A.71), les parties affines $P_i^{(n)}$ sont invariantes avec le temps. Alors, si on estime $\widehat{\widetilde{P}}_i^{(n)}[k]$, on peut détecter l'attaques si il y a une différence entre la valeur estimée et la nominal $\bar{P}_i^{(n)}$. On peut comparer pour n'importe quel mode non creux [17]. Par contre, comme on aura besoin de $\widehat{\widetilde{P}}_i^{(0)}[k]$ pour la mitigation, on peut l'utiliser pour la détection aussi.

En fixant une borne $\epsilon_{P_i^{(0)}}$, et en calculant une érreur

$$E_i^{(0)}[k] = \left\| \widehat{\widetilde{P}}_i^{(0)}[k] - \bar{P}_i^{(0)} \right\|_F, \tag{A.100}$$

avec une fonction de détection associée $\mathfrak{d}_i^{(0)}$:

$$\mathfrak{d}_i^{(0)} = \mathbb{1}_{\{E_i^{(0)}[k] \geqslant \epsilon_{P_i^{(0)}}\}}, \tag{A.101}$$

on détecte un attaque une fois que la borne soit surpassée, i.e., $\mathfrak{d}_i^{(0)} = 1$.

A.8.6 Estimation de paramètres multiples

On utilise une méthode de la *famille de partitions* [18] La méthode choisie est l'Expectation Maximization (EM) [DLR77 ; Bis06].

Expectation Maximization

Son objectif est de trouver, à partir de un jeux de données observables $\mathcal{B}$, des estimateurs des paramètres $\mathcal{P}$ qui maximisent le log de la vraisemblance marginal $\ln \mathbb{P}(\mathcal{B}; \mathcal{P})$. Les modèles normalement on des variables latentes (pas observables) dans un ensemble $\mathcal{U}$.

17. 2^{Nc}-zones sont completèment creuses et ne changent pas avec la triche (§A.8.3).
18. Les méthodes dans cette famille résolvent des problèmes de régression et classification successifs.

Ce problème de maximiser $\ln \mathbb{P}(\mathcal{B}; \mathcal{P})$ n'a pas de solution analytique.

À la place on maximise la valeur espérée du log de la vraisemblance de données complète $\ln \mathbb{P}(\mathcal{B}, \mathcal{U}; \mathcal{P})$ par rapport les probabilités à posterior $\mathbb{P}(\mathcal{U}|\mathcal{B}; \mathcal{P})$.

Pour calculer les probabilités à posteriori on utilise d'estimations currantes $\mathcal{P}$ dénotés $\mathcal{P}_{\text{cur}}$.

L'algorithm est résumé en Algorithm A.6.

Algorithm A.6: Expectation Maximization

Initialiser $\mathcal{P}_{\text{new}}$
repeat
  $\mathcal{P}_{\text{cur}} \leftarrow \mathcal{P}_{\text{new}}$
  Pas E :
    Calculer $\mathbb{P}(\mathcal{U}|\mathcal{B}; \mathcal{P}_{\text{cur}})$
  Pas M :
    Estimer $\mathcal{P}_{\text{new}}$:

$$\mathcal{P}_{\text{new}} = \underset{\mathcal{P}}{\operatorname{argmax}} \ \mathbb{E}_{\mathbb{P}(\mathcal{U}|\mathcal{B};\mathcal{P}_{\text{cur}})} \left[\ln \mathbb{P}(\mathcal{B}, \mathcal{U}; \mathcal{P})\right] \tag{A.102}$$

until $\mathcal{P}_{\text{cur}}$ *a convergé*

A.8.7 Adaptation pour le problème étudié

Comme EM est probabilistique on doit incorporer un comportement probabilistique dans notre modèle (A.71).

On enlève les indices i et $[k]$, pour faciliter la lecture.

D'abord, on utilise les homolques stochastiques $\underline{\boldsymbol{\theta}_i}$ and $\underline{\boldsymbol{\lambda}_i}$. Si on surveille O échanges entre agent et coordinateur, on voit variables d'entrée et réponses, $\underline{\boldsymbol{\theta}_o}$ et $\underline{\boldsymbol{\lambda}_o}$, $o \in \mathcal{O} = \{0 : O-1\}$. On les organise en $\underline{\Theta}, \underline{\Lambda} \in \mathbb{R}^{c \times O}$. Et la tuple $(\underline{\Theta}, \underline{\Lambda})$ est l'ensemble observable $\mathcal{B}$.

Comme chaque $\underline{\boldsymbol{\theta}_o}$ reste dans une région, on l'associe à une variable $z \in \mathcal{Z} = \{0 : Z-1\}$ qui indique le numéro de zone. Notre modèle devient

$$\underline{\boldsymbol{\lambda}_o} = \begin{cases} -\tilde{P}^0 \underline{\boldsymbol{\theta}_o} - \tilde{\boldsymbol{s}}^0, & \text{si dans zone } 0 \\ \vdots & \vdots \\ -\tilde{P}^{Z-1} \underline{\boldsymbol{\theta}_o} - \tilde{\boldsymbol{s}}^Z, & \text{si dans zone } Z-1 \end{cases}. \tag{A.103}$$

En [FS10], une méthode similaire est appellée *mixture de régressions linéaires*, mais comme on a des fonctions PWA, on l'apelle *mixture de régressions affines*.

Comme on ne connait pas l'appartenance de chaque point observé, associe l'index z de chaque observation $(\underline{\boldsymbol{\lambda}_o}, \underline{\boldsymbol{\theta}_o})$ à une variable latente $\underline{z_o}$. On organise $\underline{z_o}$ en $\underline{Z} \in \mathbb{R}^{1 \times O}$, qui est notre ensemble pas observable $\mathcal{U}$.

On suppose que $\underline{z}_o$ suit une distribution catégorique, avec des probabilités associées $\Pi = \{\pi^z | z \in \mathcal{Z}\}$ tels que

$$\mathbb{P}(\underline{z}_o = z) = \pi^z \in [0, 1], \qquad \sum_{z \in \mathcal{Z}} \pi^z = 1.$$

On construit l'ensemble de paramètres $\mathcal{P} = \{\mathcal{P}^z \mid z \in \mathcal{Z}\}$, avec $\mathcal{P}^z = (\tilde{P}^z, \tilde{\boldsymbol{s}}^z, \pi^z)$.

Remarque 1.17

On estime π^z et $\tilde{\boldsymbol{s}}^z$ puremen pour mettre le modèle à jour et contraindre l'identification de $\tilde{P}^z$.

Pour $\boldsymbol{\theta}$, on considère une densité de probabilité impropre et non informative [Chr+10]

$$\mathbb{P}(\underline{\boldsymbol{\theta}}_o) \propto 1,$$

que signifie qu'on connait presque sûrement la valeur de $\underline{\boldsymbol{\theta}}_o$.

Une fois définies les variables d'entrées et latents, on modèle la réponse $\underline{\lambda}_o$ comme une variable normale multivariables avec fonction de densité de probabilité

$$\mathbb{P}(\underline{\boldsymbol{\lambda}}_o | \underline{\boldsymbol{\theta}}_o, \underline{z}_o = z; \mathcal{P}^z) = \mathcal{N}(\underline{\boldsymbol{\lambda}}_o; f(\underline{\boldsymbol{\theta}}_o; \mathcal{P}^z), \Sigma^z), \tag{A.104}$$

qui suivant le modèle de la mixture de régressions affines (A.103) a comme valeur espérée

$$f(\underline{\boldsymbol{\theta}}_o; (P, \boldsymbol{s}, \pi)) = -P\underline{\boldsymbol{\theta}}_o - \boldsymbol{s}, \tag{A.105}$$

Les matrices de covariance Σ^z tend à 0_c, s'approachant des valeurs originales des fonctions en (A.71).

Un exemple de la répresentation de tel fonction pour 1 dimension est vue en Fig. 7.6, où points plus sombres présentent plus de probabilité que les plus clairs.

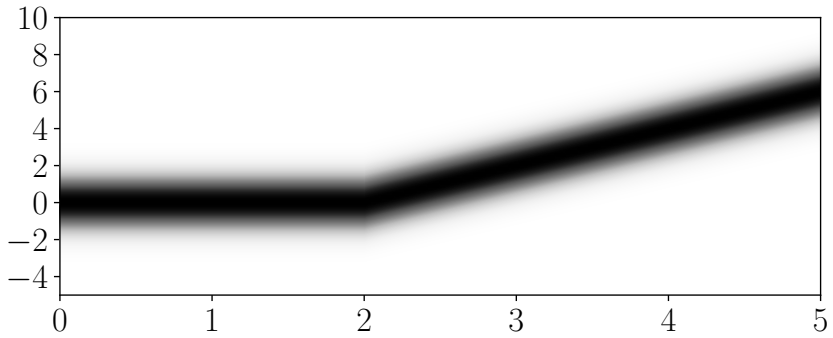


FIGURE A.29 – Mixture gaussianne pour PWA d'une dimension.

Les probabilités à posteriori $\zeta_{zo}(\mathcal{P}) = \mathbb{P}(\underline{z}_o = z | \underline{\boldsymbol{\lambda}}_o, \underline{\boldsymbol{\theta}}_o; \mathcal{P})$, aussi appellées *responsabilités*,

sont calculées :

$$\zeta_{zo}(\mathcal{P}) = \frac{\pi_z \mathcal{N}(\underline{\boldsymbol{\lambda}}_o; f(\underline{\boldsymbol{\theta}}_o; \mathcal{P}^z), \Sigma^z)}{\sum_{j=1}^{Z} \pi_j \mathcal{N}(\underline{\boldsymbol{\lambda}}_o; f(\underline{\boldsymbol{\theta}}_o; \mathcal{P}^j), \Sigma^j)}. \tag{A.106}$$

On les appelle responsabilités car on calcule la probabilité de que la zone z soit responsable d'avoir généré l'observation o.

On calcule la valeur espérée de $\ln \mathbb{P}(\underline{\Theta}, \underline{\Lambda}, \underline{Z}; \mathcal{P})$ par rapport à $\zeta_{zo}(\mathcal{P}_{\text{cur}})$ [Bis06, Chapitre 9] en utilisant

$$\mathbb{E}_{\zeta_{zo}(\mathcal{P}_{\text{cur}})} \left[\ln \mathbb{P}(\underline{\Theta}, \underline{\Lambda}, \underline{Z}; \mathcal{P})\right] = \sum_{o \in \mathcal{O}} \sum_{z \in \mathcal{Z}} \zeta_{zo}(\mathcal{P}_{\text{cur}}) \alpha_{zo}, \tag{A.107}$$

où $\alpha_{zo} = \ln \pi_z + \ln \mathcal{N}(\underline{\boldsymbol{\lambda}}_o; f(\underline{\boldsymbol{\theta}}_o; \mathcal{P}^z), \Sigma^z)$.

Pour faciliter les calculs, on introduit une variable $\boldsymbol{\phi}^z$ pour intégrer les paramètres, ainsi comme on a fait dans le cas avec le cas RLS (§A.7.6)

$$\boldsymbol{\phi}^z = \begin{bmatrix} \text{vec}(\tilde{P}^z)^T \\ \tilde{\boldsymbol{s}}^z \end{bmatrix}. \tag{A.108}$$

Pour trouver le $\boldsymbol{\phi}^z$ optimal pour le problème en (A.102), on prend les gradients de (**??**) par rapport aux vecteurs $\boldsymbol{\phi}^z$ et on les fait disparaître.

On trouve comme solution

$$\boldsymbol{\phi}^z_{\text{new}} = (\Xi^z \underline{\Omega})^\dagger \Xi^z \text{vec}(\underline{\Lambda}), \tag{A.109}$$

où $\underline{\Omega} = [(\Upsilon \underline{\Theta} \Delta) \circ Y; G]$, avec des matrices $\Upsilon = \mathbf{1}_c^T \otimes I_c$, $\Delta = I_O \otimes \mathbf{1}_c^T$, $Y = G \otimes \mathbf{1}_c$, $G = \mathbf{1}_O^T \otimes I_c$, et

$$\Xi^z = \text{diag}(\sqrt{\zeta(z_{z1}; \mathcal{P}_{\text{cur}})} I_c, \cdots, \sqrt{\zeta(z_{zO}; \mathcal{P}_{\text{cur}})} I_c).$$

Si on fait le même, mais pour π^z on trouve

$$\pi^z = \sum_{o \in \mathcal{O}} \frac{\zeta_{zo}(\mathcal{P}_{\text{cur}})}{O}.$$

On voit que (A.109) est une solution pondéré d'un Least Squares (LS), avec les responsabilités comme poids, qui ajustent la contribution de tous les observations aux modèles de régression.

Il existe quelques similarités avec la méthode K-planes (et k-means) (voir [BM00]), mais l'EM est plus transigen. Au lieu d'affecter l'observation à un zone avec 100% de sûreté, EM utilise la responsabilité de chase zone (*affectation fluide*).

Quand $\boldsymbol{\phi}_z^{\text{new}}$ converge, on peut récupérer les estimations de $\tilde{P}^z$ et $\tilde{\boldsymbol{s}}^z$, pour utiliser dans les schémas de détection et mitigation.

Comme les indices z ne correspondent pas forcément à ceux de (A.71), il faut trouver le z qui correspond à la 0-zone. On prend l'observation o qui correspond à $\overset{\varnothing}{\boldsymbol{\theta}}_i$, qui comme on sait appartient à la 0-zone, et trouve la région plus probable avec

$$\arg\max_z.\ \zeta_{zo}(\mathcal{P}).$$

Le paramètre avec cet index correspond à notre $\widehat{\widetilde{P}}_i^{(0)}[k]$.

Quelques réserves

C'est connu [CG92 ; Bis06 ; BC15] que l'algorithme peut présenter une convergence lente. En [FS10], les autrices comparent la convergence de l'EM avec quelques variantes (stochastic Expectation Maximization (sEM) et Classification Expectation Maximization (CEM)).

Un autre point est la dépendance de la solution par rapport l'initialisation de l'algorithme [KX03 ; BC15].

Encore un autre et l'inversion de quelques variables (notably the covariance Σ) quand utilisant modèles de dimensions élevés. Ça va dépendre des observations O et le nombre de zones Z. Une façon de contrer l'éffet est inclure dans les éléments estimés. Une autre solution est considérer la matrice diagonal avec les mêmes valeurs [KX03]. Encore autre solution est d'utiliser la technique appellée *recuite simulée* [CG92 ; OF10], où les covariances sont initialisées avec des valeurs importantes, indiquant l'incertitude des paramètres, et les faires decroitre à chaque itération. Dans ce travail on a utilisé les deux dernières solutions combinées.

A.8.8 La dMPC résiliente utilissant la pénurie artificielle

En mettant ensemble toutes les parties on a l'Algorithme A.7, qui résume la dMPC résiliente basée sur la décomposition primale utilisant la pénurie artificielle, ou en anglais Resilient Primal Decomposition-based dMPC under Artificial Scarcity (RPdMPC-AS). De nouveau, on peut voir la détection et reconstitution conditionnel de $\boldsymbol{\lambda}_i$ comme un superviseur, comme montré dans la dernière section.

A.8.9 Expériment Numérique

On utilise le même étude de cas que en §A.7.10, avec des moidication pour respecter les hypothèses de cette section.

Pour utiliser la RPdMPC-AS on suppose que $\mathring{\boldsymbol{U}}_i^\star[k]$ respecte les contraintes locales. Alors, comme on a $\boldsymbol{u}_i[k] \succeq \mathbf{0}$ en §A.7.10, on suppose que $\mathring{\boldsymbol{U}}_i^\star[k] \succeq \mathbf{0}$.

Cela est accompli en changeant les états initiaux et les références ($\boldsymbol{x}_i[k]$ et $\boldsymbol{w}_i[k]$).

Algorithm A.7: La dMPC résiliente basée sur la décomposition primale utilisant pénurie artificielle.

Coordinator initialization:
  Initialiser k, p, a, b, $p_{\max}$ and ϵ
while $k \geqslant 0$ **do**
  Detection Phase:
    Initialiser taille de la boule r
    for $o \in \mathcal{O}$ **do**
      Coordinateur envoie $\boldsymbol{\theta}_i^o \in \mathcal{B}(\overset{\varnothing}{\boldsymbol{\theta}}_i, r)$
      Sous-systèmes résolvent (5.6a), et envoient $\tilde{\boldsymbol{\lambda}}_i^o$
    Coordinator estime $\widehat{\widetilde{P}}_i^{(0)}[k]$ et $\widehat{\widetilde{\boldsymbol{s}}}_i^{(0)}[k]$ avec EM (Algorithme A.6)
    Coordinator compute $\mathfrak{d}_i$ utilisant (**??**)
  Negotiation Phase:
    Coordinator initialise $\boldsymbol{\theta}^{(p)}$ et les envoie aux sous-systèmes
    repeat
      Agents résolvent (A.19a) et envoient $\boldsymbol{\lambda}_i[k]^{(p)}$
      Coordinateur m-à-j les allocations (A.20) avec versions adéquades de $\boldsymbol{\lambda}_i$:
        $\boldsymbol{\lambda}_i^\star(\boldsymbol{\theta}^{(p)})$, si $\mathfrak{d}_i = 0$ et $\overset{\text{rec}}{\boldsymbol{\lambda}}_i$, si $\mathfrak{d}_i = 1$ (7.45)
      $p \leftarrow p + 1$
    until $\left[\left\|\boldsymbol{\theta}[k]^{(p)} - \boldsymbol{\theta}[k]^{(p-1)}\right\| \leqslant \epsilon\right] \vee [p \geqslant p_{\max}]$
    Agents appliquent la dernière commande $\boldsymbol{u}_i^\star[0|k]$ calculée
    $k \leftarrow k + 1$

De cette façon on utilise pour les états $\boldsymbol{x}_{\mathrm{I}}[0] = [17.\ \ 3.2]^{\mathrm{T}}$, $\boldsymbol{x}_{\mathrm{II}}[0] = [20.\ \ 5.3]^{\mathrm{T}}$, $\boldsymbol{x}_{\mathrm{III}}[0] = [15.\ \ 3.1]^{\mathrm{T}}$, and $\boldsymbol{x}_{\mathrm{IV}}[0] = [17.\ \ 5.7]^{\mathrm{T}}$, et pour les référénces $w_{\mathrm{I}}[0] = 25.5$, $w_{\mathrm{II}}[0] = 24.0$, $w_{\mathrm{III}}[0] = 18.0$, and $w_{\mathrm{IV}}[0] = 20.4$. On maintien tous les autres paramètres, avec qui quand on décompose la MPC on a

$$\begin{aligned} \underset{\boldsymbol{U}_i[k]}{\text{minimiser}} \quad & J_i = \tfrac{1}{2}\left\|\boldsymbol{U}_i[k]\right\|_{H_i}^2 + \boldsymbol{f}_i[k]^T \boldsymbol{U}_i[k] \\ \text{sous} \quad & \bar{\Gamma}_i \boldsymbol{U}_i[k] \succeq \boldsymbol{\theta}_i[k] : \boldsymbol{\lambda}_i[k] \\ & \boldsymbol{U}_i[k] \succeq \boldsymbol{0} \end{aligned} \tag{A.110}$$

et fonction de mise à jour (A.20).

Résultats

On test l'algorithm avec la même période de simulation 5.0 heures, i.e., $k \in \{0, 5.0/T_s\}$. Les mêmes scénarios sont utilisés :

1. Comportement Nominal (dénoté N)
2. Agent I attaque le système commandé par dMPC standard (dénoté S)
3. Agent I attaque le système commandé par RPdMPC-DS (dénoté C)

L'attaque choisi pour les scenarios 2 et 3, sont

$$T_I = \begin{cases} \begin{bmatrix} 14.43288267 & 0. & 0. & 0. \\ 0. & 13.4590903 & 0. & 0. \\ 0. & 0. & 6.93065061 & 0. \\ 0. & 0. & 0. & 3.4447393 \end{bmatrix}, & \text{si } k > 10 \\ I_c, & \text{sinon} \end{cases} \tag{A.111}$$

Si on regarde Fig. 7.7, on peut comparer la température de l'air à l'intérieur de la maison I avec ses références w_I. On peut voir aussi dans la figure les variables $E_I^{(0)}[k]$ et borne $\epsilon_{P_i^{(0)}}$. Observe comment les références w_I ne sont achevées dans le comportement nominal (en magenta), du aux conditions initiales et contraintes de puissance. Comme attendu, da variable de decision se situe sous la borne $\epsilon_{P_i^{(0)}} = 10^{-4}$ avec valeurs de l'orde $E_I^N(k) \approx 10^{-10}$ (pas d'attaque détecté).

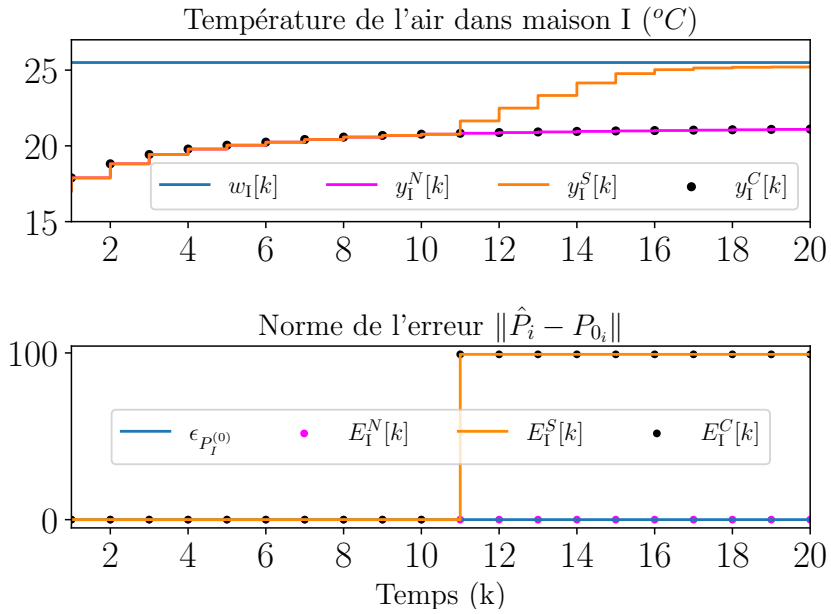


FIGURE A.30 – Température de la maison I et la variable $E_I[k]$ pour les 3 scénarios.

Quand agent I attaque le système (en orange), l'erreur de suivi $w_I - y_I$ décroît, et l'agent presque achève la référence. En Fig. A.31, on voit que quand agent I triche, les ressources sont transférées de toutes les autres maisons, qui perdent en performance.

Dans ce cas, la variable de détection surpasse la borne $\epsilon_{P_i^{(0)}}$, $E_I^S \approx 200$, ce qu'indique le changement de comportement, un possible attaque.

Si on applique l'algorithme proposé (point noires), la température approche sa valeur nominale y_I^N. On peut comparer aussi les entrées en Fig. A.32. Quand agent I est égoïste, sa commande devienne plus importante, tant que celles des autres maisons diminuent, montrant le déplacement de la négociation. Et quand la correction est activée, la commande aussi récupere sa valeur d'origine.

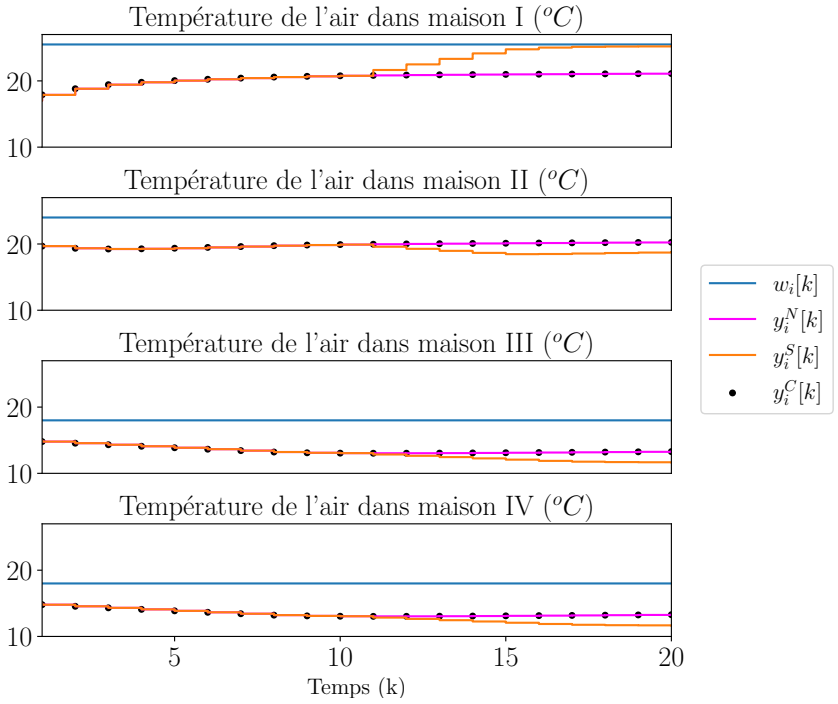


FIGURE A.31 – Température de l'air dans toutes les maisons pour les 3 scénarios.

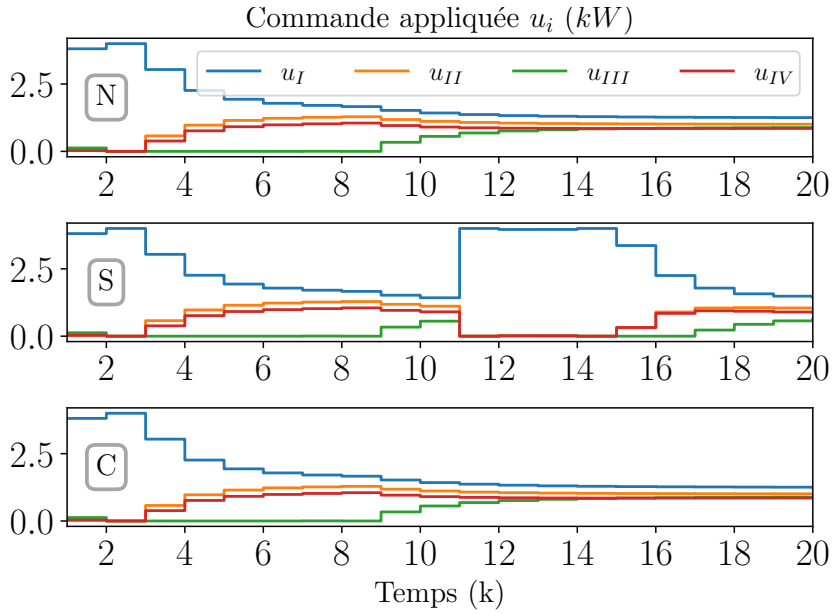


FIGURE A.32 – Commande appliquée en toutes les maisons pour les 3 scénarios.

Une autre façon de comparer les performances est d'accumuler les fonctions objectif pendant la simulation pour les trois scénarios (Tab. A.5). On calcule aussi l'erreur percentuel $\frac{J^i - J^N}{J^N}$. Quand agent I attaque, on voit une décroissance de son objectif ($\approx -40\%$), dégradant la performance global en $\approx +10\%$. Cependant, quand la correction est activée, l'erreur percentuel est de l'ordre $|\frac{J^C - J^N}{J^N}| \leqslant 10^{-8}$.

TABLE A.5 – Fonctions objectif J_i (% erreur).

Agent	Scénario N	Scénario S	Scénario C
I	19868.2	12618.5 (−36.5)	19868.2 (−0.0)
II	13784.5	18721.1 (35.8)	13784.5 (0.0)
III	17276.0	22324.9 (29.2)	17276.1 (0.0)
IV	10086.0	13872.4 (37.5)	10086.0 (0.0)
Global	61014.7	67536.9 (10.7)	61014.7 (−0.0)

A.9 Conclusion

A.9.1 Rétrospective et Contributions

Dans cette thèse on a étudié des attaques de type FDI sur les CPS commandés par la dMPC basée sur la décomposition primale.

D'abord, on a présenté la dMPC et des méthodes pour la décomposer. De la topologie jusqu'à communication. On a discuté la sécurité des CPS avec un focus sur les attaques en systèmes commandés par la dMPC. Ce qui est assez récent (moins de 5 ans).

On a présenté la dMPC basée sur la décomposition primale et nous avons étudié ses vulnerabilités.

À partir de quelques hypothèses et analyse du système, on a créé des schémas de détection et mitigation des effet des attaques.

Après on a relaxé une hypothèse, ce qui a augenté de manière exponentiel la complexité du problème. En utilisant quelques proprietés, on a adapté les méthodes présentées pour contourner la complexité.

A.9.2 Contributions

À part de la systematization du déssin des CPS commandés par dMPC, on liste comme contribution :

- L'étude de la vulnerabilité de la dMPC basée sur la décomposition primale
- Stratégies Résilientes pour 2 types de systèmes de complexités croissantes
  - Systèmes dépourvus (demande plus grande que les ressources)
  - Systèmes avec pénurie artificielle possible (demande optimale raisonable)

A.9.3 Inconvénient

L'inconvénient principal de l'approche est la necessité de l'information de pénurie du système. D'abord, c'était donné par la nature du système, avec contraints qui privaient le système d'achever ses objectifs. Après, même avec un système pas dépourvu, une privation complète pourrait être imposée par le coordinateur, de manière artificielle en changeant les allocations. Par contre, on nécessitait de supposer que le point de pénurie artificielle respectait les contraintes locales. Alors, la question en ouvert est qu'est-ce que ce passe quand on ne peut pas imposer la pénurie complète, c'est à dire quand on n'a pluse les points de pénurie artificielle.

A.10 Directions Futures

Comme tout travail, plusieurs question sont émergées pendant les études et nous sommes nous rencontrés plusieurs fois à des carrefours, qui nous ont obligé de choisir une direction.

Ayant à l'esprit les inconvénient et aussi des choix faits, on liste quelques questions et chemins non croisés. Et on espère qu'ils servent comme inspiration pour des travaux futurs. On les liste en ordre de moins marche arrière nécessaire, i.e., quant fundamentaux les changements nécessaires pour achever l'étude.

— Étude de la robustesse de la stratégie sous bruit de communication

— Réconstruction partiel de la matrice de triche, quand $\widehat{\widetilde{P}}_i^{(0)}[k]$ n'est pas disponible

— Adaptation de la stratégie résiliente pour les contraintes souples [AB09]

— Utiliser l'EM (ou alternative) récursive pour l'estimation

— Étude de l'adaptation de la stratégie résilient avec arrêt acceleré (comme en [DBG17])

— Étude de l'adaptation de la stratégie pour autres méthodes de décomposition

— Étude d'autre modèles d'attaque (attaque affin initialement)

BIBLIOGRAPHY

[AB09] Alessandro Alessio and Alberto Bemporad, « A Survey on Explicit Model Predictive Control », *in*: *Nonlinear Model Predictive Control: Towards New Challenging Applications*, ed. by Lalo Magni, Davide Martino Raimondo, and Frank Allgöwer, Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 345–369, ISBN: 978-3-642-01094-1, DOI: 10.1007/978-3-642-01094-1_29 (cit. on pp. 84, 109, 133, 182, 200).

[ACM21] T. Arauz, P. Chanfreut, and J.M. Maestre, « Cyber-Security in Networked and Distributed Model Predictive Control », *in*: *Annu Rev Control* (Nov. 2021), ISSN: 1367-5788, DOI: 10.1016/j.arcontrol.2021.10.005 (cit. on pp. 42, 45, 58, 60, 62, 145, 146, 152–155).

[ACS09] Saurabh Amin, Alvaro A. Cárdenas, and S. Shankar Sastry, « Safe and Secure Networked Control Systems under Denial-of-Service Attacks », *in*: *Hybrid Systems: Computation and Control*, ed. by Rupak Majumdar and Paulo Tabuada, Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 31–45, ISBN: 978-3-642-00602-9, DOI: 10.1007/978-3-642-00602-9_3 (cit. on pp. 54, 56, 149).

[Alo+12] Javier Alonso et al., « Software Rejuvenation: Do IT & Telco Industries Use It? », *in*: *2012 IEEE 23rd International Symposium on Software Reliability Engineering Workshops*, 2012, pp. 299–304, DOI: 10.1109/ISSREW.2012.96 (cit. on pp. 60, 153).

[Ana+18] Wicak Ananduta et al., « Resilient Distributed Energy Management for Systems of Interconnected Microgrids », *in*: *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 3159–3164, DOI: 10.1109/CDC.2018.8619548 (cit. on pp. 32, 59, 62, 66, 67, 141, 152, 154–157).

[Ana+19] Wicak Ananduta et al., « A Resilient Approach for Distributed MPC-Based Economic Dispatch in Interconnected Microgrids », *in*: *2019 18th European Control Conference (ECC)*, 2019, pp. 691–696, DOI: 10.23919/ECC.2019.8796208 (cit. on pp. 59, 62, 66, 67, 152, 154–157).

[Ana+20] Wicak Ananduta et al., « Resilient Distributed Model Predictive Control for Energy Management of Interconnected Microgrids », *in*: *Optimal Control Applications and Methods* 41.*1* (2020), pp. 146–169, DOI: 10.1002/oca.2534, URL: https://onlinelibrary.wiley.com/doi/pdf/10.1002/oca.2534 (cit. on pp. 59, 62, 66, 67, 152, 154–157).

[ÅW89] K.J. Åström and B. Wittenmark, *Adaptive Control*, Addison-Wesley series in electrical and computer engineering: Control engineering, Addison-Wesley, 1989, ISBN: 9780201097207, DOI: 10.1007/978-3-662-08546-2_24 (cit. on pp. 91, 93, 168, 170).

[BAL20a] Sarah Braun, Sebastian Albrecht, and Sergio Lucia, « A Hierarchical Attack Identification Method for Nonlinear Systems », *in*: *2020 59th IEEE Conference on Decision and Control (CDC)*, 2020, pp. 5035–5042, DOI: 10.1109/CDC42340.2020.9304231 (cit. on pp. 62, 155).

[BAL20b] Sarah Braun, Sebastian Albrecht, and Sergio Lucia, « Hierarchical Attack Identification for Distributed Robust Nonlinear Control », *in*: *IFAC-PapersOnLine* 53.*2* (2020), 21th IFAC World Congress, pp. 6113–6120, ISSN: 2405-8963, DOI: 10.1016/j.ifacol.2020.12.1688 (cit. on pp. 62, 155).

[Bar+20] A. Barboni et al., « Detection of Covert Cyber-Attacks in Interconnected Systems: a Distributed Model-Based Approach », *in*: *IEEE Trans. Autom. Control* 65.*9* (2020), pp. 3728–3741, DOI: 10.1109/tac.2020.2998765 (cit. on pp. 58, 151).

[BBM17] Francesco Borrelli, Alberto Bemporad, and Manfred Morari, *Predictive Control for Linear and Hybrid Systems*, Cambridge University Press, 2017, DOI: 10.1017/9781139061759 (cit. on pp. 32, 34, 117, 142, 189).

[BC15] Jean-Patrick Baudry and Gilles Celeux, « Em for Mixtures », *in*: *Statistics and Computing* 25.*4* (June 2015), pp. 713–726, ISSN: 1573-1375, DOI: 10.1007/s11222-015-9561-x (cit. on pp. 121, 125, 194).

[Bem+02] Alberto Bemporad et al., « The Explicit Linear Quadratic Regulator for Constrained Systems », *in*: *Automatica* 38.*1* (2002), pp. 3–20, ISSN: 0005-1098, DOI: 10.1016/S0005-1098(01)00174-1 (cit. on pp. 108, 109, 181, 182).

[BGB12] Romain Bourdais, Hervé Guéguen, and Aziz Belmiloudi, « Distributed Model Predictive Control for a Class of Hybrid System Based on Lagrangian Relaxation », *in*: *IFAC Proceedings Volumes* 45.*9* (2012), pp. 46–51, DOI: 10.3182/20120606-3-nl-3011.00006 (cit. on pp. 40, 144).

[Bie+12] Benjamin Biegel et al., « Congestion Management in a Smart Grid via Shadow Prices », *in*: vol. 45, 21, 2012, pp. 518–523, DOI: 10.3182/20120902-4-FR-2032.00091, URL: https://doi.org/10.3182/20120902-4-FR-2032.00091 (cit. on pp. 40, 144).

[Bin17] A. Bindra, « Securing the Power Grid: Protecting Smart Grids and Connected Power Systems From Cyberattacks », *in*: *IEEE Power Electron. Mag.* 4.*3* (2017), pp. 20–27, DOI: 10.1109/mpel.2017.2719201 (cit. on pp. 25, 138).

[Bis05] Matthew A. Bishop, *Introduction to Computer Security*, Addison-Wesley Professional, 2005, ISBN: 9780321247445 (cit. on pp. 52–54, 59, 148, 149, 152).

[Bis06] Christopher M. Bishop, *Pattern Recognition and Machine Learning*, Springer Science and Business Media LLC, Aug. 2006, ISBN: 9780387310732 (cit. on pp. 120, 123, 125, 190, 193, 194).

[BLM08] Thomas Besselmann, Johan Löfberg, and Manfred Morari, « Explicit Model Predictive Control for Systems With Linear Parameter-Varying State Transition Matrix », *in*: *IFAC Proceedings Volumes* 41.*2* (2008), 17th IFAC World Congress, pp. 13163–13168, ISSN: 1474-6670, DOI: 10.3182/20080706-5-KR-1001.02230 (cit. on pp. 21, 137).

[BM00] P.S. Bradley and O.L. Mangasarian, « K-Plane Clustering », *in*: *J Global Optim* 16.*1* (2000), pp. 23–32, ISSN: 0925-5001, DOI: 10.1023/a:1008324625522 (cit. on pp. 124, 193).

[BM14] Moses Bangura and Robert Mahony, « Real-Time Model Predictive Control for Quadrotors », *in*: *IFAC Proceedings Volumes* 47.*3* (2014), 19th IFAC World Congress, pp. 11773–11780, ISSN: 1474-6670, DOI: 10.3182/20140824-6-ZA-1003.00203 (cit. on pp. 21, 32, 137, 141).

[Boe+20] F. Boem et al., « Distributed Fault-Tolerant Control of Large-Scale Systems: an Active Fault Diagnosis Approach », *in*: *IEEE Trans. Control Netw. Syst.* 7.*1* (2020), pp. 288–301, DOI: 10.1109/tcns.2019.2913557 (cit. on pp. 62, 154).

[Bof+19] Nicoletta Bof et al., « Multiagent Newton-Raphson Optimization Over Lossy Networks », *in*: *IEEE Transactions on Automatic Control* 64.*7* (2019), pp. 2983–2990, DOI: 10.1109/TAC.2018.2874748 (cit. on p. 57).

[Boy+11] S. Boyd et al., *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*, vol. 3, 1, Now Publishers Inc., 2011, pp. 1–122, DOI: 10.1561/2200000016 (cit. on pp. 40, 144).

[Boy+15] Stephen Boyd et al., « Notes on Decomposition Methods », *in*: *Notes for EE364B*, ed. by Stanford University, 2015 (cit. on pp. 35, 38, 40, 65, 71, 106, 143, 144, 158).

[BT95] Steven J. Brams and Alan D. Taylor, « An Envy-Free Cake Division Protocol », *in*: *The American Mathematical Monthly* 102.*1* (1995), pp. 9–18, ISSN: 00029890, 19300972, URL: http://www.jstor.org/stable/2974850 (cit. on p. 75).

[BV04] Stephen Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004, ISBN: 0521833787, DOI: 10.1017/cbo9780511804441 (cit. on pp. 32, 35, 71, 82, 84, 85, 108, 144, 158, 162, 164, 180).

[CAS08] Alvaro A. Cardenas, Saurabh Amin, and Shankar Sastry, « Secure Control: Towards Survivable Cyber-Physical Systems », *in*: *2008 The 28th International Conference on Distributed Computing Systems Workshops*, 2008, pp. 495–500, DOI: 10.1109/ICDCS.Workshops.2008.40 (cit. on pp. 54, 55, 59, 149, 153).

[CC18] Ryan Cole and Liang Cheng, « Modeling the Energy Consumption of Blockchain Consensus Algorithms », *in*: *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2018, pp. 1691–1696, DOI: 10.1109/Cybermatics_2018.2018.00282 (cit. on p. 49).

[CG92] Gilles Celeux and Gérard Govaert, « A Classification Em Algorithm for Clustering and Two Stochastic Versions », *in*: *Computational Statistics & Data Analysis* 14.*3* (1992), pp. 315–332, ISSN: 0167-9473, DOI: 10.1016/0167-9473(92)90042-E (cit. on pp. 125, 194).

[Che+11] Xianzhong Chen et al., « Model predictive control of nonlinear singularly perturbed systems: Application to a reactor-separator process network », *in*: *2011 50th IEEE Conference on Decision and Control and European Control Conference*, 2011, pp. 8125–8132, DOI: 10.1109/CDC.2011.6160469 (cit. on pp. 35, 143).

[Che+21] Guoyong Chen et al., « Consensus of Discrete-Time Multi-Agent Systems Over Packet Dropouts Channels », *in*: *J. Franklin Inst. B* (2021), ISSN: 0016-0032, DOI: 10.1016/j.jfranklin.2021.04.045 (cit. on pp. 57, 60, 153).

[Chr+10] Ronald Christensen et al., *Bayesian ideas and data analysis: an introduction for scientists and statisticians*, CRC press, 2010 (cit. on pp. 122, 192).

[Chr+13] Panagiotis D. Christofides et al., « Distributed Model Predictive Control: a Tutorial Review and Future Research Directions », *in*: *Computers & Chemical Engineering* 51 (Apr. 2013), pp. 21–41, DOI: 10.1016/j.compchemeng.2012.05.011 (cit. on pp. 45, 146).

[Cil+20] Thomas Cilloni et al., « Understanding and Detecting Majority Attacks », *in*: *2020 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, 2020, pp. 11–21, DOI: 10.1109/DAPPS49028.2020.00002 (cit. on p. 49).

[CMC21] Paula Chanfreut, José M. Maestre, and Eduardo F. Camacho, « A Survey on Clustering Methods for Distributed and Networked Control Systems », *in*: *Annual Reviews in Control* 52 (2021), pp. 75–90, ISSN: 1367-5788, DOI: 10.1016/j.arcontrol.2021.08.002 (cit. on pp. 62, 155).

[CMI18] P. Chanfreut, J. M. Maestre, and H. Ishii, « Vulnerabilities in Distributed Model Predictive Control based on Jacobi-Gauss Decomposition », *in*: *2018 European Control Conference (ECC)*, June 2018, pp. 2587–2592, DOI: 10.23919/ECC.2018.8550239 (cit. on pp. 59, 66, 67, 152, 156, 157).

[CNN22] Andrea Camisa, Ivano Notarnicola, and Giuseppe Notarstefano, « Distributed Primal Decomposition for Large-Scale Milps », *in*: *IEEE Trans. Autom. Control* 67.*1* (Jan. 2022), pp. 413–420, ISSN: 2334-3303, DOI: 10.1109/tac.2021.3057061 (cit. on pp. 40, 144).

[Coh78] G. Cohen, « Optimization By Decomposition and Coordination: a Unified Approach », *in*: *IEEE Trans. Autom. Control* 23.*2* (1978), pp. 222–232, DOI: 10.1109/TAC.1978.1101718 (cit. on pp. 40, 144).

[Con+06] Antonio J Conejo et al., *Decomposition techniques in mathematical programming: engineering and science applications*, Springer Science & Business Media, 2006 (cit. on pp. 35, 37, 40, 65, 71, 143, 144, 158).

[Con10] Juan Pablo Conti, « The Day the Samba Stopped », *in*: *Engineering & Technology* 5.*4* (2010), pp. 46–47, DOI: 10.1049/et.2010.0410 (cit. on pp. 25, 138).

[DBG17] Xiang Dai, Romain Bourdais, and Hervé Guéguen, « Dynamic Reduction of The Iterations Requirement in A Distributed Model Predictive Control », *in*: 2017, DOI: 10.1109/CDC40024.2019.9029783 (cit. on pp. 133, 200).

[DI15] Seyed Mehran Dibaji and Hideaki Ishii, « Consensus of Second-Order Multi-Agent Systems in the Presence of Locally Bounded Faults », *in*: *Systems & Control Letters* 79 (2015), pp. 23–29, DOI: 10.1016/j.sysconle.2015.02.005 (cit. on pp. 48, 61, 153).

[Dib+19] Seyed Mehran Dibaji et al., « A Systems and Control Perspective of Cps Security », *in*: *Annual Reviews in Control* 47 (2019), pp. 394–411, ISSN: 1367-5788, DOI: 10.1016/j.arcontrol.2019.04.011 (cit. on pp. 25, 54, 56, 138).

[Din+18] Derui Ding et al., « A Survey on Security Control and Attack Detection for Industrial Cyber-Physical Systems », *in*: *Neurocomputing* 275 (2018), pp. 1674–1683, ISSN: 0925-2312, DOI: 10.1016/j.neucom.2017.10.009 (cit. on pp. 25, 59, 60, 138, 153).

[DLR77] Arthur P Dempster, Nan M Laird, and Donald B Rubin, « Maximum Likelihood From Incomplete Data Via the Em Algorithm », *in*: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.*1* (1977), pp. 1–22 (cit. on pp. 120, 190).

[EPE22] EPE, *Balanço Energético Nacional 2022: Ano base 2021*, tech. rep., 2022 (cit. on p. 49).

[Fer+01] G. Ferrari-Trecate et al., « Identification of Piecewise Affine and Hybrid Systems », *in*: *Proceedings of the 2001 American Control Conference. (Cat. No.01CH37148)* (2001), DOI: 10.1109/acc.2001.946178 (cit. on p. 120).

[FLT22] Giuseppe Franzè, Walter Lucia, and Francesco Tedesco, « Resilient Model Predictive Control for Constrained Cyber-Physical Systems Subject To Severe Attacks on the Communication Channels », *in*: *IEEE Transactions on Automatic Control* 67.*4* (2022), pp. 1822–1836, DOI: 10.1109/TAC.2021.3084237 (cit. on pp. 58, 152).

[For+16] Nicola Forti et al., « A Bayesian approach to joint attack detection and resilient state estimation », *in*: *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016, pp. 1192–1198, DOI: 10.1109/CDC.2016.7798428 (cit. on pp. 62, 154).

[FS10] Susana Faria and Gilda Soromenho, « Fitting Mixtures of Linear Regressions », *in*: *J Stat Comput Sim* 80.*2* (Feb. 2010), pp. 201–225, ISSN: 1563-5163, DOI: 10.1080/00949650802590261 (cit. on pp. 122, 125, 191, 194).

[GDU02] M.M. Gouda, S. Danaher, and C.P. Underwood, « Building Thermal Model Reduction Using Nonlinear Constrained Optimization », *in*: *Build Environ* 37.*12* (2002), pp. 1255–1265, ISSN: 0360-1323, DOI: 10.1016/S0360-1323(01)00121-4 (cit. on pp. 99, 175).

[Gis+13] Pontus Giselsson et al., « Accelerated Gradient Methods and Dual Decomposition in Distributed Model Predictive Control », *in*: *Automatica* 49.*3* (2013), pp. 829–833, ISSN: 0005-1098, DOI: 10.1016/j.automatica.2013.01.009 (cit. on pp. 35, 143).

[GP21] Alexander Gepperth and Benedikt Pfülb, « Gradient-Based Training of Gaussian Mixture Models for High-Dimensional Streaming Data », *in*: *Neural Processing Letters* 53.*6* (Aug. 2021), pp. 4331–4348, ISSN: 1573-773X, DOI: 10.1007/s11063-021-10599-3 (cit. on p. 121).

[GPM89] Carlos E. García, David M. Prett, and Manfre Morari, « Model Predictive Control: Theory and Practice a Survey », *in*: *Automatica, Vol. 25, No. 3, pp. 335-348*, 1989, DOI: 10.1016/0005-1098(89)90002-2 (cit. on pp. 21, 137).

[Gri+20] Paul Griffioen et al., « Secure Networked Control for Decentralized Systems via Software Rejuvenation », *in*: *2020 American Control Conference (ACC)*, 2020, pp. 1266–1273, DOI: 10.23919/ACC45564.2020.9147232 (cit. on pp. 60, 153).

[GS10] Federica Garin and Luca Schenato, « A Survey on Distributed Estimation and Control Applications Using Linear Consensus Algorithms », *in*: *Networked Control Systems*, ed. by Alberto Bemporad, Maurice Heemels, and Mikael Johansson, London: Springer London, 2010, pp. 75–107, ISBN: 978-0-85729-033-5, DOI: 10.1007/978-0-85729-033-5_3 (cit. on pp. 47, 87, 147, 165).

[Hes09] João P. Hespanha, *Linear Systems Theory*, ISBN13: 9780691179575, Princeton, New Jersey: Princeton Press, Feb. 2009, DOI: 10.23943/9781400890088 (cit. on pp. 87, 165).

[HML22] Ning He, Kai Ma, and Huiping Li, « Resilient Predictive Control Strategy of Cyber-Physical Systems Against Fdi Attack », *in*: *IET Control Theory & Applications* n/a.*n/a* (2022), DOI: 10.1049/cth2.12289 (cit. on pp. 32, 141).

[Hu+21] Songlin Hu et al., « Co-Design of Dynamic Event-Triggered Communication Scheme and Resilient Observer-Based Control Under Aperiodic Dos Attacks », *in*: *IEEE Transactions on Cybernetics* 51.*9* (2021), pp. 4591–4601, DOI: 10.1109/TCYB.2020.3001187 (cit. on pp. 61, 154).

[Hus+21] Bilal Hussain et al., « Deep Learning-Based Ddos-Attack Detection for Cyber-Physical System Over 5g Network », *in*: *IEEE Transactions on Industrial Informatics* 17.*2* (2021), pp. 860–870, DOI: 10.1109/TII.2020.2974520 (cit. on pp. 57, 62, 154).

[HZ16] Andreas Hoehn and Ping Zhang, « Detection of covert attacks and zero dynamics attacks in cyber-physical systems », *in*: *2016 American Control Conference (ACC)*, 2016, pp. 302–307, DOI: 10.1109/ACC.2016.7524932 (cit. on pp. 58, 62, 151, 154).

[Iid19] H. Iiduka, « Distributed Optimization for Network Resource Allocation With Nonsmooth Utility Functions », *in*: *IEEE Trans. Control Netw. Syst.* 6.*4* (2019), pp. 1354–1365, DOI: 10.1109/tcns.2018.2889011 (cit. on pp. 40, 144).

[Jen03] Erica Jen, « Stable Or Robust? What's the Difference? », *in*: *Complexity* 8.*3* (2003), pp. 12–18, DOI: 10.1002/cplx.10077 (cit. on pp. 60, 153).

[Jur62] E. I. Jury, « A Simplified Stability Criterion for Linear Discrete Systems », *in*: *Proc. IRE* 50.*6* (June 1962), pp. 1493–1500, ISSN: 2162-6634, DOI: 10.1109/JRPROC.1962.288193 (cit. on pp. 87, 165).

[KT13] Jinsub Kim and Lang Tong, « On Topology Attack of a Smart Grid: Undetectable Attacks and Countermeasures », *in*: *IEEE Journal on Selected Areas in Communications* 31.*7* (2013), pp. 1294–1305, DOI: 10.1109/JSAC.2013.130712 (cit. on pp. 57, 151).

[KV17] Nir Kshetri and Jeffrey Voas, « Hacking Power Grids: a Current Problem », *in*: *Computer* 50.*12* (2017), pp. 91–95, DOI: 10.1109/MC.2017.4451203 (cit. on pp. 62, 154).

[KX03] Dimitris Karlis and Evdokia Xekalaki, « Choosing Initial Values for the Em Algorithm for Finite Mixtures », *in*: *Computational Statistics & Data Analysis* 41.*3* (2003), Recent Developments in Mixture Model, pp. 577–590, ISSN: 0167-9473, DOI: 10.1016/S0167-9473(02)00177-9 (cit. on pp. 121, 125, 194).

[Lan11] Ralph Langner, « Stuxnet: Dissecting a Cyberwarfare Weapon », *in*: *IEEE Security Privacy* 9.*3* (2011), pp. 49–51, DOI: 10.1109/MSP.2011.67 (cit. on pp. 25, 57, 138, 151).

[LB19] Fabien Lauer and Gérard Bloch, « Hybrid System Identification », *in*: *Hybrid System Identification: Theory and Algorithms for Learning Switching Models*, Cham: Springer International Publishing, 2019, pp. 77–101, ISBN: 978-3-030-00193-3, DOI: 10.1007/978-3-030-00193-3_4 (cit. on pp. 109, 120, 181).

[LCK20] Xiaonan Lu, Mark Cannon, and Denis Koksal-Rivet, « Robust Adaptive Model Predictive Control: Performance and Parameter Estimation », *in*: *Int J Robust Nonlin* n/a.*n/a* (2020), DOI: 10.1002/rnc.5175, URL: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rnc.5175 (cit. on pp. 62, 155).

[LGG21] Walter Lucia, Kian Gheitasi, and Mohsen Ghaderi, « Setpoint Attack Detection in Cyber-Physical Systems », *in*: *IEEE Trans. Autom. Control* 66.*5* (2021), pp. 2332–2338, DOI: 10.1109/TAC.2020.3004326 (cit. on pp. 62, 154).

[Liu+10] Jinfeng Liu et al., « Sequential and iterative architectures for distributed model predictive control of nonlinear process systems. Part I: Theory », *in*: *Proceedings of the 2010 American Control Conference*, 2010, pp. 3148–3155, DOI: 10.1109/ACC.2010.5531017 (cit. on pp. 46, 147).

[LMC09] Jinfeng Liu, David Muñoz de la Peña, and Panagiotis D. Christofides, « Distributed Model Predictive Control of Nonlinear Process Systems », *in*: *AIChE Journal* 55.*5* (2009), pp. 1171–1184, DOI: 10.1002/aic.11801 (cit. on pp. 45, 146).

[Mae+21] José M. Maestre et al., « Scenario-Based Defense Mechanism Against Vulnerabilities in Lagrange-Based Dmpc », *in*: *Control Eng Pract* 114 (2021), p. 104879, ISSN: 0967-0661, DOI: 10.1016/j.conengprac.2021.104879 (cit. on pp. 61, 66, 67, 82, 96, 153, 156, 157, 162, 171).

[McN+18] P. McNamara et al., « Life lessons from and for distributed MPC – Part 1: Dynamics of cooperation », *in*: *IFAC-PapersOnLine* 51.*30* (2018), 18th IFAC Conference on Technology, Culture and International Stability TECIS 2018, pp. 101–106, ISSN: 2405-8963, DOI: 10.1016/j.ifacol.2018.11.256, URL: https://www.sciencedirect.com/science/article/pii/S240589631832929X (cit. on pp. 47, 147).

[MMC11] J. M. Maestre, D. Muñoz de la Peña, and E. F. Camacho, « Distributed model predictive control based on a cooperative game », *in*: *Optimal Control Applications and Methods* 32.*2* (2011), pp. 153–176, DOI: 10.1002/oca.940 (cit. on pp. 35, 60, 143).

[MN+14] José M Maestre, Rudy R Negenborn, et al., *Distributed Model Predictive Control made easy*, vol. 69, Springer, 2014, ISBN: 978-94-007-7005-8 (cit. on pp. 40, 144).

[Mor+11] Petru-Daniel Moroşan et al., « A Distributed Mpc Strategy Based on Benders' Decomposition Applied To Multi-Source Multi-Zone Temperature Regulation », *in*: *Journal of Process Control* 21.*5* (2011), Special Issue on Hierarchical and Distributed Model Predictive Control, pp. 729–737, ISSN: 0959-1524, DOI: 10.1016/j.jprocont.2010.12.002 (cit. on pp. 40, 144).

[MS09] Yilin Mo and Bruno Sinopoli, « Secure control against replay attacks », *in*: *2009 47th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2009, pp. 911–918, DOI: 10.1109/ALLERTON.2009.5394956 (cit. on pp. 56, 62, 154).

[MTI18] José M. Maestre, Paul A. Trodden, and Hideaki Ishii, « A Distributed Model Predictive Control Scheme with Robustness Against Noncompliant Controllers », *in*: *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 3704–3709, DOI: 10.1109/CDC.2018.8619079 (cit. on pp. 62, 154, 155).

[MWS15] Yilin Mo, Sean Weerakkody, and Bruno Sinopoli, « Physical Authentication of Control Systems: Designing Watermarked Control Inputs To Detect Counterfeit Sensor Outputs », *in*: *IEEE Control Syst. Mag.* 35.*1* (2015), pp. 93–109, DOI: 10.1109/MCS.2014.2364724 (cit. on pp. 62, 154).

[Nak08] Satoshi Nakamoto, *Bitcoin: A peer-to-peer electronic cash system*, 2008, URL: http://bitcoin.org/bitcoin.pdf (cit. on p. 48).

[NBG21] Rafael Accácio Nogueira, Romain Bourdais, and Hervé Guéguen, « Detection and Mitigation of Corrupted Information in Distributed Model Predictive Control Based on Resource Allocation », *in*: *2021 5th Conference on Control and Fault-Tolerant Systems (SysTol)*, 2021, pp. 329–334, DOI: 10.1109/SysTol52990.2021.9595927 (cit. on pp. 26, 140).

[NM14] R. R. Negenborn and J. M. Maestre, « Distributed Model Predictive Control: an Overview and Roadmap of Future Research Opportunities », *in*: *IEEE Control Syst. Mag.* 34.*4* (Aug. 2014), pp. 87–97, ISSN: 1941-000X, DOI: 10.1109/MCS.2014.2320397 (cit. on pp. 45, 146).

[Nog+22] Rafael Accácio Nogueira et al., « Expectation-Maximization Based Defense Mechanism for Distributed Model Predictive Control », *in*: *IFAC-PapersOnLine* 55.*13* (2022), 9th IFAC Conference on Networked Systems NECSYS 2022, pp. 73–78, ISSN: 2405-8963, DOI: 10.1016/j.ifacol.2022.07.238 (cit. on pp. 26, 140).

[NTK05] Hayato Nakada, Kiyotsugu Takaba, and Tohru Katayama, « Identification of Piecewise Affine Systems Based on Statistical Clustering Technique », *in*: *Automatica* 41.*5* (2005), pp. 905–913, ISSN: 0005-1098, DOI: 10.1016/j.automatica.2004.12.005 (cit. on p. 120).

[OF10] Alexey Ozerov and Cédric Fevotte, « Multichannel Nonnegative Matrix Factorization in Convolutive Mixtures for Audio Source Separation », *in*: *IEEE Audio, Speech, Language Process.* 18.*3* (2010), pp. 550–563, DOI: 10.1109/TASL.2009.2031510 (cit. on pp. 125, 194).

[Ola+18] S. Olaru et al., « Life lessons from and for distributed MPC – Part 2: Choice of decision makers », *in*: *IFAC-PapersOnLine* 51.*30* (2018), 18th IFAC Conference on Technology, Culture and International Stability TECIS 2018, pp. 107–111, ISSN: 2405-8963, DOI: 10.1016/j.ifacol.2018.11.257 (cit. on pp. 47, 147).

[OV14] Daniel O'Connor and Lieven Vandenberghe, « Primal-Dual Decomposition By Operator Splitting and Applications To Image Deblurring », *in*: *SIAM J. Imaging Sci.* 7.*3* (2014), pp. 1724–1754, DOI: 10.1137/13094671X (cit. on pp. 40, 144).

[Pau+16] R. Paulen et al., « Primal and dual decomposition for distributed MPC - Theory, implementation, and comparison in a SoS simulation framework », *in*: *2016 24th Mediterranean Conference on Control and Automation (MED)*, June 2016, pp. 286–291, DOI: 10.1109/MED.2016.7536022 (cit. on pp. 40, 144).

[PBB12] F. Pasqualetti, A. Bicchi, and F. Bullo, « Consensus Computation in Unreliable Networks: a System Theoretic Approach », *in*: *IEEE Trans. Autom. Control* 57.*1* (2012), pp. 90–104, DOI: 10.1109/TAC.2011.2158130 (cit. on pp. 62, 154).

[PDB13] F. Pasqualetti, F. Dörfler, and F. Bullo, « Attack Detection and Identification in Cyber-Physical Systems », *in*: *IEEE Trans. Autom. Control* 58.*11* (Nov. 2013), pp. 2715–2729, ISSN: 1558-2523, DOI: 10.1109/TAC.2013.2266831 (cit. on pp. 58, 62, 151, 154).

[Qia+22] Tiantian Qian et al., « N-1 Static Security Assessment Method for Power Grids With High Penetration Rate of Renewable Energy Generation », *in*: *Electric Power Systems Research* 211 (2022), p. 108200, ISSN: 0378-7796, DOI: 10.1016/j.epsr.2022.108200 (cit. on pp. 61, 153).

[RBL04] Jacob Roll, Alberto Bemporad, and Lennart Ljung, « Identification of Piecewise Affine Systems Via Mixed-Integer Programming », *in*: *Automatica* 40.*1* (2004), pp. 37–50, ISSN: 0005-1098, DOI: 10.1016/j.automatica.2003.08.006 (cit. on p. 120).

[RD22] Martin Roeck and Thomas Drennen, « Life cycle assessment of behind-the-meter Bitcoin mining at US power plant », *in*: *The International Journal of Life Cycle Assessment* 27.*3* (2022), pp. 355–365, ISSN: 1614-7502, DOI: 10.1007/s11367-022-02025-0 (cit. on p. 49).

[Rie10] Craig G. Rieger, « Notional examples and benchmark aspects of a resilient control system », *in*: *2010 3rd International Symposium on Resilient Control Systems*, 2010, pp. 64–71, DOI: 10.1109/ISRCS.2010.5603123 (cit. on pp. 60, 153).

[SBR22] Lucas Schulze, Douglas W. Bertol, and Guilherme V. Raffo, « Fast Computation of Binary Search Tree for Pwa Functions Representation Using Intersection Classification », *in*: *Automatica* (2022), p. 110217, ISSN: 0005-1098, DOI: 10.1016/j.automatica.2022.110217 (cit. on p. 84).

[SK17] B. Satchidanandan and P. R. Kumar, « Dynamic Watermarking: Active Defense of Networked Cyber-Physical Systems », *in*: *Proc. IEEE* 105.*2* (2017), pp. 219–240, DOI: 10.1109/JPROC.2016.2575064 (cit. on pp. 62, 154).

[Smi15] Roy S. Smith, « Covert Misappropriation of Networked Control Systems: Presenting a Feedback Structure », *in*: *IEEE Control Systems Magazine* 35.*1* (2015), pp. 82–92, DOI: 10.1109/MCS.2014.2364723 (cit. on pp. 58, 151).

[Sta88] M.G. Staskauskas, « The Formal Specification and Design of a Distributed Electronic Funds-Transfer System », *in*: *IEEE Transactions on Computers* 37.*12* (1988), pp. 1515–1528, DOI: 10.1109/12.9730 (cit. on p. 46).

[Sun+22] Ziwen Sun et al., « Adaptive Event-Triggered Resilient Control of Industrial Cyber Physical Systems Under Asynchronous Data Injection Attack », *in*: *Journal of the Franklin Institute* 359.*7* (2022), pp. 3000–3023, ISSN: 0016-0032, DOI: 10.1016/j.jfranklin.2022.02.009 (cit. on pp. 61, 154).

[SY19] Yuan-Cheng Sun and Guang-Hong Yang, « Robust Event-Triggered Model Predictive Control for Cyber-Physical Systems Under Denial-Of-Service Attacks », *in*: *International Journal of Robust and Nonlinear Control* 29.*14* (2019), pp. 4797–4811, DOI: https://doi.org/10.1002/rnc.4654, eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rnc.4654 (cit. on pp. 57, 58, 61, 151, 152, 154).

[Tay+21] Michael Taylor et al., « Model Predictive Control of Smart Districts With Fifth Generation Heating and Cooling Networks », *in*: *IEEE Transactions on Energy Conversion* 36.*4* (2021), pp. 2659–2669, DOI: 10.1109/TEC.2021.3082405 (cit. on pp. 21, 35, 137).

[Tei+12] André Teixeira et al., « Revealing stealthy attacks in control systems », *in*: *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2012, pp. 1806–1813, DOI: 10.1109/Allerton.2012.6483441 (cit. on pp. 58, 151).

[Tei+15] A. Teixeira et al., « A Secure Control Framework for Resource-Limited Adversaries », *in*: *Automatica* 51 (2015), pp. 135–148, DOI: 10.1016/j.automatica.2014.10.067 (cit. on pp. 56–58, 60, 62, 150, 151, 153, 154).

[VE22] Petros G. Voulgaris and Nicola Elia, « When Selfish is Socially Optimal », *in*: *IEEE Transactions on Automatic Control* 67.*5* (2022), pp. 2359–2372, DOI: 10.1109/TAC.2021.3081795 (cit. on pp. 45, 145).

[Vel+17a] Pablo Velarde et al., « Scenario-based defense mechanism for distributed model predictive control », *in*: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, IEEE, Dec. 2017, pp. 6171–6176, DOI: 10.1109/CDC.2017.8264590 (cit. on pp. 59, 61, 66, 67, 152, 153, 156, 157).

[Vel+17b] Pablo Velarde et al., « Vulnerabilities in Lagrange-Based DMPC in the Context of Cyber-Security », *in*: *2017 IEEE International Conference on Autonomic Computing (ICAC)*, July 2017, pp. 215–220, DOI: 10.1109/ICAC.2017.53 (cit. on pp. 59, 61, 66, 67, 76, 152, 156, 157).

[Vel+18] Pablo Velarde et al., « Vulnerabilities in Lagrange-Based Distributed Model Predictive Control », *in*: *Optimal Control Applications and Methods* 39.*2* (Sept. 2018), pp. 601–621, DOI: 10.1002/oca.2368 (cit. on pp. 40, 59, 61, 62, 66, 67, 72, 77, 82, 96, 144, 152, 153, 155–157, 159, 160, 162, 171).

[VSK14] S. Vahid Naghavi, A.A. Safavi, and M. Kazerooni, « Decentralized fault tolerant model predictive control of discrete-time interconnected nonlinear systems », *in*: *Journal of the Franklin Institute* 351.*3* (2014), pp. 1644–1656, ISSN: 0016-0032, DOI: 10.1016/j.jfranklin.2013.12.005, URL: https:

//www.sciencedirect.com/science/article/pii/S001600321300433X (cit. on pp. 45, 60, 145, 153).

[WDL16] Hao Wu, Xianglei Dang, and Bin Li, « A topology attack on averaging consensus convergence », *in*: *2016 Chinese Control and Decision Conference (CCDC)*, 2016, pp. 3343–3348, DOI: 10.1109/CCDC.2016.7531560 (cit. on pp. 57, 151).

[WI19] Yuan Wang and Hideaki Ishii, « A Distributed Model Predictive Scheme for Resilient Consensus with Input Constraints », *in*: *2019 IEEE Conference on Control Technology and Applications (CCTA)*, 2019, pp. 349–354, DOI: 10.1109/CCTA.2019.8920700 (cit. on pp. 61, 153).

[WL13] Wenye Wang and Zhuo Lu, « Cyber Security in the Smart Grid: Survey and Challenges », *in*: *Computer Networks* 57.*5* (2013), pp. 1344–1371, ISSN: 1389-1286, DOI: 10.1016/j.comnet.2012.12.017 (cit. on pp. 57, 59, 152).

[XBK07] Lin Xiao, Stephen Boyd, and Seung-Jean Kim, « Distributed Average Consensus With Least-Mean-Square Deviation », *in*: *J Parallel Distr Com* 67.*1* (2007), pp. 33–46, ISSN: 0743-7315, DOI: 10.1016/j.jpdc.2006.08.010 (cit. on pp. 87, 165).

[Xie+16] Lei Xie et al., « Ga Based Decomposition of Large Scale Distributed Model Predictive Control Systems », *in*: *Control Engineering Practice* 57 (2016), pp. 111–125, ISSN: 0967-0661, DOI: 10.1016/j.conengprac.2016.08.016 (cit. on pp. 35, 143).

[Yan+19] Hongjiu Yang et al., « MPC-based Defense Strategy for Distributed Networked Control Systems Under Dos Attacks », *in*: *Systems & Control Letters* 128 (2019), pp. 9–18, ISSN: 0167-6911, DOI: 10.1016/j.sysconle.2019.04.001 (cit. on pp. 57, 151).

[YZG22] Jun Yang, Wen-An Zhang, and Fanghong Guo, « Adaptive Distributed Kalman-Like Filter for Power System With Cyber Attacks », *in*: *Automatica* 137 (Mar. 2022), p. 110091, ISSN: 0005-1098, DOI: 10.1016/j.automatica.2021.110091 (cit. on pp. 62, 155).

[Zac+19] Y. Zacchia Lun et al., « State of the Art of Cyber-Physical Systems Security: an Automatic Control Perspective », *in*: *J Syst Software* 149 (2019), cited By 44, pp. 174–216, DOI: 10.1016/j.jss.2018.12.006 (cit. on p. 25).

[Zha+20] Ling Zhao et al., « Optimal Power Allocation for Multiple Dos Attackers in Wireless Networked Control Systems », *in*: *ISA Trans.* 104 (2020), pp. 204–211, ISSN: 0019-0578, DOI: 10.1016/j.isatra.2019.01.006 (cit. on pp. 57, 151).

[Zha+21a] Dan Zhang et al., « A Survey on Attack Detection, Estimation and Control of Industrial Cyber-Physical Systems », *in*: *ISA Transactions* 116 (2021), pp. 1–16, ISSN: 0019-0578, DOI: 10.1016/j.isatra.2021.01.036 (cit. on pp. 62, 154).

[Zha+21b] Yuan Zhang et al., « Economic Model Predictive Control for the Operation Optimization of Water Distribution Networks With Risks », *in*: *Asian J Control* 23.*1* (2021), pp. 128–142, DOI: 10.1002/asjc.2218, URL: https://onlinelibrary.wiley.com/doi/pdf/10.1002/asjc.2218 (cit. on pp. 21, 32, 35, 137).

[ZLW21] Hang Zhang, Bo Liu, and Hongyu Wu, « Smart Grid Cyber-Physical Attack and Defense: a Review », *in*: *IEEE Access* 9 (2021), pp. 29641–29659, DOI: 10.1109/ACCESS.2021.3058628 (cit. on pp. 57, 58, 151).

[ZM11] M. Zhu and S. Martínez, « Stackelberg-game analysis of correlated attacks in cyber-physical systems », *in*: *Proceedings of the 2011 American Control Conference*, 2011, pp. 4063–4068, DOI: 10.1109/ACC.2011.5991463 (cit. on p. 60).

[ZM14] M. Zhu and S. Martínez, « On the Performance Analysis of Resilient Networked Control Systems Under Replay Attacks », *in*: *IEEE Trans. Autom. Control* 59.*3* (2014), pp. 804–808, DOI: 10.1109/tac.2013.2279896 (cit. on pp. 56, 150).

DOCTORAT BRETAGNE LOIRE / MATHSTIC

CentraleSupélec

Titre : Sécurité de la Commande Prédictive distribuée sous injection de données faussées

Mot clés : Sécurité de systèmes cyber-physiques, Commande Prédictive distribuée, Injection de données faussées

Résumé : Les dispositifs informatiques sont partout, ils commandent systèmes de différentes échelles, dès drones de la taille d'une pièce aux systèmes de distribution d'énergie de la taille des pays. La mixture de ces dispositifs avec des systèmes physiques essentiels pour la vie quotidienne origine les systèmes cyber-physiques. Dépendant de l'échelle des systèmes, la computation est divisée en différents dispositifs, qui nécessitent de négocier pour trouver un consensus de la meilleure commande à être appliquée. La communication entre agents est d'extrême importance pour que le consensus soit achevé. Par contre, la négociation est sensible à déviations dans les valeurs échangées. An attaquant peut utiliser cette information pour influencer la négociation pour bénéficier un groupe d'agents. Cette thèse propose une étude de la conception de la commande de tels systèmes jusqu'à leur sécurité. Et après, on propose des méthodes pour sécuriser le système contre des attaques où l'agent malintentionné injecte de la fausse information pendant la négociation. La performance des méthodes est étudiée en utilisant des simulations d'un petit réseau de distribution de chaleur.

Title : Security of distributed Model Predictive Control under False Data Injection

Keywords: Security of cyber-physical systems, distributed Model Predictive Control, False Data Injection

Abstract: Computing devices are everywhere, controlling systems of different scales, from drones of the size of a coin to country-wise energy distribution grids. The amalgamation of these devices and the physical systems essential for our everyday lives originates the cyber-physical systems. Depending on their scale, the computation is divided into multiple computing devices, which need to negotiate to find a consensus of the best control to be applied. The communication among agents is vital for the consensus to be achieved. However, the negotiation is sensitive to deviations in the exchanged values. An attacker can use this information to influence the negotiation to benefit a group of agents. This thesis proposes a study from the conception of the control of such systems to its security. And then, we propose methods to secure the system from attacks where an ill-intentioned agent injects false data into the negotiation. The performance of the proposed methods is assessed through academic simulations of a small district heating network.