Commande prédictive non-linéaire : application à la production d'énergie

Manon Fouquet

N° d'ordre : 2016-03-TH

CentraleSupélec

Ecole Doctorale MATISSE

« Mathématiques, Télécommunications, Informatique, Signal, Systèmes Electroniques »

Laboratoire de IETR

THÈSE DE DOCTORAT

DOMAINE : STIC
Spécialité : Automatique

Soutenue le 30 mars 2016

par :

Manon FOUQUET

Commande prédictive non linéaire – Application à la production d'énergie

Composition du jury :

Directeur de thèse :	Hervé GUÉGUEN	Professeur (CentraleSupélec)
Co-directeur de thèse :	Didier DUMUR	Professeur (CentraleSupélec)
Président du jury :	Gérard BLOCH	Professeur (Université Lorraine, Nancy)
Rapporteurs :	Nicolas LANGLOIS	Professeur (ESIGELEC Rouen)
	Eric BIDEAUX	Professeur (INSA Lyon)
Examinateurs :	Damien FAILLE	Ingénieur Chercheur Expert (EDF)
	Alina VODA	Maître de Conférences (Université Grenoble Alpes)
Membres invités :	Stéphane VELUT	Ingénieur (Modelon)

Remerciements

Pendant cette thèse, une de mes meilleures amies a fait naître une petite fille et m'a raconté que cela avait été la douleur la plus intense qu'elle ait jamais connue, mais qu'une fois l'enfant posée sur elle, le souvenir même de la souffrance s'était évanoui. En écrivant ces remerciements quelques jours après la soutenance de ce manuscrit, je ressens un peu la même chose, certes à une tout autre échelle. De ces trois années de travail il ne me reste que ce texte, avec ses qualités et ses défauts, que je vais accepter de ne plus retoucher une fois de plus. Déjà les instants de doute et de fatigue n'ont plus jamais existé et laissent place à la joie d'avoir petit à petit appris des choses et gagné en sérénité, même si ces processus lents de transformation ont souvent la pudeur de passer inaperçus. Je dois dire que j'ai eu la chance d'être entourée pendant toute cette période de personnes formidables qui m'ont permis de traverser presque sans encombre cette épreuve personnelle et intellectuelle tant redoutée au début. Toutes ces personnes ont su par leur bienveillance ou leur curiosité protéger mon optimisme de la corrosion, et me guider sur des chemins que je n'aurais peut être pas osé prendre seule.

Je voudrais commencer par remercier mes encadrants Damien Faille, Hervé Guéguen et Didier Dumur qui malgré leurs nombreux engagements ont toujours été présents quand j'en avais besoin, sans jamais me priver d'autonomie. Je suis heureuse d'avoir pu mener ce projet à bout avec vous. Merci Damien pour ta curiosité (sans faille !), ces longues discussions où les idées fusaient à une vitesse que je n'ai pas toujours réussi à suivre, et pour cet esprit de questionnement qui m'a permis de clarifier beaucoup de choses. Merci Hervé et merci Didier pour avoir tous deux appuyé mon travail de votre intelligence et de votre rigueur avec toujours beaucoup de gentillesse. Le terme de tuteur que je vous réserve m'a toujours fait sourire car dans ma tête il a gardé son sens de soutien pour les pieds de tomates. Mais d'une certaine façon j'en garde l'analogie, vous m'avez permis de ne pas pousser dans tous les sens comme je peux en avoir la tendance, et de développer mes efforts au bon endroit pour y voir plus haut. Qui sais, la pluie de Rennes a peut être aidé elle aussi !

Je voudrais maintenant remercier mes deux rapporteurs, Nicolas Langlois et Eric Bideaux, pour avoir pris le temps de lire ce manuscrit, pour leurs questions et leurs commentaires pertinents qui m'ont permis de lui apporter les derniers coups de pinceaux, ainsi que mes examinateurs Gérard Bloch et Alina Voda. Je remercie également Stéphane Velut de Modelon d'être venu tout droit de Suède pour assister à ma soutenance.

Je voudrais maintenant témoigner ma sympathie à toute l'équipe du groupe P12 d'EDF qui m'a accueillie pendant cette thèse, la bonne ambiance qui y règne m'a été précieuse. Merci à Marie et Khadra pour leur gestion attentionnée et leur confiance, qui contribue pour beaucoup à un climat favorable aux échanges et à la recherche. Durant ces trois années, aller au travail a rarement été une contrainte. Merci à mes voisins de bureau, Thomas, Jessica, Edouard, Bruno et Loïc pour leur bonne humeur, et leur mauvaise humeur aussi parfois qui par un habile système de gages nous faisait gagner un goûter de temps à autre (je ne citerai personne !). Et à tous les autres qui ont aussi été pour moi plus des amis que des collègues, Côme, Denis, Romain, Cécile, Guillaume, Jennifer, Pierre-Louis, Frans, Sylvain, Lorenzo, Louis, Coralie, Antoine, tous les Brunos, Astrid... Je remercie aussi Alexandre dont le génie m'a souvent intimidée malgré des retours toujours constructifs, au point que je regrette un peu de ne pas avoir plus échangé sur mon travail. Je garderai à la place le souvenir de parties de mots-croisés farouches le midi ! Toute ma sympathie va aussi aux équipes de Rennes et de Gif pour m'avoir si bien reçue, merci Romain, Pierre, Maxime, Khang, Antoine et Marjorie pour toutes ces discussions intéressantes.

During my thesis, I also spent three months in Lund, Sweden. I would like to thank warmfully all the people at Modelon and Lund university that helped me a lot with the implementation of some methods in JModelica.org. It has been a really productive time. A special mention goes to Fredrik, Toivo, Stéphane and Per-Ola who followed my work with interest and gave me very insightful comments. Thank you also to all those who made my stay there so pleasant, Robert, Lorenzo, Lars, Ylva, Usa, Johan, Anna, Anders, Matthias, Olga and all those I may have forgotten.

Je veux maintenant remercier tous mes amis qui ont fait la contingence et la beauté d'aujourd'hui. Pour beaucoup, ces rencontres se sont faites dans des espaces-temps qui ne sont plus cet ici et maintenant, mais qui ont façonné le chemin qui m'y a menée. Merci Raphaël pour ta folie douce, Jeanne, Pauline, Anaïs et Adrien pour votre joie de vivre qui déborde, pour les rêveurs à l'âme pure, Thomas, Sébastien, Mathilde, Florence, Robert, Mehdi, Marina, Camille, pour mes enfants terribles et attachants, Martin, Boris, Gouge, Hugo, Sophie, Alex, Pierre, Marta, Julien, Paola, Piol, Benoît, JC, Nico, Hélios, Natacha, Jérem, Coco, Célia, les amis "suédois" de Moun, la promo du mastère OSE et plus généralement tous ceux rencontrés ces dernières années. J'espère n'oublier personne. Vous comptez pour moi malgré le temps et la distance.

Je témoigne aussi ma profonde reconnaissance à tous les membres de ma famille qui ont toujours cru en moi et m'ont donné la force de ne pas reculer devant la peur de l'échec. Merci papa de m'avoir toujours laissé voler de mes propres ailes malgré ta peur qu'en me spécialisant trop je perde un peu de mon âme. Je crois qu'il m'en reste encore. Merci maman de m'avoir appris à aller jusqu'au bout de ce que l'on entreprend. Merci Quentin d'avoir égayé ces années par ta musique. Merci papy et mamie pour votre exemplarité et votre courage malgré des temps un peu plus durs en ce moment.

Je terminerai par toi Mounir, qui t'es peut être plus fatigué que moi pour me rendre la vie simple pendant ce travail. On peut dire que tu as aussi soutenu à ta façon cette thèse et je suis en admiration devant une telle constance, un tel dévouement et une telle humilité. Tu as une force et un rayonnement rare. Le jour de la soutenance cela faisait quatre ans que tu étais entré dans ma vie. Je suis heureuse de pouvoir encore avancer à tes côtés.

Table des matières

Table des figures

Liste des Algorithmes

Acronymes

AD	Automatic Differentiation
BB	Branch and Bound
BE	Backward Euler
BLT	Block Lower Triangular
CHP	Combined Heat and Power
CNF	Conjonctive Normal Form
DAE	Differential Algebraic Equations
DNF	Disjunctive Normal Form
EDF	Électricité de France
EDP	Équations aux Dérivées Partielles
FE	Forward Euler
GAM	Generalized Additive Model
GDP	Generalized Disjunctive Programming
H-DAE	Hybrid Differential Algebraic Equations
IAPWS	International Association for the Properties of Water and Steam
IAPWS-IF97	IAPWS Industrial Formulation 1997
LICQ	Linear Independence Constraint Qualification
LMI	Linear Matrix Inequalities
LTV	Linear Time Varying
MFCQ	Mangasarian-Fromovitz Constraint Qualification
MILP	Mixed Integer Linear Programming
MINLP	Mixed Integer Nonlinear Programming
MPC	Model Predictive Control
MPCC	Mathematical Problem with Complementarity Constraints
NCP	Non Linear Complementarity Problem

NLP	Non Linear Program
NMPC	Nonlinear Model Predictive Control
OA	Outer Approximation
OCP	Optimal Control Problem
ODE	Ordinary Differential Equations
PLNE	Programmation Linéaire en Nombres Entiers
PID	Proportionnel Intégrateur Dérivateur
RK	Runge Kutta
RK4	Runge Kutta d'ordre 4
SI	International System of Units
TL-MPC	Tangent Linear Model Predictive Control
SOCP	Second Order Cone Programming
SOS1	Special Ordered Set of type 1
SUR	Sum Up Rounding
SURminT	Sum Up Rounding with Minimum switching Times

Nomenclature

Quantités physiques

A	Surface [m^2]
C	Prix normalisé [p.u]
M	Masse [kg]
P	Pression [Pa]
T	Température [^{o} C]
V	Volume [kg]
W	Puissance [W]
h	Enthalpie spécifique [J/kg]
q	Débit massique [kg/s]
s	Entropie spécifique [J/kg]
u	Énergie interne spécifique [J/kg]
v	Volume spécifique [m^3/kg]
ρ	Masse volumique [kg/m^3]

Types de variables

p	Paramètre (dimension n_p)
u	Commande continue (dimension n_u, entrée contrôlée)
x	État (dimension n_x)
$\dot{x}$	Dérivée de l'état (dimension n_x), parfois dx
w	Perturbation (dimension n_w, entrée non contrôlée)
w^m	Perturbation mesurée (dimension n_v)
y	Sortie (dimension n_y), ou variable logique en 3.3
z	Variable algébrique (dimension n_z)
ω	Commande logique (dimension n_ω)

Évolution du système

F	Fonction de l'équation d'état implicite dans (DAEi)
F_k	Fonction de l'équation d'évolution discrète du modèle linéarisé tangent
$\bar{F}_k$	Fonction de l'équation d'évolution discrète du modèle linéarisé tangent dans le cas nominal
f	Fonction de l'équation d'état dans (ODE)
$f^{(d)}$	Fonction de l'équation d'évolution discrète du modèle à temps continu
$\bar{g}$	Fonction de l'équation algébrique explicite dans (ODE)
g	Fonction de l'équation algébrique implicite dans (DAEse)
h	Fonction de l'équation des sorties dans (NLc)
t_0	Temps initial
t_f	Temps final
δf	Fonction de l'équation dynamique du modèle linéarisé tangent à temps continu
ϕ	Fonction de transition pour les systèmes hybrides

Optimisation dynamique - Chapitre 3

$\mathcal{H}$	Hamiltonien
$\mathcal{L}$	Lagrangien
$\mathcal{L}_f$	Lagrangien au temps final
h_I	Fonction des contraintes inégalité
h_E	Fonction des contraintes égalité
l	Terme intégral dans la fonction objectif
J	Fonction objectif
Φ	Coût terminal dans la fonction objectif
λ	État adjoint

Méthode de collocation hybride - Chapitre 3

$\mathcal{M}(n_j)$	Ensemble des modes admissibles pour une disjonction de n_j termes
T	Grille temporelle donnée par la durée de ses éléments
T_k	Durée de l'élément k
T_{pre}	Durée depuis laquelle les modes sont actifs à l'instant initial
T^j_{min}	Durée d'activation minimale des modes de la disjonction j
$\mathcal{T}$	Grille temporelle donnée par ses instants
$\mathcal{T}_0$	Grille temporelle des éléments de la méthode de collocation
$\mathcal{T}_1$	Grille temporelle des points de collocation
$\tilde{X}$	Trajectoire prédite par la méthode de collocation
$\hat{X}$	Valeur estimée
c_{up}	Coût de transition à la hausse
m	Mode actif

m_{pre}	Mode actif à l'instant initial
n_c	Nombre de points de collocation par élément
n_d	Nombre de disjonctions
n_e	Nombre d'éléments
$\mathbf{s}$	Grille temporelle donnée par ses indices communs avec la grille principale
tol_J	Tolérance sur la différence entre l'objectif relaxé et binaire
tol_ϵ	Tolérance sur la précision de la méthode de collocation
ϵ_{rel}	Erreur relative sur les variables
ζ_{rel}	Erreur relative sur la dynamique
τ	Temps local sur un élément
τ_c	Instant de collocation
ω_k^{ji}	Variables binaires discrétisées pour les disjonctions SOS1
ω_k^j	Variables binaires discrétisées pour les disjonctions binaires

Indices

bin	Binaire
c	Point de collocation
k	Élément
i	Terme d'une disjonction
j	Disjonction
n	Itération
rel	Relaxé
sim	Simulé

Commande prédictive - Chapitre 4

$\mathcal{K}$	Retour d'état implicite de la commande MPC
$\mathcal{K}_f$	Contrôleur terminal
P	Horizon de prédiction du MPC
$\mathbb{P}$	Trajectoire du système
T_s	Pas d'échantillonnage
U^i	Vecteur de commande du problème MPC_i
$\mathcal{U}$	Ensemble admissible pour u
x_a	État augmenté
X^i	États prédits pour le problème MPC_i
$\mathcal{X}$	Ensemble admissible pour x
Y^i	Sorties prédites pour le problème MPC_i
δ	Variation par rapport au nominal
$\delta\mathcal{X}_k$	Ensemble admissible pour δx_k
$\delta\mathcal{U}_k$	Ensemble admissible pour δu_k
Δ	Variation entre deux pas de temps
Ω	Ensemble terminal pour δx

Indices

i	Instant courant
k	Instant de l'horizon de prédiction
mes	Mesuré
nom	Nominal
r	Référence

CHAPITRE 1

Introduction

1.1 Contexte et motivations de la thèse

On s'intéresse dans cette thèse à la conduite optimisée de centrales de production d'énergie pour lesquelles on dispose de modèles physiques. L'utilisation de modèles basés sur les premiers principes physiques (aussi appelés modèles de connaissance boîte blanche) s'est beaucoup développée ces dernières années dans le domaine des procédés et de la production d'énergie, poussée par le développement de langages de modélisation de plus en plus sophistiqués et de compilateurs de plus en plus performants. EDF développe dans cette optique une librairie nommée ThermoSysPro pour la simulation statique et dynamique de centrales d'énergie. Les modèles physiques développés possèdent de nombreux avantages par rapport à des modèles identifiés boîte noire ou des modèles linéaires simplifiés : ils permettent en effet de simuler le comportement d'une installation avec précision sur toute sa plage de fonctionnement, et de prédire l'évolution de grandeurs physiques [3][4]. Les modèles physiques peuvent aussi être utilisés dans une optique de surveillance : le modèle physique, calibré lors du fonctionnement normal de l'installation sert alors de référence pour détecter un comportement anormal de la centrale, ou une variation de certains paramètres [5]. Enfin, les modèles physiques peuvent également être utilisés dans une optique de dimensionnement : dans ce cas, on recherchera les caractéristiques des composants permettant d'aboutir à un point de fonctionnement donné [6].

Une piste encore peu explorée concerne l'utilisation des modèles physiques pour le pilotage optimisé des installations, qui vise à la fois l'optimisation (hors-ligne) du planning de production et la commande de l'installation en temps réel. Les modèles physiques ont été utilisés dans ce contexte chez [7][8][9][10][11]. Le besoin en pilotage optimisé s'est récemment accru depuis la dérèglementation et la mise en concurrence du marché de l'énergie, qui modifient progressivement les règles du jeu pour les producteurs, avec notamment l'introduction de prix de l'électricité variant dans le temps, la tendance vers des moyens de production décentralisés, la gestion simultanée de différentes commodités (électricité, chaleur, vapeur,...), l'introduction de stockages, ou encore l'apparition de nouvelles opportunités de services-système (réserve secondaire). Ces divers aspects introduisent une dimension temporelle qui impacte fortement

l'optimisation économique d'une installation.

Optimisation du planning de production

Aujourd'hui, l'optimisation du planning de production est souvent réalisée en utilisant la Programmation Linéaire en Nombres Entiers (PLNE) car c'est une méthode qui permet de modéliser un grand nombre de problèmes, de prendre en compte des aspects discrets (démarrages et arrêts), et qui offre de plus des garanties de convergence vers une solution globalement optimale. EDF développe dans ce cadre un logiciel nommé PILOT : l'installation y est modélisée par des stockages et des composants capables de transformer des commodités (électricité, chaleur, combustible, etc.) en d'autres commodités selon des courbes de rendement affines par morceaux. La plupart du temps, les grandeurs considérées dans ces modèles sont ainsi décrites par des couples puissance/énergie ou débits/volume, mais rarement par des grandeurs physiques telles que des pressions ou des températures. Or ces grandeurs du système sont souvent soumises à des contraintes : en conséquence, un planning optimal fourni par un modèle de PLNE, réalisable d'un point de vue énergétique, peut s'avérer infaisable dans l'installation réelle (on doit alors vérifier la faisabilité par simulation *a posteriori*). L'utilisation de modèles physiques pour l'optimisation prend alors tout son sens car elle permet de formuler explicitement les contraintes sur ces variables. Cependant cette approche a été peu considérée jusqu'alors en raison de la complexité des modèles mis en jeu : il s'agit de modèles non linéaires avec souvent des aspects hybrides (le système peut se trouver dans différents modes). Des méthodes adaptées, prenant en compte ces deux spécificités doivent êtres développées. En particulier, la complexité du modèle d'optimisation doit souvent être réduite par rapport à un modèle de simulation détaillé, et la prise en compte d'aspects discrets dans un cadre non linéaire devra être considérée.
Dans cette thèse, on développe une méthodologie d'optimisation dynamique pour une classe de modèles hybrides qui inclut les centrales de production d'énergie étudiées. Cette méthodologie fournit des trajectoires optimisées et permet de prendre en compte des contraintes à la fois sur les variables continues et sur les aspects discrets du problème. Le formalisme permet ainsi de considérer des plages de fonctionnement différentes pour chacun des modes du système ainsi que des durées minimales de marche et d'arrêt pour les composants.

Commande des centrales d'énergie

L'autre piste complémentaire étudiée dans cette thèse concerne l'utilisation de modèles physiques pour la commande des centrales d'énergie, et en particulier la commande prédictive. La commande prédictive, ou MPC, est une technique de commande fondée sur l'optimisation d'un modèle de l'installation sur un horizon glissant. Le MPC est particulièrement intéressant car il permet de prendre en compte des contraintes sur les commandes (comme la plupart des régulateurs classiques) mais aussi et surtout sur les sorties de l'installation. Son principe est le suivant : à chaque pas d'échantillonnage de la commande, un problème d'optimisation sous contraintes est résolu sur un horizon de prédiction fini, et le premier terme de cette séquence de commande optimale est appliqué au système. Cette procédure est ensuite répétée aux pas d'échantillonnage suivants.
On pourrait en théorie appliquer toute la séquence de commande optimisée. Cependant, de

nombreux facteurs tels que des perturbations agissant sur le système, ou encore des erreurs de modélisation, font que le système réel n'évolue pas exactement comme prévu lors de l'optimisation. Le fait de recalculer à chaque pas de temps une nouvelle commande prenant en compte l'état du système est ainsi une façon d'introduire une rétroaction dans le système. Cependant, on peut noter que par rapport à une régulation classique (PID, ...), la commande ne s'exprime en général plus de façon explicite, mais comme le résultat d'une séquence de problèmes d'optimisation. Cette définition implicite de la commande apporte une complexité évidente dans l'analyse de stabilité du système en boucle fermée.
Par ailleurs le MPC introduit des problématiques liées à son application en temps réel : il s'agit de garantir que le problème d'optimisation peut être résolu dans un temps compatible avec le pas d'échantillonnage. La convergence de l'optimisation, ou *a minima* l'obtention d'une solution faisable est ainsi nécessaire, faute de quoi la commande devient indéterminée. C'est une des raisons pour lesquelles la grande majorité des algorithmes de commande prédictive implémentés considèrent un modèle linéaire, des contraintes linéaires et une fonction objectif quadratique. Le problème résultant est un problème convexe qui peut être résolu en temps polynomial. Dans le cas non linéaire en revanche, les problèmes d'optimisation sont souvent non convexes, et aucune garantie ne peut être donnée sur le temps de résolution, qui reste fortement lié à l'initialisation des variables. Des techniques spécifiques peuvent alors être mises en place pour respecter la contrainte temps réel, en jouant par exemple sur la structure du problème, la parallélisation ou encore l'initialisation d'un problème MPC d'après la solution précédente (*warm start*). Une introduction aux problématiques du MPC non linéaire (NL-MPC) peut par exemple être trouvée dans [12][13].
Pour éviter les difficultés posées par le MPC non linéaire, on développe dans cette thèse une commande MPC à mi-chemin entre le MPC linéaire et non linéaire : le modèle Modelica détaillé de la centrale y est utilisé pour élaborer des modèles linéarisés tangents, menant à une commande MPC linéaire variant dans le temps avec coût quadratique que l'on dénomme TL-MPC. Cette commande est utilisée pour le suivi du planning optimisé de production, afin de corriger les écarts dus aux perturbations et erreurs de modélisation.

1.2 Architecture de commande proposée et organisation du manuscrit

Dans cette thèse, le chapitre 2 fait figure de chapitre introductif et traite de la modélisation et de la simulation des centrales de production d'énergie à l'aide du langage Modelica. On y rappelle à la fois des aspects concernant la modélisation physique mais également le formalisme mathématique employé par Modelica et les algorithmes utilisés pour, partant d'un jeu d'équations, aboutir à une trajectoire du système.
Le plan de cette thèse s'articule ensuite autour de l'architecture de commande proposée en Figure 1.1, qui a pour but la génération d'une trajectoire optimale pour l'installation d'une part, et le suivi de cette trajectoire optimale en temps réel d'autre part. Les différentes briques de cette architecture sont les suivantes :

1. **Outil de prévision**. Un outil de prévision génère les trajectoires de référence w_r pour les perturbations, qui sont des entrées exogènes du système (demande de chaleur, apports externes...). Ces prévisions sont utilisées par la suite lors de l'optimisation de trajectoire. Cette partie ne sera pas traitée dans cette thèse, aussi on considèrera par la

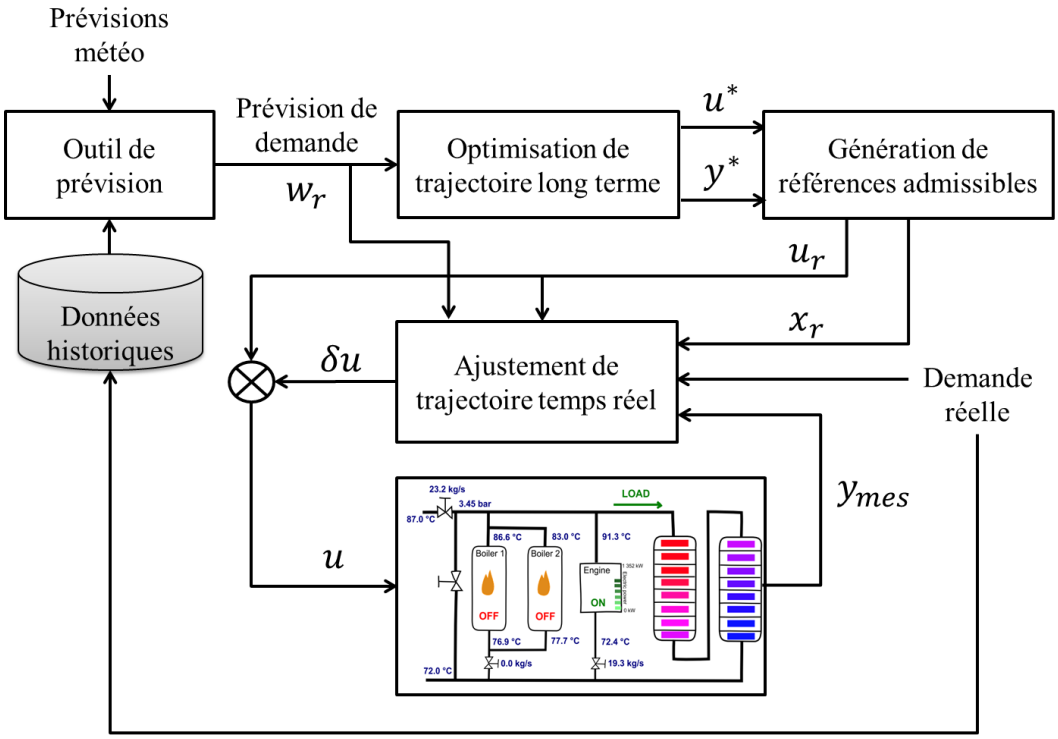


FIGURE 1.1 – Première architecture de commande

suite w_r comme une donnée d'entrée.

2. **Optimisation de trajectoire long terme**. L'optimisation du planning de production s'effectue sur un modèle physique simplifié ou un modèle de PLNE et fournit des trajectoires de référence optimales pour les entrées (u^*) et les sorties (y^*) sur un horizon H long. Le chapitre 3 est consacré au développement d'un algorithme d'optimisation dynamique hybride remplissant cette fonction.
3. **Génération de trajectoires de référence admissibles**. Cet aspect est aussi brièvement traité dans le chapitre 3. Comme le modèle d'optimisation est souvent un modèle simplifié, il ne définit pas nécessairement les variables de la même façon que le modèle détaillé de l'installation. Pour cette raison, les trajectoires optimales de référence doivent être redéfinies pour les variables du modèle détaillé. On applique alors une méthode permettant de générer des trajectoires de référence u_r, x_r, y_r respectivement pour les commandes, états et sorties du modèle non-linéaire, cohérentes avec y^*, u^*.
4. **Ajustement de trajectoire temps réel**. Le système réel est soumis à diverses perturbations, et diffère du modèle utilisé pour l'optimisation. Pour ces raisons, un ajustement δu doit être apporté à la trajectoire de commande u_r dans le but d'assurer au mieux le suivi de la trajectoire x_r ou y_r. On propose dans le chapitre 4 de remplir cette fonction par l'algorithme de commande prédictive TL-MPC. Celui-ci fonctionne avec un pas d'échantillonnage T_s et considère un horizon de prédiction de $P * T_s << H$.

Le chapitre 5 apporte enfin des conclusions sur la solution proposée et des perspectives d'amélioration.

CHAPITRE 2

Modélisation des centrales d'énergie

EDF développe depuis plusieurs années ThermoSysPro, une librairie de composants thermohydrauliques pour la modélisation de centrales d'énergie (thermique, nucléaire, solaire). Cette librairie de composants est écrite dans le langage de modélisation Modelica [14], un langage libre soutenu par l'association éponyme.
Ce chapitre est consacré à la modélisation à l'aide de Modelica des centrales d'énergie étudiées dans cette thèse. Après une brève introduction au langage Modelica en 2.1.1, on présente en 2.1.2 le formalisme de modélisation employé dans la librairie ThermoSysPro d'EDF : la modélisation 0D / 1D. La compréhension de cette modélisation est en effet nécessaire dans la perspective d'obtenir des modèles adaptés à l'optimisation, ce qui fera l'objet du chapitre 3.
On décrit ensuite plus généralement la formulation mathématique employée par Modelica en 2.2 : les systèmes d'équations algébro-différentielles (DAE). Les DAE regroupent une large classe de modèles, et englobent les équations différentielles ordinaires (ODE) comme un cas particulier.
Pour traiter les systèmes DAE, les compilateurs Modelica emploient une combinaison d'algorithmes de manipulations symboliques afin de mettre les modèles dans une forme adaptée pour les algorithmes d'intégration numérique. On présentera ces méthodes en 2.2.2 car elles nous permettront d'obtenir des modèles dans une forme adaptée à la commande, qui sera exploitée dans les chapitres 3 et 4, puis l'on donne quelques rappels sur les algorithmes d'intégration numérique en 2.2.3.1.
Enfin, afin d'illustrer les concepts introduits, on présente deux cas d'étude : un premier modèle de compresseur issu de la librairie ThermoSysPro et un second modèle simplifié de centrale de cogénération, qui sera repris par la suite dans le chapitre 3.

2.1 Modélisation des systèmes étudiés

2.1.1 Présentation de Modelica

Modelica est un langage pour la simulation de systèmes multi-physiques, ce qui est typiquement le cas des centrales d'énergie que nous allons étudier, dans lesquelles peuvent se

retrouver des aspects hydrauliques, de la combustion, de la neutronique dans le cas des installations nucléaires, ou encore de l'optique dans le cas des centrales solaires. Modelica est un langage orienté objet où la construction de modèles complexes (une centrale de production d'énergie) se fait par assemblage de composants élémentaires (une vanne, une turbine,etc.). Les composants possèdent pour cela des connecteurs, qui jouent le rôle d'interface avec les autres composants du système, leur permettant d'échanger deux types d'informations : des grandeurs intensives (pression, température), et des flux traversant la connexion (débits, intensités). On introduit d'ores et déjà la notation "." (point) utilisée par Modelica pour décrire cette composition des systèmes à l'aide de composants élémentaires. A titre d'exemple, la variable V présente dans le connecteur C d'un composant E sera notée

$$E.C.V$$

La connexion de deux composants E_1 et E_2 génère des équations de connexions :

- $E_1.C.V = E_2.C.V$ si V est une variable intensive
- $E_1.C.V + E_2.C.V = 0$ si V est une variable de flux

D'un point de vue utilisateur, Modelica est un langage équationnel (dit aussi déclaratif) : les modèles y sont décrits par un ensemble d'équations différentielles, algébriques et discrètes, dans un formalisme nommé *Hybrid DAE* qui sera présenté en 2.2. On verra dans la partie 2.1.2 que ce formalisme est tout a fait adapté à la modélisation retenue pour les centrales d'énergie.
L'intérêt d'un langage équationnel comme Modelica est que la représentation déclarative permet de décrire comment le système fonctionne sans se soucier de la façon dont les variables sont calculées. Cette tâche est laissée au programme de compilation, dont l'objectif est de transformer le jeu d'équations en un code exécutable par une machine. La compilation consiste en un traitement symbolique des équations du modèle permettant de les réorganiser et de calculer les valeurs des variables de façon séquentielle. Elle permet également de transformer le modèle sous une forme adaptée aux algorithmes d'intégration numérique utilisé. Ces diverses manipulations du modèle par le compilateur permettent à l'utilisateur de se concentrer sur la partie modélisation à proprement parler, sans avoir à réfléchir aux aspects de résolution numérique du problème.

La contrepartie de cet avantage indéniable est qu'il est beaucoup plus difficile de tracer les erreurs dans le modèle. La mise à plat et le traitement automatisé de toutes les équations empêchent en effet tout déboggage pas à pas par l'utilisateur lors de la compilation ou de la simulation. Il est donc important de comprendre les diverses manipulations effectuées par le compilateur pour pouvoir comprendre les erreurs qu'il affiche, et qui peuvent parfois être difficiles à interpréter.
Par ailleurs, les méthodes développées dans la suite de cette thèse nécessitent que les modèles soient donnés sous forme d'Equations Différentielles Ordinaires (ODE), et c'est justement ce que permettent les algorithmes introduits précédemment.
On donne donc dans ce chapitre une présentation succincte de ces algorithmes de traitement symbolique en partie 2.2.2 en s'appuyant sur des exemples simples de réseaux hydrauliques.

2.1.2 Modélisation 0D/1D des centrales d'énergie

On introduit dans cette partie le formalisme de modélisation utilisé pour représenter les centrales d'énergie étudiées dans cette thèse : la modélisation 0D/1D.

Les centrales d'énergies sont des installations thermo-hydrauliques utilisant divers fluides de travail. On considère ici des modèles physiques de ces installations, c'est-à-dire des modèles capables de représenter les températures, pressions, débits et puissances au sein des différents composants. On ne s'intéressera que dans certains cas particuliers à la distribution spatiale de ces propriétés. Afin d'obtenir des modèles de taille raisonnable mais suffisamment précis pour fournir le niveau d'information souhaité, la modélisation retenue pour ces installations est la modélisation dite 0D/1D, en opposition aux codes de simulation 2D/3D utilisés en mécanique des fluides.

Définition (Modèle 0D/1D) Un modèle 0D est un modèle ignorant la notion de dimension spatiale pour le composant considéré. Cette simplification est souvent réalisée en considérant des valeurs moyennes ou des lois de conservation macroscopiques. Dans un modèle 1D, une dimension spatiale est considérée, résultant la plupart du temps de la discrétisation du phénomène par une méthode par différences finies.

D'autres simplifications peuvent parfois être utilisées pour réduire la complexité du modèle, lorsque certains phénomènes peuvent être négligés aux échelles de temps considérées :

- Équivalence lent/constant : les variables évoluant très lentement peuvent être considérées comme des paramètres constants pour le système.
- Équivalence rapide/instantané : les variables ayant des constantes de temps courtes atteignent rapidement un état d'équilibre. Il est alors possible de considérer des équations statiques (autrement dit à l'équilibre) pour ces équations, ce qui permet d'éliminer également des variables différentielles.

Enfin, la complexité peut être réduite en ne modélisant que la physique nécessaire. Il n'est par exemple pas nécessaire de modéliser la conservation des espèces chimiques pour un fluide ne subissant pas de transformation chimique.

2.1.2.1 Choix des variables d'état

L'état thermodynamique d'un fluide peut être décrit par plusieurs jeux de variables. Ainsi, en l'absence de réaction chimique, il est par exemple possible d'utiliser indifféremment le couple pression/enthalpie spécifique (P, h), le couple masse volumique/énergie interne (ρ, u) ou encore le couple masse volumique/enthalpie spécifique (ρ, h). Le couple (P, T) peut être employé pour les fluides qui demeurent monophasiques pendant toute la période étudiée. Cependant, lors d'un changement de phase, en raison du palier de température qui se produit, la donnée (P, T) n'est plus suffisante pour caractériser l'état du fluide et il faut alors préciser la fraction de vapeur α du fluide, un entier compris entre 0 et 1 (ou utiliser le couple (P, h)).

2.1.2.2 Schéma à grille décalée (*Staggered Grid*)

Les modèles de centrales étudiés sont construits par assemblage de composants élémentaires et tirent ainsi parti de la modélisation orientée objet de Modelica. Le fonctionnement de chaque composant, régi par des lois de conservation, est formulé au sein de chaque composant par un jeu d'équations. A celles-ci s'ajoutent des équations dites de connexion. En effet, lorsque deux composants sont assemblés, ceux-ci partagent au niveau de la connexion des variables intensives (pression, température, tension, etc.) et des variables de flux (débit, vitesse, intensité, etc). On distingue principalement deux types de composants :

- Les composants bi-ports (turbines, valves, échangeurs, canalisations, etc.)

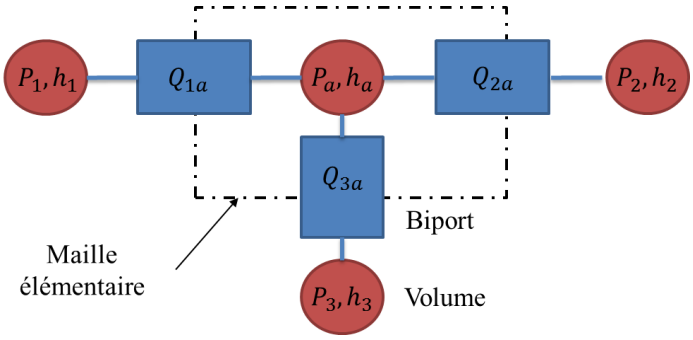


FIGURE 2.1 – Schéma thermohydraulique à grille décalée

– Les éléments de volume (nœuds de mélange, ballons, etc.)

Le schéma dit à grille décalée est utilisé dans la librairie ThermoSysPro. Dans ce schéma, les quantités intensives sont calculées au sein des volumes où l'on suppose qu'un mélange parfait existe (et donc que l'état thermodynamique peut être décrit par des propriétés moyennes). Les variables de flux sont quant à elles calculées aux jonctions entre ces volumes (c'est de là que vient le nom de grille décalée). Chaque composant bi-ports doit donc être relié à des composants de volume, comme illustré en Figure 2.1.

Le schéma à grille décalée est également utilisé à l'intérieur de certains composants où la modélisation d'une dimension spatiale présente un intérêt. C'est notamment le cas de composants tels que les échangeurs ou encore les stockages thermiques, pour lesquels le profil de température est important (stratification dans le stockage, pincement pour l'échangeur). Le composant est discrétisé spatialement en un nombre fini de volumes sous l'hypothèse de propriétés homogènes au sein de chaque volume (*lumped approximation*). Les Équations aux Dérivées Partielles (EDP) qui décrivent le modèle sont ainsi converties en un système ODE ou DAE qui sera par la suite résolu de la même façon que le reste du système.

2.1.2.3 Lois de conservation sur un volume de contrôle

La modélisation des systèmes thermo-hydrauliques s'appuie sur les lois de conservation de la masse M, de l'énergie E et de la quantité de mouvement I au sein d'un volume V donné que l'on appelle volume de contrôle. Dans le cas des systèmes à plusieurs phases (comme un mélange eau/vapeur), ces lois de conservations s'appliquent à chacune des phases et une description des échanges entre phases doit être ajoutée. Les équations de conservation présentées ci-dessous permettent de modéliser l'évolution d'un système fluide au cours du temps. Sous l'hypothèse 0D/1D, le fluide est unidirectionnel et un tronçon élémentaire de contrôle est montré en Figure 2.2. De manière générale [1], le taux de variation d'une quantité extensive Ψ au sein du volume de contrôle s'obtient en intégrant sur celui-ci la variable intensive (par unité de masse) $\psi = \frac{d\Psi}{dM}$ correspondante.

$$\frac{d\Psi}{dt} = \frac{d}{dt}\int_V \rho\psi dV = \int_V \frac{\partial(\rho\psi)}{\partial t} dV + \int_A \rho\psi\vec{\omega_A}\vec{n}dA \tag{2.1}$$

avec ω_a la vitesse de déplacement de la frontière de surface A et $\vec{n}$ son vecteur normal. Il existe deux approches pour le choix du volume de contrôle :

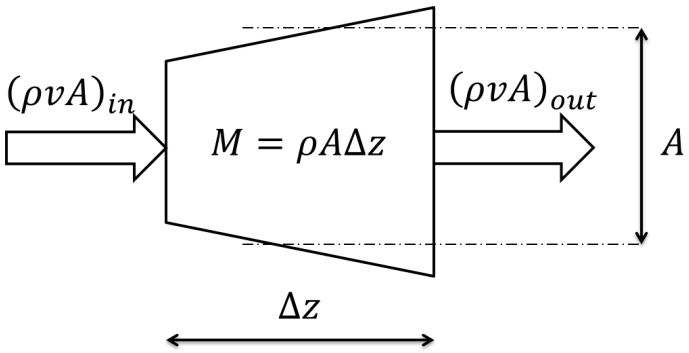


FIGURE 2.2 – Volume de contrôle d'après [1]

- Approche Lagrangienne : on considère que le volume de contrôle contient toujours les mêmes particules fluides. Dans ce cas, le volume est "mobile", sa surface se déplace à la vitesse des particules $\omega = \omega_a$ et le volume de contrôle a une masse constante : $\frac{dM}{dt} = 0$.
- Approche Eulérienne : le volume de contrôle est ici fixé : $\omega_a = 0$, et le deuxième terme de (2.1) s'annule. On retiendra par la suite cette approche qui correspond mieux à la modélisation composant par composant adoptée (on s'intéresse aux flux traversant les composants).

Les lois de conservation de la quantité extensive Ψ dans le volume de contrôle relient son taux de variation aux flux entrants et sortants. Par exemple, la loi de conservation de la masse nous indique que la variation de masse dans un volume de contrôle fixe est liée aux débits massiques q sortant (négatifs) et entrant (positifs) dans ce volume. On dérive ci-dessous les équations de conservation sur un volume fixe V, ainsi que leur version statique dans le cas où l'échelle de temps des phénomènes étudiés permet de négliger ces transitoires.

1. Conservation de la masse.
   La conservation de la masse correspond au cas $\psi = 1$.

$$\frac{dM}{dt} = \frac{d(\rho V)}{dt} = V\frac{d\rho}{dt} = \sum_{j=1}^{n_p} q_j \tag{2.2}$$

   La version statique de la conservation de la masse se réduit à $0 = \sum_{j=1}^{n_p} q_j$.

2. Conservation de l'énergie.
   La conservation de l'énergie pour un système fermé est quant à elle dérivée du premier principe de la thermodynamique, qui indique que la variation de l'énergie totale est égale à l'échange de travail W et de chaleur Q avec le milieu extérieur : $E = W + Q$. On montre ici comment dériver la loi de conservation de l'énergie dans un cas simplifié où l'on néglige les énergies cinétique et potentielle du fluide (hypothèse souvent vérifiée dans les systèmes étudiés [1]) : on a ainsi $E \approx U$ avec U l'énergie interne. La conservation de l'énergie correspond alors au cas $\psi = u$, et sous l'hypothèse que u est constante dans le volume de contrôle on peut écrire $E = \rho V u$. On montre ici l'équation de conservation de l'énergie pour les couples de variables d'état (ρ, u) et (P, h). Le passage des variables (ρ, u) aux variables (P, h) utilise les relations suivantes : $\rho = f(P, h)$ et $h = u + Pv$,

avec v le volume spécifique du fluide valant $v = 1/\rho$.

$$\frac{d\rho}{dt} = \left.\frac{\partial\rho}{\partial P}\right|_h \frac{dP}{dt} + \frac{\partial\rho}{\partial h}|_P \frac{dh}{dt}$$
$$\frac{du}{dt} = \frac{d(h-Pv)}{dt} = \frac{dh}{dt} - \frac{d\frac{P}{\rho}}{dt}$$

La conservation de l'énergie s'écrit :

$$\begin{aligned}\sum_i q_i h_i + \sum_j Q_j + \sum_j W_j &= \frac{dE}{dt} \\ &= \frac{d(\rho V u)}{dt} \\ &= V\frac{d(\rho(h-\frac{P}{\rho}))}{dt} \\ &= V\frac{d(\rho h - P)}{dt} \\ &= V\left(h\left(\frac{\partial\rho}{\partial P}\frac{dP}{dt} + \frac{\partial\rho}{\partial h}\frac{dh}{dt}\right) + \rho\frac{dh}{dt} - \frac{dP}{dt}\right) \\ &= V\left(\left(h\frac{\partial\rho}{\partial P}|_h - 1\right)\frac{dP}{dt} + \left(\rho - h\left.\frac{\partial\rho}{\partial h}\right|_P\right)\frac{dh}{dt}\right)\end{aligned} \tag{2.3}$$

La version statique de la conservation de l'énergie est quant à elle

$$0 = \sum_{j=1}^{n_p} q_j h_j + P_{in} - P_{out} \tag{2.4}$$

Les termes sources P_{in} et P_{out} sont l'apport et le prélèvement d'énergie respectivement. Dans l'équation ci-dessus, les h_j sont les enthalpies spécifiques **en amont** de la branche, pris dans le sens effectif du débit. Cela pose en pratique certaines difficultés de modélisation lorsque le sens des débits dans les branches n'est pas connu *a priori* mais résulte d'une équation de conservation. On verra par la suite que la première approche qui consiste à traiter cette question par un test conditionnel sur le sens du débit est problématique à la fois en simulation et en optimisation.

3. De façon similaire, il est possible de dériver des équations de conservation pour la quantité de mouvement et la quantité de matière. On pourra se référer à [1].

2.1.2.4 Éléments bi-ports

Parmi les éléments biports se rencontrant fréquemment dans les centrales d'énergie, on peut citer les conduites, les turbines, les pompes, les échangeurs de chaleur, les vannes, et d'une façon générale tous les composants liés à une perte de charge. Dans la modélisation 0D/1D, l'expression de la perte de charge peut être fonction de paramètres agrégés (comme le rendement isentropique pour les pompes et turbines), ou donnée par des corrélations empiriques sur des grandeurs adimensionnelles (comme les nombres de Reynolds, de Nusselt, etc.). Les corrélations sont souvent utilisées pour déterminer le transfert de chaleur ou la perte de charge, lorsque la géométrie des composants est trop complexe pour en donner une formule explicite.

2.1.2.5 Modélisation des propriétés physiques

On présente ici succinctement comment sont modélisées les propriétés des trois principaux fluides que l'on rencontre dans les systèmes étudiés : l'eau / vapeur, les gaz et les combustibles.

Tables de l'eau

Le standard de modélisation pour le calcul des propriétés de l'eau / vapeur est donné par l'*International Association for the Properties of Water and Steam* (IAFPWS)[15]. Les tables IAPWS-IF97 sont découpées en 5 régions qui sont fonction de la température T et de la pression P comme montré en figure 2.3. Comme l'on peut le voir sur la figure, les limites

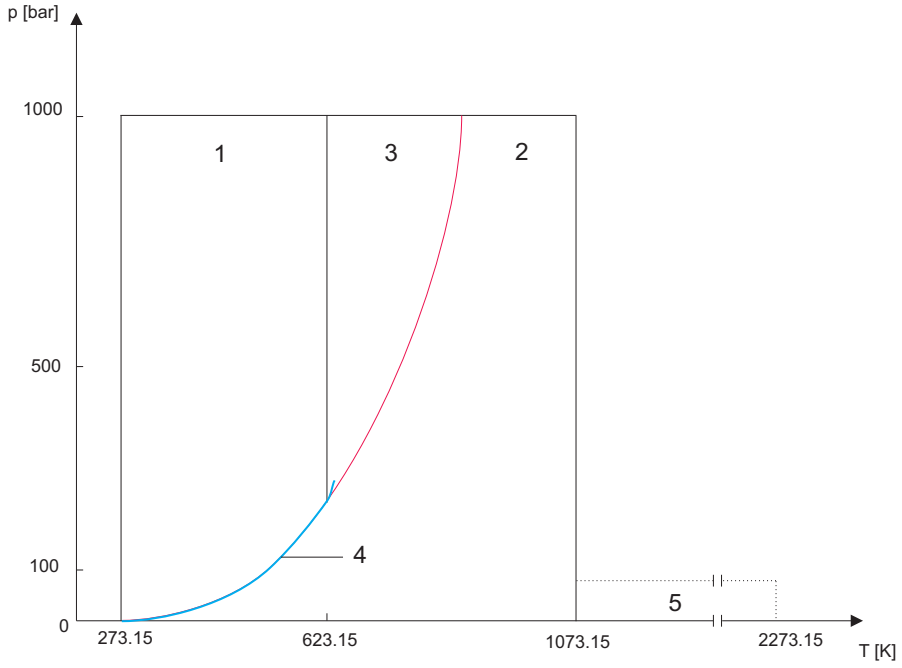


FIGURE 2.3 – Régions (P, T) des tables de l'eau IAPWS-IF97 d'après [2]

entre régions ont un comportement monotone pour les variables (P, T). Ainsi, les régions sont délimitées par les fonctions mono-variables $T_{14}(P)$, $T_{42}(P)$, $T_{32}(P)$ et les bornes T_{13}, T_{25}. Dans chacune de ces régions, des équations différentes sont ensuite utilisées pour le calcul des propriétés.
Cependant, comme mentionné précédemment, une subtilité dans la modélisation des propriétés est que plusieurs couples de variables thermodynamiques peuvent être utilisés pour décrire le fluide. Ainsi, le couple de variables retenues pour les connecteurs fluides est souvent le couple (P, h) et non le couple (P, T). Avec ces variables, l'arbre de décision pour déterminer dans quelle région se trouve le fluide n'est alors plus aussi direct comme on peut le voir en Figure 2.4.
Par ailleurs, il peut arriver dans certains cas que les équations se formulent plus facilement en fonction de variables autres que celles des connecteurs (P, h) (par exemple (P, T), (P, u) ou (P, s)). Dans ce contexte, un point fort des tables IAPWS-IF97 est de fournir des équations inverses sur chaque région r, par exemple $T = f_r(P, h)$, ce qui dans la pratique évite la

résolution de problèmes d'inversion qui nuisent à l'efficacité de la simulation.

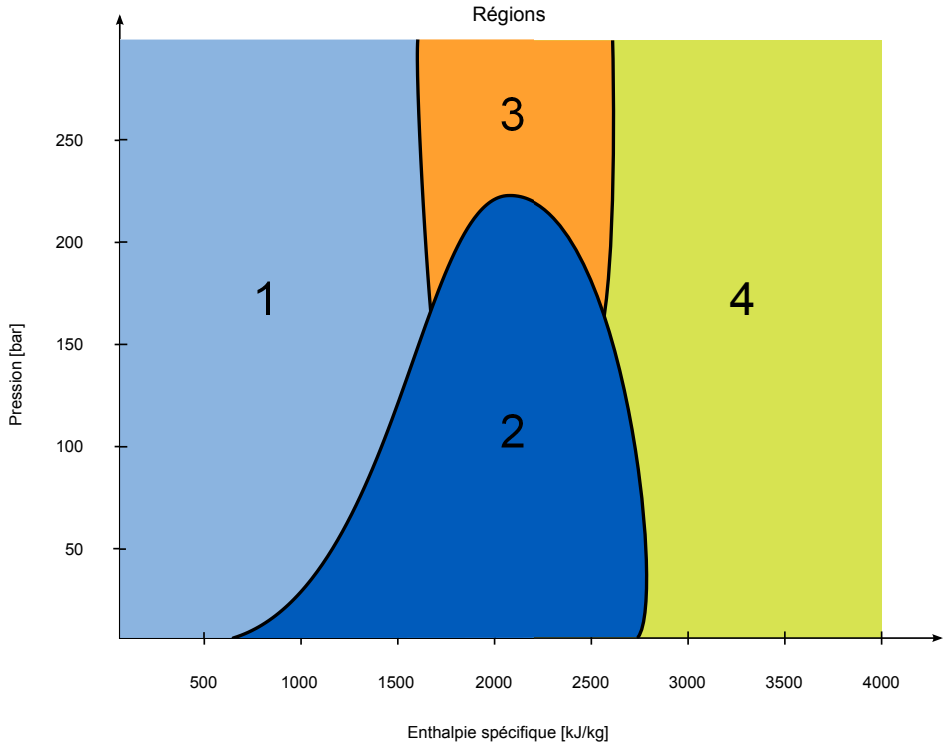


FIGURE 2.4 – Régions (P, h) des tables de l'eau IAPWS-IF97

Les systèmes étudiés dans cette thèse se limitent normalement aux régions 1 (eau liquide), 2 (vapeur sous-critique) et 4 (mélange eau/vapeur).

Gaz

Dans ThermoSysPro, les gaz (du moins ceux qui ne sont pas considérés comme des combustibles, à l'instar du gaz naturel) sont donnés par leur composition massique dans les quatre gaz O_2, CO_2, H_2O et SO_2. Ces données sont suffisantes pour la plupart des gaz étudiés qui sont souvent de l'air ou des fumées de combustion. Les propriétés sont calculées selon le standard NASA [16].
La librairie Media de la librairie standard Modelica offre quant à elle une description plus générale des fluides, en utilisant les capacités d'héritage permises par Modelica : tous sont décrits par un tableau de composition massique mais le calcul des propriétés est spécifique au fluide considéré. Avec cette approche, la même description est utilisée pour l'eau, les gaz et les autres fluides (fluides cryogéniques par exemple). Il devient également possible de modifier le fluide de travail dans un composant sans avoir à redéfinir celui-ci.

Combustibles

Dans les centrales d'énergie étudiées, la chaleur peut être produite par combustion entre un gaz comburant (le plus souvent de l'air) et un combustible. Dans ThermoSysPro, les combustibles sont décrits par leur Pouvoir Calorifique Inférieur (PCI) ainsi que leur composition

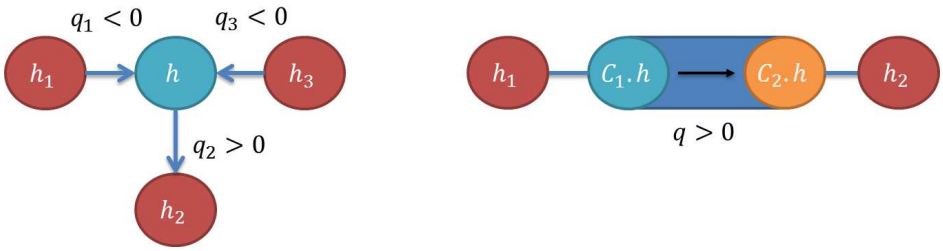


FIGURE 2.5 – Définition des sens positifs des débits. Noeud de mélange (gauche), Composant bi-ports (droite)

massique dans les éléments C, H, O, N et S. Ces composants sont en effet les constituants principaux des combustibles usuels. La réaction de combustion considérée peut être plus ou moins complexe. Dans ThermoSysPro, l'hypothèse est faite que les réactions de combustion sont faites en excès d'air, c'est-à-dire qu'il n'y a pas de formation de monoxyde de carbone dans les fumées (un gaz qui ne fait d'ailleurs pas partie des 4 gaz cités dans le paragraphe précédent). La réaction considérée est ainsi :

$$C_xH_yO_zN_rS_t + u0_2 + vN_2 \rightarrow aC0_2 + bH_2 + cS0_2 + dN_2 + eO_2$$

Lors d'une réaction de combustion, les lois de conservation de la masse s'appliquent sur chaque élément CHONS et sur la masse totale. Ces deux équations permettent de déterminer la masse de 0_2, $C0_2$, H_20 et $S0_2$ dans les fumées. La conservation de l'énergie prend quant à elle en compte le PCI du combustible et la variation d'enthalpie des composants formés lors de leur montée en température et permet de déterminer la température des fumées.

2.1.2.6 Modélisation des inversions de débit

La modélisation des inversions de débit requiert une attention particulière. On retient ici comme variables le couple (P, h) et l'on considère deux cas : les nœuds de mélange et les éléments biports. Dans un nœud de mélange idéalisé, on définit comme positifs les débits sortants du nœud (respectivement entrant dans les composants connectés). Une connexion de n composants correspond aux équations de conservation de la masse et de l'énergie suivantes :

$$0 = \sum_{p=1}^{n} q_p$$

$$0 = \sum_{p=1}^{n} q_p * \begin{cases} h, & \textbf{si } q_p > 0 \\ h_p, & \textbf{si } q_p \leq 0 \end{cases}$$

Où h est l'enthalpie au nœud de mélange et h_p l'enthalpie en amont du composant p. Cette situation est représentée sur la figure de gauche en Figure 2.5.
La première façon de modéliser cette connexion, et qui est celle retenue dans la librairie ThermoSysPro, est une transcription directe du jeu d'équations suivant, en utilisant des branches conditionnelles **if**. h et h_p sont ici définies au sein des connecteurs fluide, et leur valeur est déterminée par des équations au sein des composants auxquels ils appartiennent.
Dans le cas de deux composants connectés à un nœud statique (n=2), il est aisé de constater

que h est discontinu lors d'une inversion de débit, passant directement de h_1 à h_2 lorsque le débit devient négatif. On constate également que lorsque le débit est nul ($q_1 = q_2 = 0$), l'équation de conservation de l'énergie est satisfaite pour n'importe quelle valeur de h et il n'est plus possible de la définir uniquement. On parle de singularité. Ces deux aspects peuvent poser problème lors de la résolution du système. Pour cette raison, ThermoSysPro propose pour la plupart des composants une régularisation pour des débits inférieurs à un débit minimal q_ϵ. La convention est qu'au sein des composants de perte de charge, on ne calcule l'enthalpie de mélange h que pour le connecteur en amont du débit. Pour les éléments biports, le sens positif du débit est défini par convention : le connecteur en amont par défaut est le connecteur 1 comme montré dans l'illustration de droite en Figure 2.5. On pose alors $q = q_1 = -q_2$.

$$0 = \begin{cases} h_2 - C_2.h, & \textbf{si} \quad q < -q_\epsilon \\ h_1 - \frac{1}{2}\left(C_1.h + C_2.h + (C_1.h - C_2.h)\sin(\frac{\pi}{2}\frac{q}{q_\epsilon})\right), & \textbf{si} \quad -q_\epsilon \leq q \leq q_\epsilon \\ h_1 - C_1.h, & \textbf{si} \quad q > q_\epsilon \end{cases}$$

La deuxième formulation que l'on va présenter est inspirée de celle retenue dans la libraire *Media* du standard Modelica [14], qui introduit un nouveau type de donnée dans les connecteurs fluides : *stream*. Une variable *stream* correspond à une grandeur qui transite par une variable de flux (ici le débit). C'est le cas par exemple de l'enthalpie ou de la composition massique d'un fluide. Les connecteurs fluides définis avec ce standard contiennent ainsi comme variable *stream* l'enthalpie h_i en amont du composant. L'enthalpie de mélange ne fait plus partie du connecteur lui-même. A la place, une fonction nommée *inStream* est utilisée afin de calculer pour chacun des composants i connectés ensemble la valeur qu'aurait h si le débit allait du nœud vers ce composant (soit $q_i > 0$) . C'est comme si l'on calculait n versions possibles de h. Ces n versions permettent comme on le verra par la suite de régulariser la transition lors d'une inversion de débit. La formulation *inStream* n'utilise plus des tests conditionnels comme précédemment mais une version régularisée de la fonction max. L'expression de *inStream* pour le connecteur i relié au noeud de mélange est la suivante :

$$inStream(h_i) = \frac{\sum\limits_{j=\overline{1..n}, j\neq i} \max(-q_j, 0) h_j}{\sum\limits_{j=\overline{1..n}, j\neq i} \max(-q_j, 0)} \tag{2.5}$$

Comme on peut le constater, cette formulation pose problème lorsque tous les débits sont nuls. Pour cette raison, une régularisation est aussi introduite lorsque *tous* les débits entrant dans le nœud de mélange sont proches de 0, c'est-à-dire $|\sigma_i| \leq q_\epsilon$, avec

$$\sigma_i = \sum_{j=\overline{1..n}, j\neq i} \max(-q_j, 0) \tag{2.6}$$

La régularisation employée pour $|\sigma_i| \leq q_\epsilon$ et que l'on retiendra par la suite diffère quelque peu de celle employée dans ThermoSysPro. Elle sera importante dans le chapitre 3 consacré à l'optimisation, où l'on requiert que les modèles soient différentiables.

Régularisation des inversions de débit :

On introduit la fonction différentiable $\text{posMax}_{\epsilon,w}(x)$ représentée dans la Figure 2.6. Le paramètre w définit la largeur de la zone de régularisation de la fonction et ϵ sa valeur

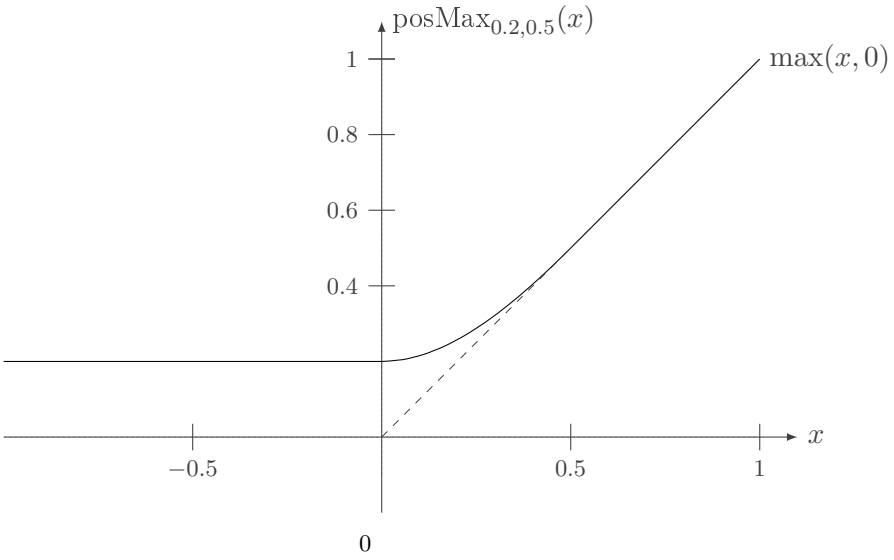


FIGURE 2.6 – Fonction posMax

minimale. Cette fonction vérifie $\text{posMax}_{\epsilon,w}(x<0)=\epsilon$, $\text{posMax}_{\epsilon,w}(x>\epsilon)=x$ et est définie par :

$$\text{posMax}_{\epsilon,w}(x)=\begin{cases}\epsilon, & \text{si } \tau<0\\ (2\epsilon-w)\tau^3+(2w-3\epsilon)\tau^2+\epsilon, & \text{si } 0\leq\tau\leq 1\\ x, & \text{si } \tau>1\end{cases}, \quad \text{avec } \tau=\frac{x}{w} \tag{2.7}$$

On pourra omettre les paramètres w et ϵ pour raccourcir la notation.

Inversion de débit avec la fonction posMax :

La fonction posMax peut être employée dans le schéma à grille décalée, par exemple utilisé lors de la discrétisation spatiale d'une EDP. Dans ce cas, l'enthalpie au sein de la maille i est notée $h_{vol,i}$:

$$H_{i-1}-H_i=\rho_i V_i\frac{dh_{vol,i}}{dt} \tag{2.8}$$

Avec H_i donné par :

$$H_i=q\begin{cases}h_{vol,i}, & \text{si } q\geq 0\\ h_{vol,i+1}, & \text{si } q_i<0\end{cases} \tag{2.9}$$

Un lissage de cette expression s'effectue en remplaçant les équations de (2.9) par

$$H_i=\text{posMax}(q)h_{vol,i-1}-\text{posMax}(-q)h_{vol,i+1}$$

Ce principe sera appliqué dans le cas d'étude 2.3.2.

2.2 Formalisme DAE pour la modélisation 0D/1D

On a présenté jusqu'alors des concepts de modélisation pour les centrales d'énergie. Dans cette partie, on détaillera le formalisme mathématique utilisé dans Modelica pour la modélisation de ces systèmes et le principe des algorithmes mis en œuvre pour les simuler. La

compréhension de ces étapes est nécessaire pour écrire convenablement les équations du système à modéliser, et comprendre les messages d'erreur lorsque l'initialisation ou la simulation du modèle échoue. Disposer d'un modèle robuste à l'initialisation et à la simulation sera notamment un point essentiel dans l'architecture de commande proposée.

2.2.1 Equations Algébro-Différentielles (DAE)

Un formalisme assez général pour modéliser les systèmes décrits précédemment est celui des systèmes d'équations algébro-différentielles ou DAE (*Differential Algebraic Equations*). Les DAE font intervenir des variables différentielles notées x (dont l'évolution est caractérisée par une équation différentielle), des variables algébriques notées z, des commandes u, des paramètres p, ainsi que le temps t. Plusieurs classes de DAE peuvent être retenues et sont présentées ci-dessous.

2.2.1.1 Équations différentielles ordinaires (ODE) - modèle d'état

Le formalisme DAE englobe comme cas le plus simple les systèmes d'équations différentielles ordinaires (ODE). Les variables algébriques z sont définies explicitement en fonction de x (équations de sortie).

$$\begin{aligned} \dot{x} &= f(x, u, p, t) \\ z &= \bar{g}(x, u, p, t) \\ x(0) &= x_0 \end{aligned} \tag{ODE}$$

2.2.1.2 DAE Semi-explicite

Une forme plus générale de DAE est la forme semi-explicite

$$\begin{aligned} \dot{x} &= f(x, z, u, p, t) \\ 0 &= g(x, z, u, p, t) \\ x(0) &= x_0 \end{aligned} \tag{DAEse}$$

Le théorème des fonctions implicites indique que z peut être calculé comme une fonction de x : $z = \bar{g}(x, u, p, t)$ si g est inversible, c'est-à-dire si le jacobien de g en z $\nabla_z g(x, z, u, p, t)$ n'est pas singulier. En effet, si l'on dérive la seconde équation, on obtient :

$$0 = \nabla_x g\dot{x} + \nabla_z g\dot{z} + \nabla_u g\dot{u} + \nabla_t g$$

Si $\nabla_z g$ est inversible on obtient l'équation différentielle suivante pour z (sur des intervalles où u est différentiable) :

$$\dot{z} = -(\nabla_z g)^{-1} \left(\nabla_x g\dot{x} + \nabla_u \bar{g}\dot{u} + \nabla_t g\right) \tag{2.10}$$

Il peut arriver que ce jacobien soit toujours singulier : on parlera alors de singularité structurelle, ou qu'il le devienne uniquement pour certaines valeurs des variables : on parlera alors de singularité locale.
Dans la pratique, z ne sera pas calculé par intégration de $\dot{z}$, mais par recherche du 0 de la fonction g. Cela se traduit par la résolution d'une ou plusieurs boucles algébriques, un point qui sera évoqué en partie 2.2.2.

2.2.1.3 Forme générale : DAE implicite

Sous sa forme la plus générale, un DAE peut se représenter sous forme implicite :

$$\begin{aligned} F(x, \dot{x}, z, u, p, t) &= 0 \\ x(0) &= x_0 \end{aligned} \tag{DAEi}$$

Si l'on note $X = \begin{pmatrix} \dot{x} \\ z \end{pmatrix}$ le vecteur composé des variables différentielles et algébriques, alors X peut être résolu si $\nabla_X F(x, X, u, p, t)$ est inversible.

2.2.2 Traitement symbolique des DAE

On a présenté pour l'instant la modélisation mathématique des systèmes étudiés. Cela ne présage en rien de la façon dont sont résolus ces systèmes d'équations. Cette tâche est remplie par les compilateurs du langage Modelica, qui sont capables de générer un code exécutable pour résoudre numériquement ces systèmes d'équations.

La simulation d'un système physique modélisé par un DAE consiste à assigner à tout instant t une valeur numérique vérifiant le jeu d'équations aux variables $x(t)$, $z(t)$ et $\dot{x}(t)$. La présence de variables différentielles implique de plus le couplage avec un algorithme d'intégration numérique. Deux problèmes doivent ainsi être résolus : un problème d'initialisation qui permet de trouver un jeu cohérent de variables différentielles et algébriques à t_0, puis un problème d'intégration numérique. Dans cette partie, on s'intéresse aux traitements symboliques effectués par les compilateurs Modelica pour résoudre les systèmes DAE ainsi qu'aux principaux algorithmes d'intégration numérique utilisés. Cette compréhension nous permettra de mieux appréhender les deux chapitres suivants : dans le chapitre 3 on fera usage d'une méthode d'intégration numérique connue sous le nom de méthode de collocation, et dans chapitre 4 l'algorithme de commande prédictive développé suppose que le modèle a été transformé symboliquement dans une forme adaptée.
Deux approches se dégagent pour la simulation des systèmes DAE. Une première est de transformer symboliquement le DAE en ODE (ODE) pouvant être alors résolue par un intégrateur ODE. La seconde approche résout directement le système implicite (DAEi) en utilisant des intégrateurs appropriés pour les DAE tels que DASSL [17], sur lequel on reviendra en partie 2.2.3.2.
On décrit ici la première méthode permettant d'aboutir à un système d'équations sous forme de modèle d'état. Cette méthode fait intervenir plusieurs algorithmes de traitement symbolique, très bien détaillés dans le livre [18] et dont on donne ici un bref aperçu. Les principales étapes permettant permettant d'aboutir à un code de simulation sont les suivantes :

- On détermine une causalité pour le calcul des variables $\dot{x}$ et z à l'aide d'un algorithme de manipulation de graphe présenté en 2.2.2.1 : l'algorithme de Tarjan [19].
- Si le système d'équation contient des boucles algébriques (c'est-à-dire des dépendances cycliques entre les variables), l'algorithme de Tarjan ne parviendra pas à définir une causalité pour les variables impliquées. Des méthodes pour traiter les boucles algébriques et débloquer l'algorithme de Tarjan sont présentées en 2.2.2.2 .
- Il peut aussi arriver que le système présente des singularités, c'est-à-dire des points où le système est indéterminé. Une singularité peut se produire pour certaines valeurs des

variables (division par 0 par exemple), ou provenir de couplages entre certaines variables différentielles dans le jeu d'équations, auquel cas on parlera de singularité structurelle. Un système présentant une singularité structurelle est dit d'index élevé et ne peut pas être simulé par un intégrateur ODE. On présente en 2.2.2.3 un algorithme de réduction d'index symbolique permettant de traiter ce cas particulier.
- Enfin, une fois la causalité établie, les variables x sont calculées par intégration des $\dot{x}$ grâce à un algorithme d'intégration numérique, et les étapes précédentes sont réitérées à chacun des pas d'intégration.

Remarque : Les modèles Modelica sont construits par assemblage de composants. Une première étape pour obtenir le DAE du système consiste à regrouper les équations de tous les composants ainsi que les équations générées par les connexions de ces composants entre eux (égalité des variables de potentiel et loi des nœuds pour les variables de flux). Cette étape est appelée *flattening*[14].

2.2.2.1 Causalisation des équations

Initialement, un DAE consiste en un ensemble acausal d'équations reliant les variables entre elles. Par acausal, on entend qu'il n'y a pas d'orientation dans le calcul des variables d'une équation au sens des entrées/sorties (et non pas que les phénomènes modélisés défient la causalité). On présente ici l'algorithme de Tarjan [19] qui permet d'assigner une valeur aux variables de façon séquentielle, établissant ainsi une causalité de calcul. Les variables d'états $x(t)$ sont supposées connues car fournies par la sortie de l'algorithme d'intégration. Il en va de même pour les entrées u qui sont données de façon externe. En revanche, les variables différentielles $\dot{x}$, tout comme les variables algébriques z sont les inconnues du problème de causalisation. On s'intéresse aux systèmes d'équations complets (carrés), c'est-à-dire possédant le même nombre d'équations et d'inconnues.
La structure de tels systèmes d'équations peut être transcrite de façon matricielle par la *matrice d'incidence* ou par un graphe nommé *digraphe*. Ces deux outils équivalents définissent quelles variables interviennent dans une équation et dans quelles équations intervient une variable.

Définition (Digraphe) Le digraphe structurel est un graphe orienté dans lequel les arcs relient les équations aux variables qu'elles font intervenir.

Définition (Matrice d'incidence) Dans une matrice d'incidence $M \in \mathcal{M}_N$, chaque ligne représente une équation, chaque colonne une variable. $m_{ij} = 1$ si la variable j apparaît dans l'équation i, 0 sinon.

Afin d'illustrer les méthodes présentées, des exemples vont être présentés dont certains inspirés par [20] et [18].
L'algorithme de Tarjan a pour but d'établir l'ordre dans lequel les variables sont calculées. Les équations permettant de déterminer ces variables sont numérotées par $n \in \overline{1..N}$. Une interprétation visuelle est souvent donnée pour cet algorithme, et procède en colorant les arcs du digraphe comme détaillé dans l'Algorithme 1.

Algorithme 1 Algorithme de Tarjan

Entrées: Digraphe structurel de N équations et N inconnues. Tous les arcs sont noirs.
$n = 1$
Répéter
Tant que Il existe au moins une équation dont est issu un seul arc noir **Faire**
Choisir une de ces équations. La variable associée est $\tilde{x}_n$.
Colorer en rouge cet arc et numéroter l'équation n.
Colorer en bleu les arcs noirs issus de $\tilde{x}_n$.
$n \leftarrow n + 1$
Fin Tant que
Tant que Il existe au moins une variable dont est issu un seul arc noir **Faire**
Choisir une de ces variables. L'équation associée est $\tilde{e}_{N-n}$.
Colorer en rouge cet arc et numéroter l'équation $N - n$.
Colorer en bleu les arcs noirs issus de $\tilde{e}_{N-n}$.
Fin Tant que
Jusqu'à Plus aucun arc ne peut être coloré selon les règles précédentes
Retourner Système d'équations triées $\tilde{e}_n, \quad n \in \overline{1..N}$

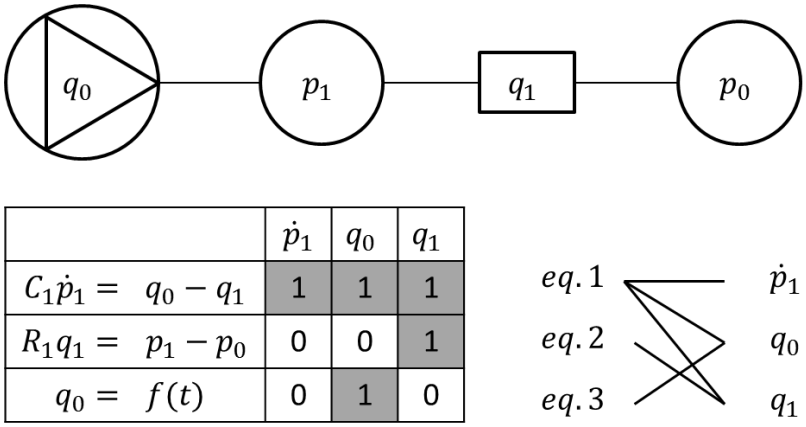


FIGURE 2.7 – Modèle hydraulique 1 : schéma, matrice d'incidence et digraphe

On illustre le fonctionnement de cet algorithme sur le réseau hydraulique très simple présenté en Figure 2.7. On donne aussi la matrice d'incidence et le digraphe du système d'équations de ce système. Le réseau est modélisé de façon capacitive : les pertes de charges sont équivalentes à des résistances R et les volumes à des capacités C. Le réseau est constitué d'une pompe donnant un débit q_0, et de deux volumes reliés par une perte de charge. Le dernier volume est à pression constante. On utilise la notation p pour les pressions et q pour les débits. L'application de cet algorithme au problème précédent est représenté dans la Figure 2.8. La première boucle *Tant que* de l'algorithme de Tarjan est ici suffisante pour rendre les équations causales.

On constate dans la Figure 2.9 que la matrice d'incidence du système trié par l'algorithme de Tarjan est triangulaire inférieure.

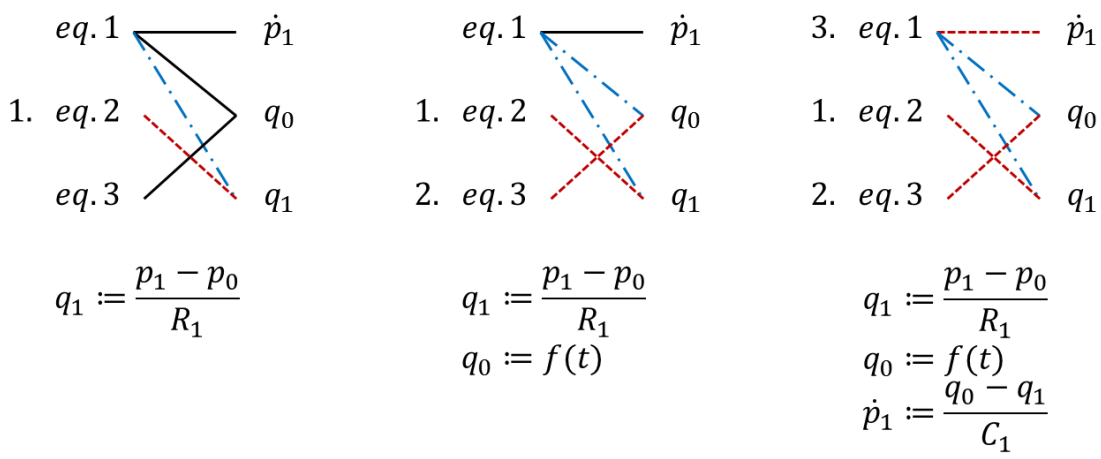


FIGURE 2.8 – Causalisation des équations du modèle hydraulique 1

	$\dot{p}_1$	q_0	q_1
$eq.1$	1	1	1
$eq.2$	0	0	1
$eq.3$	0	1	0

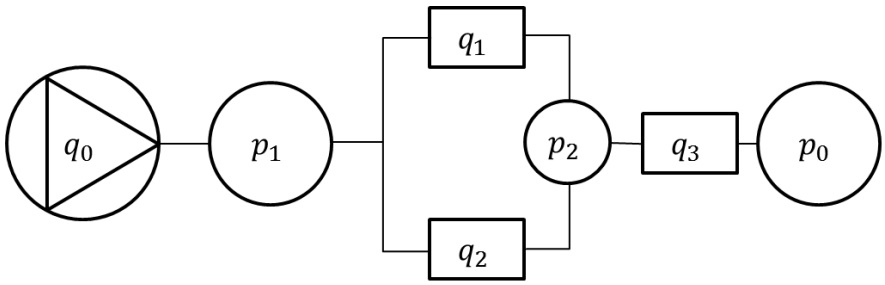
	q_1	q_0	$\dot{p}_1$
$eq.2$	1	0	0
$eq.3$	0	1	0
$eq.1$	1	1	1

FIGURE 2.9 – Matrice d'incidence du modèle hydraulique 1 avant et après causalisation

Un système pourra être complètement causalisé si sa matrice d'incidence est triangulaire inférieure après réorganisation des lignes et des colonnes par l'algorithme de Tarjan. Il peut arriver que l'algorithme de Tarjan aboutisse à un forme Bloc Triangulaire Inférieure (BLT) de la matrice d'incidence (sur le digraphe, les équations restantes font intervenir au moins deux variables et chacune des variables restantes apparaît dans au moins deux équations). Une telle situation est le signe de la présence de boucles algébriques dans le système. Le traitement des ces boucles algébriques est décrit dans le paragraphe suivant.

2.2.2.2 Traitement des boucles algébriques

On considère maintenant le réseau hydraulique représenté en Figure 2.10. Dans ce modèle, la pression p_2 est considérée comme une variable algébrique et non comme une variable différentielle, ce qui est possible si on néglige cette dynamique rapide. L'algorithme de Tarjan est à nouveau appliqué à ce système comme montré en Figure 2.11. On constate que l'algorithme ne parvient pas à assigner de causalité à quatre des équations pour calculer p_2, p_1,q_2 et q_3. La matrice d'incidence est ici bloc-triangulaire inférieure (BLT), la zone encadrée en rouge correspond à la boucle algébrique. Une boucle algébrique impliquant n_a variables peut se mettre sous forme d'une équation implicite de la forme $H(X) = 0$ avec $H : \mathbb{R}^{n_a} \to \mathbb{R}^{n_a}$. Pour



	$\dot{p}_1$	p_2	q_0	q_1	q_2	q_3
$eq.1$	1	0	1	1	1	0
$eq.2$	0	0	0	1	1	1
$eq.3$	0	1	0	1	0	0
$eq.4$	0	1	0	0	1	0
$eq.5$	0	0	1	0	0	0
$eq.6$	0	1	0	0	0	1

$$
\begin{aligned}
C_1\dot{p}_1 &= q_0 - q_1 - q_2 \\
q_3 &= q_1 + q_2 \\
R_1 q_1 &= p_1 - p_2 \\
R_2 q_2 &= p_1 - p_2 \\
q_0 &= f(t) \\
R_3 q_3 &= p_2 - p_0
\end{aligned}
$$

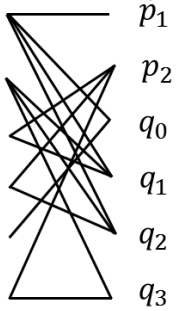


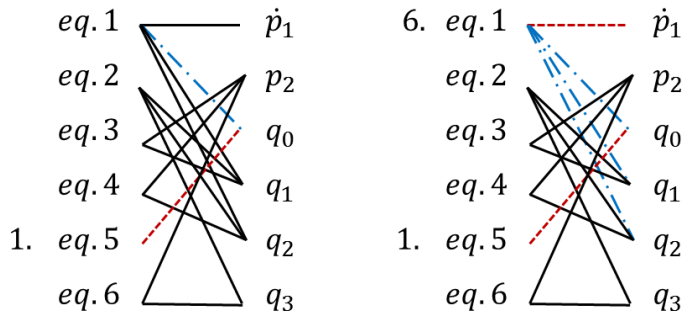
FIGURE 2.10 – Modèle hydraulique 2 : schéma, matrice d'incidence et digraphe

l'exemple présenté, on peut ainsi écrire

$$
0 = H(p_2, q_3, q_1, q_2) = \begin{pmatrix} q_3 - q_1 - q_2 \\ p_2 - p_1 + R_1 q_1 \\ p_2 - p_1 + R_2 q_2 \\ p_0 - p_2 + R_3 q_3 \end{pmatrix}
$$

Ici les équations impliquées sont linéaires, et il est possible de résoudre le système de façon matricielle. Dans le cas non linéaire, la résolution de la boucle algébrique consiste à rechercher simultanément la valeur des inconnues X solution de $H(X) = 0$. Cette tâche peut être effectuée par l'algorithme de Newton que l'on présente en Annexe B.

Remarque L'algorithme de Newton est un algorithme de recherche locale qui nécessite la plupart du temps d'être initialisée avec des valeurs des variables proches de la solution. Trouver un vecteur cohérent pour initialiser un algorithme de recherche locale peut être problématique pour les systèmes thermohydrauliques. Une des raisons est l'appel à des tables de propriétés définies par des équations définies par zone. Ainsi, la mauvaise équation peut être utilisée si une variable inconnue est mal initialisée. Par ailleurs, des ordres de grandeurs très différents pour les variables et leurs dérivées peuvent coexister dans les systèmes physiques. La normalisation de ces grandeurs est indispensable pour obtenir un bon conditionnement numérique du jacobien qui intervient dans l'algorithme de Newton.



	q_0	p_2	q_3	q_1	q_2	$\dot{p}_1$
$eq.5$	1	0	0	0	0	0
$eq.2$	0	0	1	1	1	0
$eq.3$	0	1	0	1	0	0
$eq.4$	0	1	0	0	1	0
$eq.6$	0	1	1	0	0	0
$eq.1$	1	0	0	1	1	1

FIGURE 2.11 – Modèle hydraulique 2 : blocage de l'algorithme de Tarjan

Algorithme de séparation (*Tearing*)

Il est possible de réduire la taille d'une boucle algébrique de façon symbolique. Dans l'exemple précédent, si l'on supposait par exemple que la variable q_2 était connue (arc vert) par l'équation ($eq.4$) (arc violet), alors la causalité pourrait être établie pour le restant des variables selon l'algorithme de Tarjan comme montré dans la Figure 2.12.

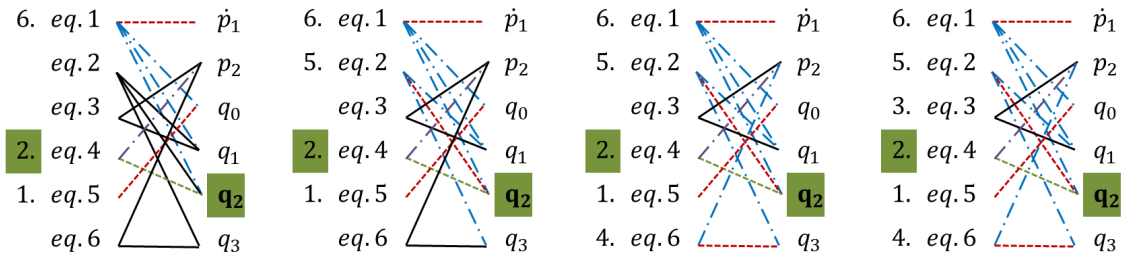


FIGURE 2.12 – Modèle hydraulique 2 : hypothèse q_2 connue, ($eq.4$) utilisée comme équation des résidus, déblocage de l'algorithme de Tarjan

A la fin de cet algorithme (4^{me} schéma), on constate que l'équation ($eq.2$) permet à son tour de déterminer q_2, la variable au niveau de laquelle on avait "ouvert" la boucle. Le système sera alors résolu si cette équation, que l'on peut réécrire sous la forme $0 = q_3 - q_1 - q_2$ est vérifiée. Or comme ce n'est pas nécessairement le cas (q_2 peut être initialisé de façon arbitraire), on introduit une nouvelle variable $r := q_3 - q_1 - q_2$ nommée résidu. Le système causalisé devient :

$$q_0 := f(t) \qquad \text{(eq. 5)}$$
$$p_2 := p_1 - R_2\mathbf{q_2} \qquad \text{(eq. 4)}$$
$$q_1 := \frac{p_1 - p_2}{R_1} \qquad \text{(eq. 3)}$$
$$q_3 := \frac{p_2 - p_0}{R_3} \qquad \text{(eq. 6)}$$
$$\mathbf{r} := q_3 - q_1 - q_2 \qquad \text{(eq. 2)}$$
$$\dot{P}_1 := \frac{q_0 - q_1 - q_2}{C_1} \qquad \text{(eq. 1)}$$

Le système sera alors résolu lorsque l'on aura trouvé q_2 tel que :

$$0 = r(q_2) = r(q_3(q_2, q_1(p_2(q_2), q_2), p_2(q_2)), q_1(q_2, p_2(q_2)), q_2)$$

Ce qui peut se faire comme précédemment grâce à la méthode de Newton. Cependant, comme on peut le constater, l'itération ne se fait plus sur la fonction $H : \mathbb{R}^4 \to \mathbb{R}^4$ mais sur la fonction $r : \mathbb{R} \to \mathbb{R}$. On est ainsi passé d'un problème de 4 à 1 variable. On formalise maintenant cette démarche connue sous le nom d'algorithme de séparation, ou *tearing*.

Définition (Variables de séparation). Soit $X^a = \{x_1^t, \ldots, x_{n_t}^t, x_1^s, \ldots, x_{n_s}^s\}$ un ensemble de $n_a = n_t + n_s$ variables impliquées dans une boucle algébrique de la forme $H(X^a) = 0$ avec $H : \mathbb{R}^{n_a} \to \mathbb{R}^{n_a}$. Les variables x_i^t du sous ensemble $X^t = \{x_1^t, \ldots, x_{n_t}^t\}$ de X^a sont appelées variables de séparation (*Tearing variables*) si il existe une fonction $h : \mathbb{R}^{n_t} \to \mathbb{R}^{n_s}$ vérifiant $X^s = h(X^t)$ [21].

La résolution des variables d'une boucle algébrique peut alors s'effectuer ainsi de façon séquentielle :

$$\begin{aligned} x_1^r &:= h_1(X^t) \\ x_2^r &:= h_2(X^t, x_1^r) \\ x_3^r &:= h_3(X^t, x_1^r, x_2^r) \\ &\vdots \\ x_r^r &:= h_r(X^t, x_1^r, \ldots, x_{r-1}) \end{aligned}$$

Remarque La fonction $h(X^t)$ citée ci-dessus peut se retrouver en employant des substitutions successives

$$X^a = h(X^t) = \begin{pmatrix} X^t \\ h_1(X^t) \\ h_2(X^t, h_1(X^t)) \\ \vdots \\ h_r(X^t, h_1(X^t), \ldots, h_{r-1}(X^t \ldots, h_{r-1}(X^t, \ldots, h_{r-2}(\ldots)))) \end{pmatrix}$$

Les variables de séparation permettent ainsi d'ouvrir la boucle de causalité. On introduit maintenant la fonction des résidus qui permet de la clore.

Définition (Fonction des résidus) La fonction des résidus associée à la résolution de l'équation $H(X^a) = 0$ est une fonction $r : \mathbb{R}^{n_t} \to \mathbb{R}^{n_r}$ telle que $r(X^t) = 0 \Rightarrow H(X^a) = 0$.

La fonction des résidus se construit d'après le digraphe :

- On identifie les équations permettant de définir les variables de séparation, qui sont de la forme $x_i^t = f_j(X^a)$
- On exprime ces équations en fonction de X^t à l'aide de la relation $X^a = h(X^t)$: $x_i^t = f_j \circ h(X^t)$
- On écrit le résidu i comme $r_i = x_i^t - f_j \circ h(X^t)$

L'intérêt de résoudre $r(x^t) = 0$ plutôt que $H(x) = 0$ vient de la réduction de la dimension du problème. Cependant, le choix des variables de séparation et des équations des résidus n'est pas unique. Par exemple dans l'exemple précédent, les jeux de variables de séparation q_2,p_2,$\{q_2, p_2\}$,q_1,etc permettent tous d'établir la causalité. Certaines heuristiques permettent de choisir un petit ensemble de variables de séparation :

- Déterminer les équations d'où partent le plus grand nombre d'arcs noirs (les "plus indéterminées")
- Pour chacune des variables impliquées (au bout de l'arc), déterminer celles qui sont impliquées dans le plus d'équations (d'où part le plus grand nombre d'arcs). Noter combien d'équations peuvent être rendues causales si chacune de ces variables est supposée connue.
- Utiliser comme variable de séparation les variables qui permettent de rendre le plus d'équations causales.

2.2.2.3 La notion d'index pour les DAE

On parle de singularité lorsque la valeur d'une variable ne peut pas être calculée d'après le jeu d'équations fourni. Dans le cadre des systèmes modélisés par (DAEse) par exemple, on a vu qu'une condition nécessaire pour résoudre z était que le gradient ∇g_z soit inversible (donc de rang plein). Il peut arriver que ∇g_z ne soit pas inversible pour une valeur particulière de x, mais aussi qu'il ne le soit jamais : on parle alors de singularité structurelle.

Définition (Index d'un DAE) [18] Un DAE sans boucle algébrique est dit d'index 0, un DAE contenant des boucles algébriques mais aucune singularité structurelle est d'index 1. Pour les autres, on parlera de DAE d'index élevé. Les DAE sous forme (ODE) sont ainsi d'index 0 et les DAE sous forme (DAEse) avec g inversible sont d'index 1.

Les singularités structurelles sont dues à l'existence de couplage entre certaines variables différentielles [20]. Ces couplages peuvent être dus au choix des variables d'état ou à la présence de dynamiques négligées. La conséquence est que les variables $\dot{x}_i$ ne sont en fait pas toutes des inconnues et ne peuvent plus être intégrées séparément : le système est sur-contraint. Dans ce cas, l'algorithme de Tarjan aboutira sur des équations de contraintes, c'est-à-dire des équations qui couplent certaines variables sans pour autant permettre de leur attribuer une valeur. Par exemple si l'équation $x_1 = x_2$ est présente dans un système, il n'est évidemment pas possible d'intégrer séparément $x_1 = \int \dot{x}_1 dt$ et $x_2 = \int \dot{x}_2 dt$ pour n'importe quelle condition initiale. Une première technique pour identifier de nouvelles relations entre les variables consiste alors à différentier les équations de contraintes : ainsi, si à tout instant l'équation $x_1(t) = x_2(t)$ est vérifiée, l'équation $\dot{x}_1(t) = \dot{x}_2(t)$ l'est aussi : on constate que seule une des variables (par exemple x_1) est réellement une variable d'état, l'autre (x_2) est en fait une variable algébrique z qui doit être ajoutée aux inconnues. Puis afin d'éviter toute confusion, comme $\dot{x}_2$ n'est plus envoyée à un intégrateur, on la renomme $der x_2$ (on parle de *dummy derivative*). Ce principe est à la base des procédures symboliques de réduction d'index dont la plus connue est l'algorithme (2) de Pantelides, qui est implémenté dans le logiciel Dymola. Des explications plus poussées sont données dans [18] et [22].

Dans cette thèse, on fera par la suite l'hypothèse que les systèmes sont sous formes de DAE d'index au plus 1. L'algorithme de Pantelides pourra être utilisé pour s'y ramener lorsque ce sera nécessaire.

Algorithme 2 Algorithme de Pantelides

Entrées: Système d'équation traité par l'algorithme de Tarjan et de Séparation si nécessaire

Tant que Une équation de contrainte apparaît (équation avec des arcs bleus uniquement) **Faire**
Différentier l'équation de contraintes
Ajouter cette équation au système
Transformer une des nouvelles variables $\dot{x}_i$ en variable algébrique $derx_i$, et ajouter x_i comme nouvelle inconnue (équilibre du nombre d'équations et d'inconnues)
Si la différentiation a fait apparaître de nouvelles variables $\dot{x}_j$, différentier toutes les équations où apparaissent x_j
Fin Tant que

En pratique, même si l'on a rendu le système régulier à l'aide de l'algorithme de Pantelides, il se peut que des singularités apparaissent pour certaines valeurs. Certains algorithmes [23] peuvent alors changer de jeu de variables d'état en cours de simulation (*Pivoting dummy derivatives*).

2.2.2.4 Différentiation automatique

La différentiation automatique (*Automatic Differentiation*), abrégée AD, est une technique algorithmique permettant de calculer des informations sur les gradients des variables en même temps que les valeurs de celles-ci. Cette méthode s'appuie sur une application systématique de la règle de dérivation en chaîne pour les fonctions composées. La causalisation des équations dans Modelica, qui assigne une valeur aux variables v pour un jeu de variables indépendantes u peut être écrite comme les n_e étapes de calcul suivantes :

$$\begin{aligned} v_1 &:= f_1(u) \\ v_2 &:= f_2(v_1, u) \\ &\vdots \\ v_{n-1} &:= f_{n-1}(v_1, v_2, \ldots, v_{n_e-1}) \\ y &:= F(v_1, v_2, \ldots, v_{n_e-1}) \end{aligned}$$

(les fonction f_i et F utilisées ici n'ont rien à voir avec celles utilisées pour les équations de la dynamique). La causalisation définit en fait les f_i comme les fonctions composées suivantes :

$$v_k(u) = f_k(f_1(u), f_2(f_1(u), u), \ldots, \ldots, f_{k-1}(f_{k-2}(\ldots), \ldots, u)), \quad \forall k \in \overline{1..n_e}$$

L'application de la règle de dérivation en chaîne donnerait ici

$$\frac{dv_k}{du} = \frac{\partial f_1}{\partial u} + \frac{\partial f_2}{\partial f_1}\frac{\partial f_1}{\partial u} + \ldots$$

Le calcul simultané de $y = F(v)$, avec $F : \mathbb{R}^{n_v} \to \mathbb{R}^{n_e}$ et de son jacobien $J = \frac{dF}{dv}$ en un point $\bar{v}$ peut ainsi s'effectuer selon l'Algorithme 3 [24] :

Des variantes de cet algorithme existent pour ne calculer les dérivées que dans certaines directions, par exemple $\frac{\partial F}{\partial v_3}$, ou pour certaines équations, par exemple $\frac{\partial f_1}{\partial v}$. Dans le premier

Algorithme 3 Calcul de $y = F(\bar{v})$ et $J = \frac{dF}{dv}\big|_{\bar{v}}$

$v_0 \leftarrow \bar{v}$
$\frac{dv_0}{dx} \leftarrow Id$
Pour $k = 1$ **à** n **Faire**
  $v_k \leftarrow \left(\{v_i\}_{i \in \overline{1..k-1}}\right)$
  $d\frac{dv_k}{dx} \leftarrow \sum_{i \in \overline{1..k-1}} \frac{\partial f_k}{\partial v_i} \frac{dv_i}{dx}$
Fin Pour
$y \leftarrow v_n$
$J \leftarrow \frac{dv_n}{dx}$
Retourner y, J

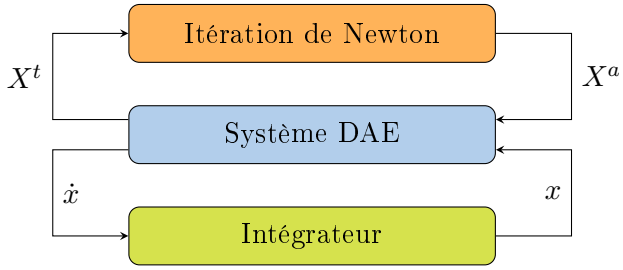


FIGURE 2.13 – Éléments nécessaires à la simulation d'un DAE

cas, on parle de mode avant (*forward mode of AD*), dans le second de mode arrière (*reverse mode of AD*). Ces deux variantes permettent de limiter la mémoire allouée au programme s'il n'est pas nécessaire de calculer intégralement le jacobien.
Dans le cadre de cette thèse, la différentiation automatique, disponible dans les logiciels Dymola et JModelica, sera utilisée pour obtenir les matrices de linéarisation du système le long de certaines trajectoires, et servira d'élément majeur pour l'algorithme de commande prédictive présenté au Chapitre 4.

2.2.3 Simulation des systèmes DAE

2.2.3.1 Algorithmes d'intégration pour les ODE

Les algorithmes de traitement symbolique décrits précédemment permettent d'aboutir à un code de calcul capable de déterminer en chaque instant t les valeurs des variables $\dot{x}(t)$ et $z(t)$ en fonction de $x(t)$ et $u(t)$. Ce code de calcul inclut éventuellement des instructions pour résoudre les boucles algébriques si celles-ci sont présentes dans le modèle. Cependant, afin de simuler le système, il convient de prendre en compte le fait que $\dot{x}(t) = \frac{dx(t)}{dt}$. Cette tâche est remplie par un algorithme d'intégration numérique, comme illustré en Figure 2.13. On présente ici les méthodes de Runge Kutta, dont l'exemple le plus simple est le schéma d'Euler avant.

Un algorithme d'intégration numérique a pour but de calculer la trajectoire des variables différentielles x aux instants t_k d'une grille temporelle. On considère pour l'instant un pas de temps T fixe sur toute la grille, c'est-à-dire $t_{k+1} - t_k = T$. La $i^{\text{ème}}$ coordonnée de x à l'instant

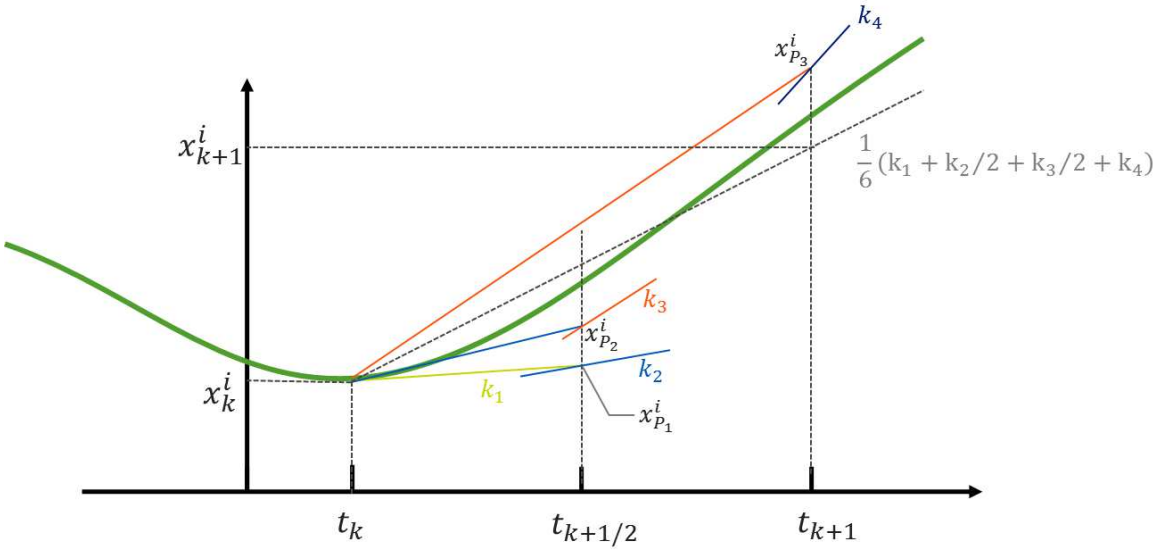


FIGURE 2.14 – Illustration de l'algorithme d'intégration numérique RK4

t_k est notée x^i_k. On commence par rappeler que la valeur de x^i_{k+1} peut être déterminée d'après l'approximation de Taylor autour du point x^i_k par :

$$x^i_{k+1} = x^i_k + T\dot{x}^i(t_k) + \frac{T^2}{2}\ddot{x}^i(t_k) + \dots$$

Les algorithmes d'intégration à un pas utilisent plus ou moins directement ce résultat pour déterminer récursivement x_{k+1} d'après x_k. La méthode d'intégration la plus simple est ainsi la méthode d'Euler avant (FE), qui tronque cette série au premier ordre :

$$\begin{aligned} x^i_{k+1} &= x^i_k + T \left.\frac{dx^i}{dt}\right|_{t_k} \\ &= x^i_k + Tf(x_k, u(t_k), z(t_k), t_k) \end{aligned} \tag{2.12}$$

Pour abréger la notation, on pose maintenant $\bar{f}(x_k, t_k) = f(x_k, u(t_k), z(t_k), t_k)$. Dans les algorithmes de Runge Kutta d'ordre plus élevé, plutôt que d'évaluer toutes les dérivées $\frac{dx^i}{dt}$, $\frac{d^2x^i}{dt^2}, \dots$ au point (x_k, t_k), l'idée est de calculer $\frac{dx^i}{dt}$ en plusieurs points de l'intervalle $[t_k, t_{k+1}]$. La méthode de Runge Kutta la plus connue est celle d'ordre 4 (RK4), qui utilise 3 points intermédiaires de coordonnées $(x_{P_1}, t_{k+1/2})$, $(x_{P_2}, t_{k+1/2})$ et (x_{P_3}, t_{k+1}), comme illustrée dans la Figure 2.14.

$$\begin{aligned} k_1 &= \bar{f}(x^i_k, t_k) && \text{Étage 0} \\ k_2 &= \bar{f}(x^i_k + T/2k_1, t_k + T/2) && \text{Étage 1} \\ k_3 &= \bar{f}(x^i_k + T/2k_2, t_k + T/2) && \text{Étage 2} \\ k_4 &= \bar{f}(x^i_k + Tk_3, t_k + T) && \text{Étage 3} \\ x^i_{k+1} &= x^i_k + \frac{T}{6}(k_1 + 2k_2 + 2k_3 + k_4) && \text{Étage 4} \end{aligned}$$

Les coefficients utilisés ainsi que les endroits d'évaluation des dérivées ont été choisis pour garantir que l'approximation est exacte jusqu'à l'ordre 4, c'est-à-dire qu'il n'y a pas d'erreur sur le développement de Taylor pour les termes jusqu'à T^4 [18]. Les méthodes FE et RK4 sont dites explicites car il est possible de calculer séquentiellement les différents termes nécessaires au calcul de x_k^i. A l'inverse, certaines méthodes sont dites implicites, l'exemple le plus simple étant la méthode d'Euler arrière (BE) :

$$x_{k+1}^i = x_k^i + T\bar{f}(x_{k+1}, t_k + T) \tag{2.13}$$

La solution de cette équation nécessite ici la résolution d'une boucle algébrique pour la variable x_{k+1}, c'est pour cela que la méthode est qualifiée d'implicite. De la même façon, il existe des méthodes de RK implicites. Les méthodes implicites sont en général plus stables mais plus gourmandes en calculs (en raison de la présence d'une boucle algébrique) [18].
On introduit enfin dans les grandes lignes une dernière méthode d'intégration numérique qui sera utilisée et présentée plus en détails dans le chapitre suivant : la méthode de collocation. Le principe de la méthode de collocation est de décrire la trajectoire d'état $x^i(t)$ comme un polynôme $\tilde{x}^i(t)$ de degré n_c, qui respecte l'équation de la dynamique en un nombre n_c de points distincts de l'intervalle $[t_k, t_{k+1}]$. Ces points sont donnés par leur fraction τ_c de l'intervalle (on a ainsi $0 \leq \tau_c \leq 1, \quad \forall c \in \overline{1..n_c}$)

$$\begin{aligned} \tilde{x}^i(t_k) &= x_k^i \\ \dot{\tilde{x}}^i(t_k + \tau_c T) &= \bar{f}(\tilde{x}^i(t_k + \tau_c T), t_k + \tau_c T), \quad c \in \overline{1..n_c} \end{aligned} \tag{2.14}$$

Il est possible de montrer que la méthode de collocation est en fait équivalente à une méthode de RK implicite [25].

2.2.3.2 Algorithmes d'intégration pour les DAE

Les algorithmes précédents supposaient que le système puisse être mis sous forme de modèle d'état (ODE) ou de DAE semi-explicite (DAEse) afin de pouvoir découpler le calcul des variables différentielles et algébriques. Cette approche a été présentée en partie 2.2.2. Cependant, il existe des algorithmes d'intégration pouvant traiter directement les systèmes sous forme de DAE implicite (DAEi), sans avoir à convertir le système. Le plus connu d'entre eux est DASSL [17] qui est l'algorithme par défaut de Dymola. En posant $X = [x, z]$, un système implicite peut s'exprimer sous la forme générale suivante :

$$\begin{aligned} F(\dot{X}, X, t) &= 0 \\ X(0) &= X_0 \\ \dot{X}(0) &= dX_0 \end{aligned} \tag{2.15}$$

L'idée majeure de DASSL est de remplacer $\dot{X}$ par son approximation par une méthode d'intégration. Dans le cas de la méthode d'Euler arrière (BE) cela donne :

$$0 = F\left(\dot{X}_{k+1}, X_{k+1}, t_{k+1}\right) \approx F\left(\frac{x_{k+1} - x_k}{T}, X_{k+1}, t_{k+1}\right)$$

Il s'agit d'une équation implicite qui peut être résolue par la méthode de Newton en une seule étape (alors que sinon la résolution de deux boucles algébriques est nécessaire : une pour

trouver le 0 de la fonction F, la seconde pour résoudre le pas de la méthode BE qui est une méthode implicite). Une itération m de cette boucle se formule comme :

$$\begin{aligned} X_{k+1}^{m+1} &= X_{k+1}^{m} - \nabla F^{-1} F\left(\frac{X_{k+1} - X_k}{T}, X_{k+1}, t_{k+1}\right) \\ &= X_{k+1}^{m} - \left(\frac{\partial F}{\partial \dot{X}} + \frac{1}{T}\frac{\partial F}{\partial X}\right)^{-1} F\left(\frac{X_{k+1}^{m} - X_k}{T}, X_{k+1}^{m}, t_{k+1}\right) \end{aligned}$$

2.2.3.3 Algorithmes d'intégration à pas variable, à ordre variable et contrôle de l'erreur

Jusqu'alors, on a considéré un pas d'intégration T fixe. En réalité, la plupart des algorithmes utilisés (dont DASSL) sont des algorithmes à pas variable T_k. L'utilisation d'un pas variable se justifie car il peut exister certains intervalles de temps où le système évolue rapidement (raideurs dans le système autour du point courant, variations importantes des entrées), et certains intervalles où le système est proche d'un état d'équilibre. Dans le premier cas, un pas fin peut être nécessaire pour conserver une précision suffisante alors que dans le second l'utilisation de pas long permettrait de limiter le nombre de pas et donc d'améliorer la précision. Cette stratégie nécessite néanmoins une mesure de l'erreur afin de déterminer quel pas de temps T_k choisir sur un élément k. En d'autres termes, l'adaptation du pas permet de contrôler l'erreur d'intégration. Dans certaines méthodes, dites à ordre variable, le contrôle de l'erreur se fait en alternant entre des méthodes d'intégration d'ordre faible (rapides mais moins précises) et des méthodes d'ordre élevé (souvent des méthodes implicites nécessitant la résolution de boucles algébriques). Le contrôle de l'erreur d'intégration est une problématique qui interviendra plus tard dans le chapitre 3 et dans un souci de concision nous ne détaillerons pas ici les méthodes mises en œuvre.

2.2.4 Modélisation causale et acausale

Ce paragraphe a pour vocation d'éclaircir les avantages et inconvénients de l'utilisation de Modelica pour la commande, vis à vis de la modélisation sous forme de schémas-blocs traditionnellement utilisée dans le domaine de l'automatique. Dans une modélisation par schémas-blocs la causalité de calcul est déterminée au moment de la conception par le modélisateur : les différents blocs communiquent entre eux par des signaux orientés *a priori*. Ce formalisme est notamment celui retenu dans les logiciels MATLAB Simulink ou Scicos. Cette définition procédurale permet de maîtriser l'ordre de calcul des variables et de localiser aisément une source d'erreur dans le modèle, mais elle présente certains inconvénients :

- Le modélisateur doit être capable de définir la causalité de calcul à l'échelle du système. Bien que des méthodes d'analyse structurelle telles que les graphes de liaison (*bond graphs*) puissent l'aider dans cette tâche [26], toute modification structurelle du système (insertion ou retrait d'un composant) peut invalider la causalité établie auparavant, surtout en présence de boucles algébriques. L'adaptabilité des modèles est alors limitée.
- En présence de boucles algébriques, il incombe au modélisateur de définir quelles seront les variables de séparation ainsi que la procédure numérique permettant de résoudre la boucle algébrique. Cette tâche peut devenir très complexe lorsque les boucles algébriques sont de grande taille.

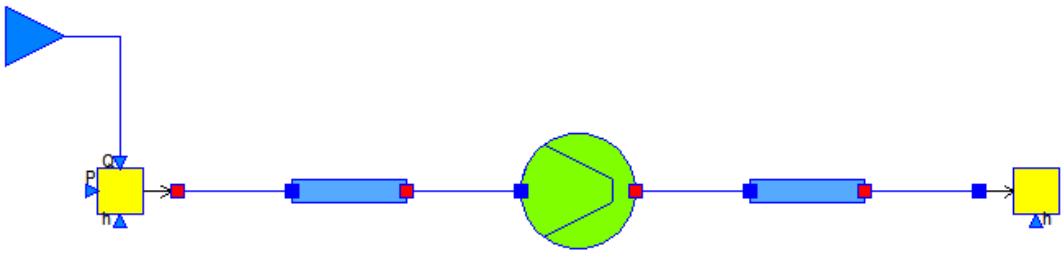


FIGURE 2.15 – Modèle ThermoSysPro d'un compresseur

– Un modèle en schéma-bloc ne peut être utilisé que dans l'optique pour laquelle il a été créé et ne permet pas d'effectuer des calculs nécessitant une inversion de la causalité (comme l'identification d'un paramètre).

A l'inverse, les traitements symboliques réalisés au moment de la compilation d'un modèle Modelica permettent d'établir automatiquement une causalité de calcul dans le modèle. La description équationnelle permet une définition plus modulaire des composants, qui peuvent alors être utilisés dans plusieurs contextes. Il devient alors possible de réutiliser un même modèle pour plusieurs besoins différents, en fixant certaines variables et en en relaxant d'autres. Par ailleurs, la résolution des boucles algébriques est automatisée avec des heuristiques permettant d'en accroître l'efficacité (algorithme de séparation). La contrepartie de cette plus grande abstraction dans la modélisation est que l'utilisateur n'a plus autant de contrôle sur la façon dont sont calculées les variables, ce qui peut compliquer la recherche d'une source d'erreur. Le modélisateur doit par ailleurs définir correctement les conditions limites pour que le modèle possède un nombre égal d'équations et d'inconnues. Les messages d'erreurs affichés par les compilateurs quand ce n'est pas le cas sont souvent peu informatifs même si des progrès ont été faits dans ce sens. Enfin, la résolution automatisée des boucles algébriques peut avoir un effet collatéral : le modélisateur peut avoir tendance à n'y prendre plus garde et décrire les équations du modèle d'une façon qui génèrera des boucles algébriques impliquant un grand nombre de variables. Cela s'avère problématique car l'algorithme de résolution utilisé (méthode de Newton) peut avoir un domaine de convergence restreint et il est alors nécessaire de fournir une bonne initialisation simultanée pour toutes ces variables, chose qui n'est pas évidente. En conclusion, Modelica est un langage de modélisation offrant au modélisateur des algorithmes de traitement symbolique performants pour la simulation de systèmes complexes, en lui ôtant la difficulté de déterminer la causalité de calcul dans le modèle. Cependant, la compréhension de ces algorithmes reste utile pour interpréter et corriger les messages d'erreur fournis par le compilateur.

2.3 Cas d'étude

2.3.1 Compresseur

On présente en Figure 2.15 un circuit de vapeur réalisé à l'aide de la librairie ThermoSysPro. Cette partie a pour but de montrer sur un exemple simple certains des points développés dans ce chapitre : la modélisation et simulation d'un système thermohydraulique avec Modelica d'une part, l'application des algorithmes de traitement symbolique d'autre part.

Il contient une source de vapeur à pression et enthalpie constante, et qui permet de contrôler le débit massique. La source est reliée à un compresseur par une conduite, lui même relié à un puits. Les propriétés de la vapeur sont calculées d'après le standard IF97. Le compresseur est un élément biport (tout comme les conduites). Il possède un connecteur d'entrée C_e et un connecteur de sortie C_s contenant les variables pressions P, enthalpie spécifique h et débit q. Les paramètres du compresseur sont son facteur de compressions $\pi = c_s.P/C_e.P$ et son efficacité isentropique η_{is} :

$$\eta_{is} = \alpha_m \frac{h_{is} - C_e.h}{C_s.h - C_e.h} \tag{2.16}$$

Où α_m est la moyenne de la fraction de vapeur α_e en entrée et α_s en sortie. Dans cette formule, l'enthalpie h_{is} correspond au cas idéal d'une transformation isentropique, où l'entropie s_e à l'entrée du compresseur est identique à l'entropie en sortie.
Suivant cette définition, il est possible de calculer h_{is} en utilisant les fonctions du standard IF97 : tout d'abord la fonction $s_{Ph}(P,h)$ donnant l'entropie en fonction de la pression et de l'enthalpie est utilisée en entrée du compresseur, puis la fonction $h_{Ps}(P,s)$ qui est une de ses fonctions inverses est utilisée en sortie du compresseur :

$$\begin{aligned} s_e &= s_{Ph}(C_e.P, C_e.h) \\ h_{is} &= h_{Ps}(C_s.P, s_e) \end{aligned} \tag{2.17}$$

En réalité, dans ThermoSysPro, toutes les propriétés du fluide (c_p, ρ, T, α...) sont stockées dans des tables de propriétés *prop*, remplies en un seul appel à une fonction de la forme $prop := \text{fillProp}_{Ph}(P,h)$. Ainsi, les propriétés à l'entrée du compresseur sont dans une table $prop_e$ et on accède à s_e par $s_e := prop_e.s$.
On néglige par ailleurs l'accumulation de masse dans le compresseur (comme c'est fréquemment le cas pour les éléments biports). Pour cette raison, $C_e.q = -C_s.q = q$. Avec η_m le rendement mécanique du compresseur, la puissance mécanique soutirée peut alors être déterminée par :

$$W = \eta_m q * (C_s.h - C_1.h)$$

Ce système est simulé avec une rampe sur le débit en entrée. L'algorithme DASSL est utilisé avec une tolérance sur l'erreur relative de 10^{-4}. Les résultats sont montrés en Figure 2.16. On constate une baisse de la température et pression en sortie lorsque le débit augmente, ainsi qu'une augmentation de la puissance soutirée. Pour ce système, DASSL a utilisé des pas $T_k \in [0.01s, 4,89s]$ et une méthode d'ordre au plus 2. Ces informations font partie d'un fichier généré par Dymola nommé *dslog.txt*.

Par ailleurs, lors de la compilation d'un modèle, il est possible de générer avec Dymola un fichier nommé *dsmodel.mof* détaillant la causalisation du modèle par les étapes décrites en 2.2.2. Ce fichier présente notamment les boucles algébriques isolées par l'algorithme de Tearing. A titre d'exemple, une boucle algébrique est présente dans le compresseur car on a considéré pour le titre de vapeur moyen α_m : celui-ci dépend des propriétés du fluide en entrée et en sortie, or d'après (2.16) et (2.17) $C_s.h$ dépend de h_{is} qui à son tour dépend de α_m. L'algorithme de Tearing donne ainsi :

– Variables de séparation X^t

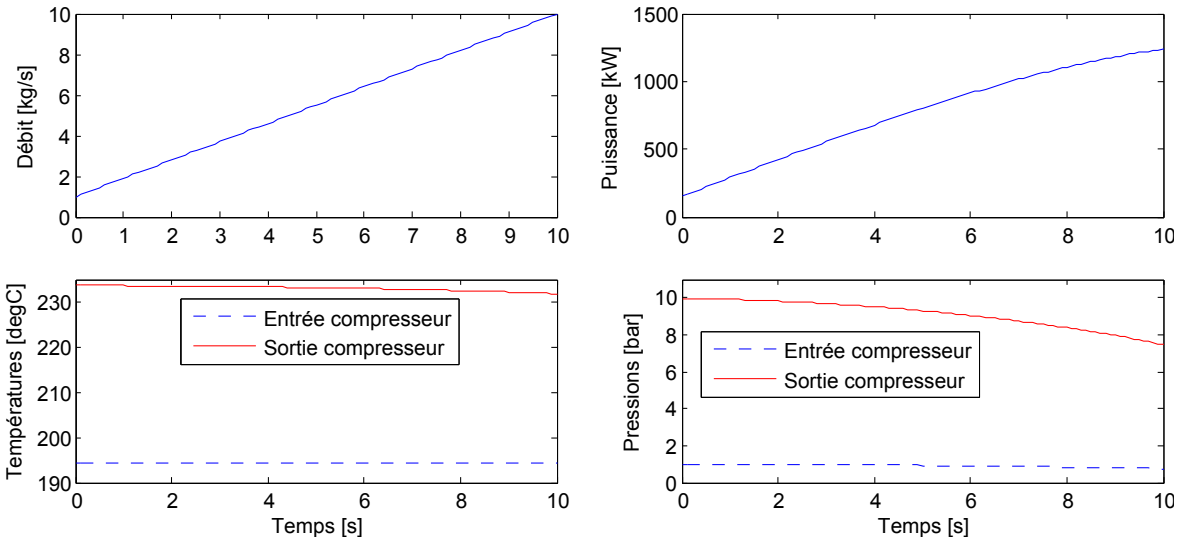


FIGURE 2.16 – Simulation du compresseur ThermoSysPro

  - $C_s.h$
  - α_m
- Variables séparées :
  - $\alpha_s := -\alpha_e + 2\alpha_m$
  - $prop_s := \text{fillProp}_{Ph}(C_s.P, C_s.h)$
- Equation des résidus $r = (r1, r_2) = (0, 0)$:
  - $r_1 := C_e.h - \alpha_m \eta_{is}(C_s.h - C_e.h)$
  - $r_2 := \alpha_s - prop_s.\alpha$

La résolution de l'équation des résidus $r(C_s.h, \alpha_m, props_s, \alpha_s) = 0$ se fait en utilisant l'algorithme de Newton, qui nécessite le calcul du jacobien $\nabla_{X^t} r$ de cette fonction. Dymola est capable de générer celui-ci de façon analytique en appliquant la procédure de différentiation automatique décrite en 2.2.2.4. Tout d'abord les expressions des variables séparées sont réintroduites dans l'équation des résidus pour aboutir à une équation des résidus de la forme $r(X^t) = r(C_s.h, \alpha_m) = 0$. On a par ailleurs besoin de la dérivée directionnelle de la fonction fillProp_{Ph} par rapport à sa variable $C_s.h$. Dymola génère automatiquement cette fonction dérivée $d_h(P, h)$ d'après la procédure décrite en [27].

$$d_h(P,h) = \left.\frac{\partial \text{fillProp}}{\partial h}\right|_{(P,h)}$$

En notant d_h^i la composante de d_h correspondant à la variable de séparation α_m. On aboutit au jacobien suivant :

$$\frac{\partial r}{\partial X^t} = \begin{pmatrix} -\alpha_m \eta_{is} & -(C_s.h - C_e.h)\eta_{is} \\ -d^i & 2 \end{pmatrix}$$

On a ici montré un exemple des transformations symboliques appliquées par Dymola. Le système était d'index 1 et n'a pas nécessité de réduction d'index mais comprenait deux boucles algébriques (dont celle présentée). Une remarque intéressante est que si l'on avait plutôt choisi le titre de vapeur α_e à l'entrée du compresseur dans l'équation 2.16 (ce qui est une

simplification acceptable dans certains cas), cette boucle algébrique aurait disparu. Il existe ainsi des compromis entre la justesse de la modélisation adoptée et une plus grande efficacité lors de la simulation (réduction du nombre de boucles algébriques ou de variables impliquées dans celles-ci).
Une autre remarque à ce sujet concerne le choix des variables thermodynamiques sélectionnées. A titre d'exemple, on a vu précédemment qu'en l'absence de changement de phase, un fluide pouvait être identiquement décrit par les couples (P, h) ou (P, T) car ces variables sont liées par des relations de la forme $T = f_1(P, h)$ ou $h = f_2(P, T)$. Il est possible que l'utilisateur écrive son modèle en appelant la fonction f_1 mais que T nécessite d'être calculé avant h lors de la causalisation. Cette situation amène à résoudre une boucle algébrique sur h alors qu'il aurait été plus judicieux d'utiliser la fonction inverse f_2. Afin d'éviter cette situation, le langage Modelica offre une syntaxe donnant la possibilité d'indiquer au compilateur l'existence de fonctions inverses.

2.3.2 Modèle simplifié de cogénération

Description du système

On détaille maintenant la modélisation simplifiée d'une centrale de cogénération alimentant un réseau de chaleur. Ce modèle sera utilisé dans le chapitre suivant traitant de l'optimisation dynamique. Les hypothèses simplificatrices utilisées permettent de disposer d'un modèle de taille adaptée à l'optimisation (qui est une procédure beaucoup plus gourmande que la simulation), tout en conservant une modélisation physique suffisamment précise pour nos besoins.
Le système est représenté en Figure 2.17. Le fluide de travail qui alimente le réseau de chaleur est de l'eau liquide. La centrale comporte un moteur produisant électricité et chaleur (récupérée au niveau du système de refroidissement du moteur et des fumées), une chaudière ne produisant que de la chaleur, un stockage thermique. Une branche de régulation q_{ret} est présente en entrée du réseau de chaleur afin de pouvoir contrôler la température fournie. La cogénération a pour fonction de fournir au réseau une puissance thermique W_{net} égale à la demande W_{demand}, à une température T_{net} proche de sa référence $T_{net,ref} = 87\ °C$. Le stock thermique peut contenir de l'eau entre 70 $°C$ et 90 $°C$. L'état de charge (SOC) du stock est un nombre compris entre 0 et 1 (0 si toute l'eau est à 70 $°C$, 1 si toute l'eau est à 90 $°C$).

On note q les débits, T les températures, W les puissances et $load$ les charges. Les indices suivants sont utilisés : f pour le combustible, h pour la chaleur, e pour le moteur, b pour la chaudière, sto pour le stock, net pour le réseau, c pour le retour. Pour le moteur et la chaudière, i désigne l'entrée et o la sortie. L'eau en sortie de ces deux composants est collectée dans un mélangeur noté mix. Les entrées contrôlées du système sont les suivantes :

- La charge du moteur $load_e = W_{fe}/W_{fe,max} \in [0; 1]$
- La charge de la chaudière $load_b = W_{fb}/W_{fb,max} \in [0; 1]$
- Le débit au réseau $q_{net} > 0$
- Le débit au moteur en marche $q_e \in [0\ kg/s, 20\ kg/s]$
- Le débit à la chaudière en marche $q_b \in [0\ kg/s, 15\ kg/s]$
- Le débit dans la branche de régulation $q_{ret} > 0$

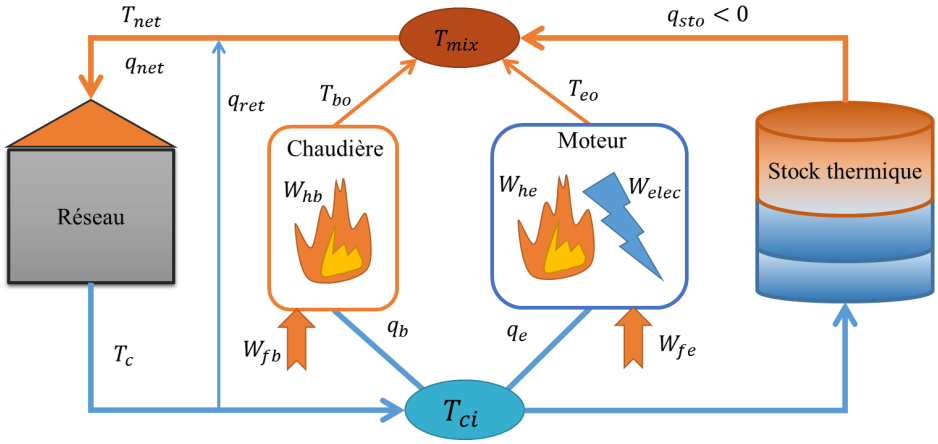


FIGURE 2.17 – Modèle simplifié de cogénération avec stockage

Hypothèses de modélisation

La plage de température utilisée dans l'installation est relativement restreinte ([70 °C, 90 °C]), et aucun changement de phase n'a lieu. Par ailleurs, comme on ne s'intéresse pas dans notre cas d'étude aux pertes de charge, on considèrera pour l'eau des propriétés constantes, en particulier la capacité calorifique C_p et la masse volumique ρ seront désormais considérées comme des paramètres. Ces hypothèses permettent d'écrire les équations de conservation de la masse et de l'énergie en fonction de la température uniquement.

Modélisation du réseau

On considère ici que la température de retour du réseau T_c est fixe. Par ailleurs, comme on ne s'intéresse pas ici à la dynamique du réseau, on considère une loi de conservation statique de l'énergie dans ce dernier :

$$W_{net} = q_{net} c_p (T_{net} - T_{ci}) \tag{2.18}$$

Modélisation du moteur et de la chaudière

On fait le choix ici de ne pas modéliser les réactions de combustion mais de décrire le moteur et la chaudière par des courbes de rendement (non-linéaires) η, fonction de la puissance combustible en entrée. La notation $\bullet$ est utilisée pour désigner selon le cas le moteur ou la chaudière : $\bullet = \{e; b\}$:

$$W_{h,\bullet} = \eta_\bullet(W_{f,\bullet}) W_{f,\bullet} \tag{2.19}$$

Le fluide en entrée du moteur et de la chaudière est à température T_{ci}. L'élévation de température est donnée par :

$$W_{h,\bullet} = q_\bullet C_p (T_{\bullet,o} - T_{ci})$$

Nœuds de mélange

Un nœud de mélange statique est utilisé pour la boucle de régulation. La conservation de la masse et de l'énergie s'écrivent :

$$\begin{aligned} q_{net} &= q_{net,mix} + q_{ret} \\ q_{net}T_{net} &= q_{net,mix}T_{mix} + q_{ret}T_c \end{aligned} \tag{2.20}$$

Deux nœuds de mélange dynamique sont placés en entrée et sortie du groupe moteur + chaudière (respectivement de température T_{ci} et T_{mix}). Une version statique de la conservation de la masse est utilisée comme précédemment (pas d'accumulation de masse) :

$$q_{net,mix} = q_b + q_e - q_{sto} \tag{2.21}$$

La conservation de l'énergie est en revanche modifiée. Dans le nœud de sortie, celle-ci s'écrit :

$$M_{mix}\frac{dT_{mix}}{dt} = q_eT_{eo} + q_bT_bo - (\text{posMax}_{q_\epsilon,w}(q_{sto}) + q_{net,mix})T_{mix} + \text{posMax}_{q_\epsilon,w}(-q_{sto})T_{sto,1}$$

On constate ici l'utilisation de la fonction posMax introduite en 2.1.2.6 pour traiter les inversions de débit dans la branche du stockage. Cette fonction n'était pas nécessaire dans l'équation de conservation de l'énergie précédente car les sens des débits étaient imposés ($q_{net} \geq q_{net,mix} > q_{ret} \geq 0$).

Modélisation du stockage

La fonction posMax est également utilisée dans la modélisation du stockage de chaleur. Il s'agit d'un stockage stratifié contenant une masse d'eau M_{sto}. La stratification consiste à laisser un gradient de température s'établir entre le haut et le bas de la cuve. Afin de maintenir cette stratification au mieux et ne pas mélanger eau chaude et froide, l'injection et le soutirage d'eau chaude se font par le haut. Pour modéliser le stockage, on utilise une variante du schéma à grille décalée décrit en 2.1.2.2. Le volume de stockage V est divisé en un nombre N de strates de volume V/N, et l'on écrit sur chacun de ces éléments les équations de conservation. Les propriétés intensives, ici les températures, sont calculées au centre des strates (dans lesquelles on suppose un mélange parfait). Dans notre cas on note $T_{sto,i}, i \in \overline{1..N}$ la température de la strate i. Les variables pour le calcul des flux sont quant à elles prises aux frontières des strates, on les note $T_i, i \in \overline{0..N}$. On retient pour cet exemple d'autres hypothèses simplificatrices : on néglige les effets de convection et la diffusion thermique entre les strates : il n'y a alors pas d'échange thermique à débit nul, toute l'énergie est transportée par le débit. Par ailleurs, on suppose qu'il n'y a pas d'accumulation de masse dans les éléments, c'est-à-dire que tous les éléments sont traversés par le même débit $q_{sto} = q_i$. Les variables T_i valent alors $T_{sto,i}$ si $q > 0$, $T_{sto,i+1}$ sinon et peuvent être éliminées des équations qui vont suivre. La conservation de la masse dans les strates (intermédiaires et connectées au reste du système) s'écrit ainsi :

$$\frac{M_{sto}}{N}\frac{dT_{sto,i}}{dt} = \begin{cases} \text{posMax}_{q_\epsilon,w}(q_{sto})(T_{net,mix} - T_{sto,1}) - \text{posMax}_{q_\epsilon,w}(-q_{sto})(T_{sto,1} - T_{sto,2}) & ,i = 1 \\ \text{posMax}_{q_\epsilon,w}(q_{sto})(T_{sto,i-1} - T_{sto,i}) - \text{posMax}_{q_\epsilon,w}(-q_{sto})(T_{sto,i} - T_{sto,i+1}) & ,i \in \overline{2..N-1} \\ \text{posMax}_{q_\epsilon,w}(q_{sto})(T_{sto,N-1} - T_{sto,N}) - \text{posMax}_{q_\epsilon,w}(-q_{sto})(T_{sto,N} - T_{ci}) & ,i = N \end{cases} \tag{2.22}$$

Simulation du système

On introduit ici la problématique de pilotage du système. Le pilotage du système consiste à déterminer en chaque instant la valeur des commandes $u(t)$ de telle sorte que les contraintes de l'installation soient respectées. Dans le chapitre 3, cette tâche sera réalisée en faisant appel à l'optimisation dynamique. Une autre approche consiste à intégrer des régulations prédéfinies au sein du modèle de simulation. Celles-ci permettent de déterminer explicitement la commande $u(t)$ en fonction de mesures sur l'état $x(t)$ ou la sortie $y(t)$. C'est une approche qui reste cependant spécifique à chaque installation. Pour l'illustrer, on donne ici un exemple de régulation possible pour la cogénération :

- On suppose que les entrées $load_e$ et $load_b$ sont imposées par l'exploitant
- On fixe les débits dans le moteur et la chaudière à leur valeur maximale lorsque le composant est en marche (respectivement 20 kg/s et 15 kg/s), au débit minimal q_ϵ sinon. Cette contrainte impose les températures en sortie du moteur d'après (2.19).
- L'objectif de la régulation est d'ajuster les commandes q_{net} et q_{ret} afin de fournir au réseau une puissance W_{net} égale à la demande W_{demand}, à une température T_{net} égale à la valeur de consigne $T_{net,ref}$. On suppose dans un premier temps que la régulation pour ces deux grandeurs est idéale. Dans ce cas, les commandes q_{net} et q_{ret} à appliquer peuvent être déterminées par inversion du modèle, en ajoutant les égalités $W_{net} = W_{demand}$ et $T_{net} = T_{net,ref}$ (ces deux débits peuvent en effet être calculés d'après la conservation de l'énergie dans le réseau (2.18) et la loi de mélange des températures (2.20).

Le débit au stockage qui n'est pas un débit commandé peut dès lors être calculé d'après la conservation de la masse (2.21).
L'application de cette loi de commande est illustrée en Figure 2.18. Malheureusement, cette régulation assez simpliste ne suffit pas la plupart du temps à respecter les contraintes de l'installation. A titre d'exemple, on montre en Figure 2.19 que le débit q_{ret} dans la branche de régulation (calculé par inversion) ne respecte pas toujours la contrainte $q_{ret} > 0$. Dans ce cas, l'ajout d'un contrôleur garantissant le respect des contraintes sur les commandes est alors préférable à un calcul inverse (on aurait pu utiliser par exemple un PI avec anti-windup). En revanche, il est moins évident de garantir le respect des contraintes sur les sorties et états du modèle, puisque leur évolution ne dépend qu'implicitement des commandes. Par ailleurs, des couplages non-linéaires peuvent exister entre les différentes variables. Pour conclure, la mise en place d'une régulation assurant le respect simultané de toutes les contraintes et à tout instant nécessite une connaissance avancée de l'installation.
La solution que l'on propose, basée sur l'optimisation de modèles physiques, est une autre approche plus générique à ce problème. Elle permet de prendre en considération les contraintes de l'installation à la fois sur les commandes mais aussi sur les états et sorties du système, tout en optimisant la performance de la régulation vis à vis d'un critère.

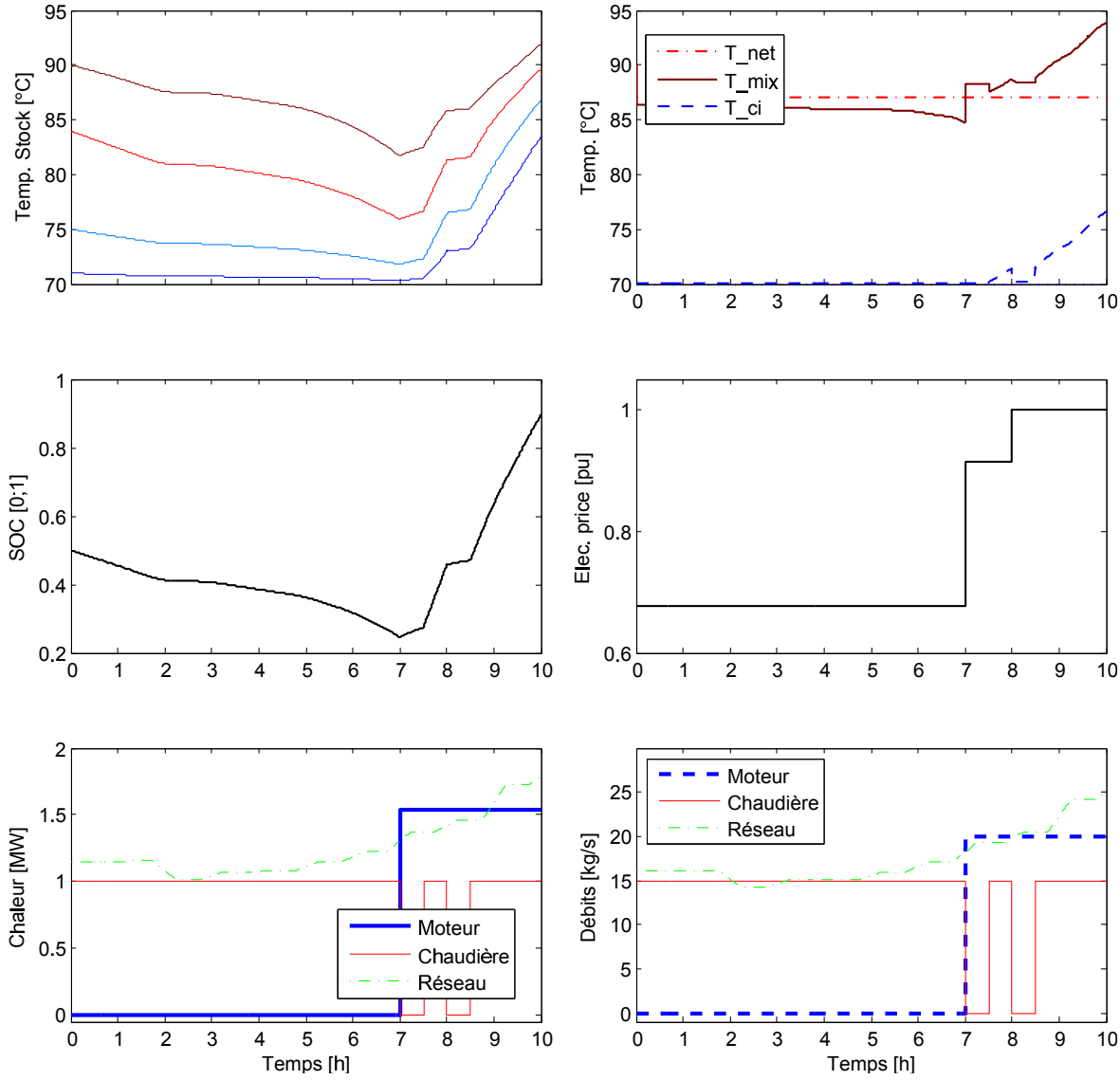


FIGURE 2.18 – Simulation de la centrale de cogénération avec régulation automatique des débits

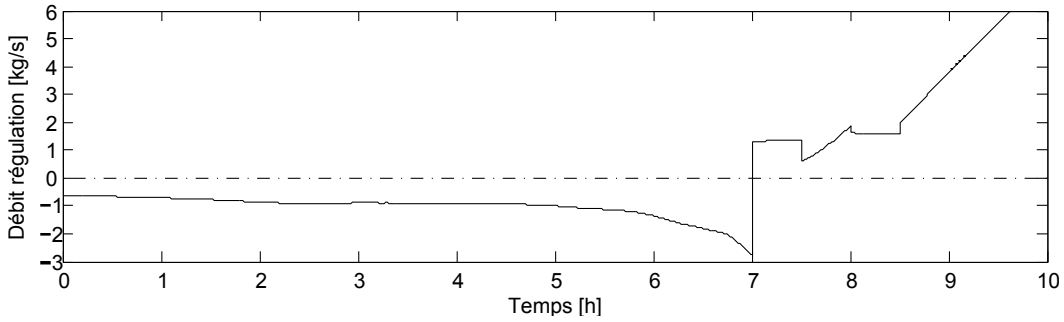


FIGURE 2.19 – Simulation de la centrale de cogénération avec régulation automatique des débits : non respect de la contrainte sur q_{ret}

CHAPITRE 3

Optimisation dynamique pour les systèmes hybrides

Dans ce chapitre, on s'intéresse à la fonction "Optimisation de trajectoire long terme" présentée dans le Chapitre 1 que l'on peut voir en Figure 1.1. On présente une méthode d'optimisation dynamique sous contrainte pour les centrales d'énergie. Cette méthode a pour but de générer des trajectoires optimales pour les commandes du système, sur un horizon d'une ou plusieurs journées. Après avoir introduit certains rappels sur l'optimisation dynamique des modèles continus, on fait le constat que les systèmes énergétiques modélisés font aussi intervenir des aspects hybrides, qui devront être pris en compte dans la résolution du problème d'optimisation. Ceux-ci incluent notamment la présence de composants pouvant être inactivés, ou de composants dont les équations diffèrent selon le mode de fonctionnement.
On formalise alors un problème de commande optimale hybride, puis on introduit une classe de méthodes candidates pour sa résolution : les méthodes directes. En particulier, on présente la méthode de collocation qui sera à la base de l'algorithme développé dans cette thèse. Cependant, dans leur forme originale, les méthodes directes ne s'appliquent qu'à des modèles continus et ne permettent pas de prendre en compte les aspects hybrides que l'on vient de mentionner. On présente alors en parallèle certaines des méthodes numériques couramment utilisées pour traiter les aspects hybrides en optimisation.
La méthode développée, qui combine optimisation dynamique et traitement des aspects hybrides, est enfin introduite. On valide pour conclure son application sur l'exemple d'une cogénération.

3.1 Problème d'optimisation dynamique hybride

3.1.1 Problème d'optimisation dynamique

3.1.1.1 Formulation du problème de commande optimale

On s'intéresse tout d'abord à l'optimisation dynamique de systèmes DAE non hybrides, puis l'on étendra les résultats présentés à une classe de DAE hybrides. On suppose désormais que le modèle est donné sous forme de DAE semi-explicite (DAEse), ou qu'il est possible de

s'y ramener lors de la compilation du code Modelica, en utilisant les algorithmes décrits précédemment en partie 2.2.2. Cette hypothèse implique que les variables algébriques z peuvent être déterminées implicitement d'après les états x. On utilisera parfois la notation z^x pour indiquer que z peut être calculé à partir de x à tout instant.

Le problème d'optimisation dynamique, ou problème de commande optimale (OCP, *Optimal Control Problem*) consiste à déterminer sur un horizon $[t_0, t_f]$ les commandes $u(t)$ à appliquer au système afin de minimiser une fonction objectif, tout en respectant un ensemble de contraintes.
On considère une fonction objectif scalaire J sous la forme générale dite forme de Bolza. La forme de Bolza est constituée d'un terme intégral nommé terme de Lagrange et d'un coût terminal nommé terme de Mayer. La valeur du critère s'exprime ainsi par :

$$J(u, t_f) = \underbrace{\int_{t_0}^{t_f} L(x(t), z^x(t), u(t))dt}_{\text{terme de Lagrange}} + \underbrace{\Phi(x(t_f))}_{\text{terme de Mayer}} \tag{3.1}$$

Le problème de commande optimale sous contrainte se formule :

$$\min_{u(\bullet)} \quad J(u, t_f) \tag{3.2a}$$

$$\text{s.c.}(DAEse) \qquad \text{Equations algébro-différentielles} \tag{3.2b}$$

$$h_I(x(t), z(t), u(t)) \leq 0 \qquad \text{Contraintes inégalités sur les variables} \tag{3.2c}$$

$$r(x(t_f)) = 0 \qquad \text{Contraintes finales} \tag{3.2d}$$

On notera J^* le coût optimal sur l'horizon $[t_0; t_f]$, u^* et x^* les trajectoire optimales associées.

Hypothèse H1. *(Faisabilité) On suppose qu'il existe au moins une commande u permettant de respecter les équations du modèle (DAEse) et les contraintes (3.2).*

On cherche désormais à résoudre ce problème. On commence pour cela par présenter un résultat classique permettant de donner des conditions nécessaires d'optimalité de ce problème : le principe du minimum. Des rappels sur les conditions d'optimalité en dimension finie sont donnés en Annexe A.

3.1.1.2 Conditions d'optimalité en optimisation dynamique : Principe du minimum

On s'intéresse maintenant aux conditions d'optimalité du problème (3.3), que l'on donne pour des raisons de simplicité dans le cas d'un système sous forme d'ODE (l'hypothèse que le système est d'index 1 garantit qu'il est possible de s'y ramener).
La recherche d'une trajectoire de commande optimale est un problème de dimension infinie, car elle consiste à trouver pour tout instant $t \in [0, t_f]$ la valeur des commandes à appliquer. On se trouve ainsi dans le cadre de l'optimisation d'une fonctionnelle, c'est-à-dire d'une fonction prenant en argument des fonctions (ici la commande $u(t)$) et non des variables. Pour cela, une théorie différente de l'optimisation en dimension finie est nécessaire. On dérive dans ce cas des

conditions nécessaires d'optimalité non plus autour d'un point mais autour d'une trajectoire optimale $(x^*(t), u^*(t))$ en utilisant le calcul des variations. Ces conditions ne seront cependant que des conditions d'optimalité locale.
Par ailleurs, dans le problème de commande optimale, la contrainte de respect de la dynamique doit être satisfaite à chaque instant. Les "multiplicateurs de Lagrange" associés aux équations dynamiques deviennent ainsi variants dans le temps eux aussi. Pour cela, on préfère dès lors les nommer états adjoints associés aux contraintes dynamiques $\dot{x}_i - f_i(x,u) = 0, \forall i \in \overline{1..n_x}$.
Le Lagrangien pour le problème de commande optimale avec l'état adjoint $\lambda(t)$ s'écrit :

$$\min_{u(\bullet),\lambda(\bullet)} \quad \bar{J}(x,u,\lambda) = \Phi\left(x(t_f)\right) + \int_{t_0}^{t_f} L(x,u)dt + \lambda^T(t)\left(f(x,u) - \dot{x}\right)dt \tag{3.3}$$

$$\text{s.c.} \quad \dot{x}(t) = f(x(t),u(t)) \tag{3.4}$$

$$x(t_0) = x_0 \tag{3.5}$$

$$h_I(x(t),u(t)) \leq 0 \quad \forall t \in [t_0,t_f] \tag{3.6}$$

$$r(x(t_f)) = 0 \tag{3.7}$$

On introduit la fonction $\mathcal{H}$ nommée Hamiltonien du système :

$$\mathcal{H}(x,u,\lambda,\mu) = L(x,u) + \lambda^T f(x,u) + \mu^T h_I(x,u)$$

avec μ le multiplicateur associé à la contrainte (3.6). Ainsi que le Lagrangien au temps final :

$$\mathcal{L}_f(x(t_f),\nu) = \Phi\left(x(t_f)\right) + \nu^T r(x(t_f)) \tag{3.8}$$

avec ν le multiplicateur associé à la contrainte finale (3.7). Les conditions nécessaires d'optimalité de ce problème sont donnés dans un célèbre théorème : le principe du minimum [28]. Ce principe permet de calculer la commande optimale $u^*(t)$ point à point (à chaque instant t).

Théorème 3.1. *Principe du minimum : Si le problème de commande optimale (3.3) admet une solution $(x^*(t), u^*(t), \lambda^*(t))$, alors il existe des multiplicateurs de Lagrange $\lambda^*(t)$ aussi nommés états adjoints, ainsi que $\mu^*(t)$ et ν^* tels que presque partout sur $[t_0,t_f]$ on ait :*

$$\dot{x} = \frac{\partial \mathcal{H}}{\partial \lambda} = f(x,u^*) \tag{3.9a}$$

$$x(t_0) = x_0 \tag{3.9b}$$

$$\dot{\lambda}^{*T} = -\frac{\partial \mathcal{H}}{\partial x} \tag{3.9c}$$

$$\lambda^{*T}(t_f) = -\frac{\partial \mathcal{L}_f}{\partial x^*}(x^*(t_f),u^*(t_f),\lambda^*(t_f),\mu^*(t_f),\nu^*) \tag{3.9d}$$

$$h_I(x^*(t),u^*(t)) \leq 0 \tag{3.9e}$$

$$r(x(t_f)) = 0 \tag{3.9f}$$

$$\mu^*(t) \geq 0 \tag{3.9g}$$

$$0 = \mu^*(t)c_i(x^*(t),u^*(t)) \tag{3.9h}$$

$$u^*(t) = \arg\min_v \mathcal{H}(x^*(t),v,\lambda^*(t),\mu^*(t)) \tag{3.9i}$$

Le problème à résoudre est dit problème aux deux bouts car il y a une condition initiale sur les variables primales $x(t)$ et une condition finale sur les variables adjointes $\lambda(t)$.
La dernière équation (3.9i) est une condition de minimisation point à point qui permet d'obtenir une trajectoire pour les commandes : dans le cas sans contrainte, $u(t)$ peut ainsi se calculer à tout instant d'après :

$$0 = \frac{\partial \mathcal{H}}{\partial u} = \lambda^T \frac{\partial f}{\partial u} + \mu^T \frac{\partial h_I}{\partial u} \tag{3.10}$$

Dans le cas où u n'apparaît pas explicitement dans l'expression, l'équation (3.9i) doit être différenciée jusqu'à obtenir une dépendance en u. On parle dans ce cas de contrôle optimal singulier.
Toutes les méthodes de résolution numérique du problème de commande optimale doivent ainsi permettre de retrouver ces conditions d'optimalité. On introduit maintenant les plus courantes de ces méthodes.

Les algorithmes de résolution du problème de commande optimale peuvent se scinder en deux catégories : les méthodes directes, et les méthodes indirectes. Dans les méthodes indirectes, on recherche des trajectoires pour $u(t)$, $x(t)$ et $\lambda(t)$ qui vérifient les conditions nécessaires d'optimalité en chaque instant tout en respectant la contrainte initiale sur x et finale sur λ. La résolution de ce problème aux deux bouts nécessite pour cela l'intégration numérique des trajectoires de $x(t)$ et de $\lambda(t)$. Pour cette raison, les méthodes indirectes sont parfois nommées *First optimize, then discretize*.

A l'inverse, dans les méthodes directes, le problème (3.3) est tout d'abord discrétisé dans le temps. Le problème initialement de dimension infinie devient alors un problème de dimension finie que l'on peut traiter numériquement. Pour cette raison, on appelle parfois les méthodes directes *First discretize, then optimize*.
Les méthodes indirectes sont connues pour leur grande précision numérique mais la prise en compte des contraintes y est particulièrement difficile. Leur convergence nécessite également une initialisation proche de la solution optimale et une connaissance de la structure d'activation des contraintes (c'est-à-dire l'ordre dans lequel celles-ci deviennent actives ou inactives) si celles-ci sont présentes, ce qui induit une dimension combinatoire. A l'inverse, les méthodes directes permettent une prise en compte aisée des contraintes, mais peuvent être moins précises [29].

Pour les systèmes considérés dans cette thèse, qui sont des systèmes de grande taille soumis à de nombreuses contraintes, et pour lesquels la précision de l'optimisation n'est pas cruciale, les méthodes directes sont les plus adaptées. On s'intéressera donc par la suite aux méthodes directes uniquement, que l'on présentera dans la partie 3.2.
On reviendra ensuite sur un des aspects que l'on a négligé jusqu'alors : les systèmes considérés contiennent certains aspects hybrides. Le formalisme DAE doit ainsi être adapté pour les prendre en compte. De même, les méthodes directes que l'on va présenter en 3.2 devront être modifiées. Ces adaptations sont au centre de la méthode d'optimisation développée dans cette thèse, que l'on présente en 3.4.

3.1.2 Modèles considérés et modes dans les systèmes hybrides

3.1.2.1 Aspects hybrides dans les systèmes énergétiques

La modélisation de centrales d'énergie peut faire intervenir des aspects hybrides : un système peut ainsi présenter plusieurs configurations, aussi nommées modes, dans lesquels les équations de fonctionnement ou les paramètres vont différer. On peut citer à titre d'exemple les phénomènes suivants :

- Inversions de débits : lorsque les phénomènes de diffusion ne sont pas modélisés dans les volumes de contrôle, l'enthalpie de mélange est prise comme celle en amont du volume dans le sens du débit. On obtient ainsi une discontinuité dans l'enthalpie de mélange lorsque le débit s'inverse.
- Tables de l'eau : les tables de l'eau donnent les propriétés physiques de celle-ci. Les tables sont divisées en 5 zones, chacune définie par des équations propres. De plus, les propriétés ne sont pas différentiables à la jonction entre les zones.
- Composants pouvant être allumés ou éteints
- Automatisme pouvant comporter des hystéresis : une mémoire de l'état (montant ou descendant) doit être stockée
- Vannes anti-retour : l'équation de perte de charge n'est valable que dans un sens du débit.
- etc.

Le formalisme DAE est étendu à celui des *Équations Algébro Différentielles Hybrides* (*H-DAE*). La modélisation fait ainsi apparaître des branches conditionnelles. On peut alors introduire des variables logiques $Y \in \{0;1\}$ pour déterminer si une branche est active (1) ou non (0), et donc le jeu d'équation du système à un instant donné. On appelle modes les configurations possibles du système : un mode correspond à une valeur particulière prise pour les variables logiques. Suivant cette définition, un seul mode est actif à chaque instant.
Dans les notations ci-dessous, les variables logiques seront indifféremment définies par leur représentation sous forme binaire ($\{0;1\}$) ou sous forme de booléen (V/F).

Définition (Disjonction) On appelle disjonction un ensemble de variables logiques mutuellement exclusives. Parmi elles exactement une des variables vaut 1, les autres 0. Une disjonction D de n_Y termes Y^i est représentée par un ensemble d'entiers $\overline{1..n_Y}$ codant pour chacun des choix possibles. La relation correspondant à une disjonction est notée $\underline{\vee}_{i\in\overline{1..n_Y}} Y^i$ ($\underline{\vee}$ sans indice désignant le "OU EXCLUSIF") :

$$\underline{\bigvee}_{i\in\overline{1..n_Y}} Y^i = \{(Y^1 \underline{\vee} Y^2 \underline{\vee} \cdots \underline{\vee} Y^{n_Y}) = 1\}$$

Pour un système hybride, on peut ainsi dire qu'il existe une disjonction entre tous les modes possibles car seul un mode est actif à chaque instant, c'est-à-dire une seule combinaison des variables logiques parmi toutes celles possibles.
On s'intéresse maintenant à la façon de représenter les modes. Dans les systèmes hybrides que nous étudions, les variables logiques correspondent souvent à l'état d'un composant du système. Les différents modes possibles peuvent ainsi soit être définis en énumérant les différentes combinaisons possibles des variables logiques, soit être définis comme la réunion des modes des différents composants. Ces deux formes sont respectivement appelées la forme normale disjonctive et conjonctive.
Formes Normales Disjonctives (DNF) et Conjonctives (CNF) :

Pour éclaircir ce propos, on donne l'exemple d'un système possédant deux composants indexés par $j \in \{1;2\}$ ayant chacun deux états possibles : soit allumé ($i = 1$), soit éteint ($i = 2$). On note Y^{ji} les variables logiques décrivant l'activation de la branche "*le composant j est dans l'état i*". Le mode courant correspond donc à une valeur particulière du vecteur des variables logiques Y. Son ensemble de définition est constitué des $n_\omega = 4$ modes suivants :

$$\Omega(Y) = \{\underbrace{(Y^{11}, \neg Y^{12}, Y^{21}, \neg Y^{22})}_{\omega^1}, \underbrace{(Y^{11}, \neg Y^{12}, \neg Y^{21}, Y^{22})}_{\omega^2}, \underbrace{(\neg Y^{11}, Y^{12}, Y^{21}, \neg Y^{22})}_{\omega^3}, \underbrace{(\neg Y^{11}, Y^{12}, \neg Y^{21}, Y^{22})}_{\omega^4}\} \quad (3.11)$$

L'ensemble admissible pour les variables logiques est ainsi décrit par :

$$\Omega(Y) = \omega^1(Y) \veebar \omega^2(Y) \veebar \omega^3(Y) \veebar \omega^4(Y)$$

Cette forme est appelée forme normale disjonctive (DNF) [30]. Le mode actif, nommé m est défini comme :

$$m(t) = \{i \quad \text{t.q. } \omega^i(Y;t) = 1\}$$

Cependant, il existe une autre façon de considérer les modes : en regardant le système composant par composant. Dans ce cas, il n'y aura plus une disjonction parmi tous les modes, mais n_d disjonctions :

$$\Omega(Y) = (Y^{11} \veebar Y^{12}) \wedge (Y^{21} \veebar Y^{22})$$

Cette seconde forme est nommée forme normale conjonctive (CNF). Dans la forme conjonctive, plusieurs variables sont nécessaires pour décrire le mode courant, que l'on code comme un vecteur $\mathbf{m} = \begin{pmatrix} m_1 & \dots & m_{n_d} \end{pmatrix}^T$. Les composantes m_j de ce vecteur correspondent à la branche active de chaque disjonction j, ainsi :

$$m_j(t) = \{i \in \veebar D_j = \overline{1..n_\omega^j} \text{ t.q. } Y^{ji}(t) = 1\} \quad \forall j \in \overline{1..n_d}$$

Elimination de variables logiques : dans l'exemple présenté, chaque composant a seulement deux états possibles, on a $Y^{i1} = \neg Y^{i2}$ et une de ces variable peut être éliminée, mais nous ne le faisons pas ici dans un souci de généralité (on pourrait par exemple considérer ici un troisième état "en veille" pour les composants).

3.1.2.2 Traitement des singularités

Une modélisation hybride peut être adoptée pour les composants d'un système pouvant être inactivés. Cependant, dans le cadre des systèmes énergétiques, une telle modélisation doit être effectuée avec précaution : en effet, pour ces systèmes, l'inactivation d'un composant correspond souvent à la présence de puissance ou débit nuls dans la branche où celui-ci se trouve, ce qui peut amener à des singularités. Pour l'illustrer, on considère l'exemple simple d'un échangeur de chaleur statique où le flux W côté chaud est fixé. On suppose que la température côté froid T_{ci} est donnée et l'on recherche la température T_{co} en sortie en fonction du débit q_c. Le liquide dans la branche froide vérifie :

$$W = q_c c_p (T_{co} - T_{ci})$$

On constate que cette équation est problématique pour les débits nuls :

$$T_{co} = T_{ci} + W/(q_c c_p)$$

L'équation de conservation statique ne permet plus de déterminer T_{co}. Ce problème peut être résolu par l'ajout d'un volume de contrôle dynamique en sortie de l'échangeur, qui jouera le rôle de "mémoire" pour T_{co}, au prix d'une augmentation de la taille du système. Si q_c est un débit contrôlé, on peut également imposer un débit minimum $q_\epsilon > 0$. Une troisième approche consiste enfin à disposer de plusieurs modèles, un contenant cette branche et possédant $q_c \geq q_{min}$ comme entrée, l'autre ne contenant pas cette branche et non influencé par q_c. Cette approche permet d'éviter les singularités, mais requiert que l'algorithme d'optimisation combine les différents modèles. Nous ne retiendrons pas cette approche par la suite. Enfin, comme évoqué dans le chapitre précédent, le cas des inversions de débit, qui peut induire des discontinuités dans l'enthalpie nécessite aussi certaines régularisations, comme celles introduites en 2.1.2.6.
Pour conclure, une attention particulière doit ainsi être portée dans la définition des équations et contraintes afin d'éviter la présence de singularités dans les modèles d'optimisation.

3.1.2.3 Transitions dans les systèmes hybrides

3.1.2.3.1 Types de transitions

Le passage d'un mode au suivant est appelé transition. Il est possible de définir plusieurs classes de transitions qui n'auront pas le même impact sur les méthodes d'optimisation employées par la suite.

1. Les transitions explicites sont celles pour lesquelles l'instant de commutation est déterminé de façon explicite. Par exemple, un composant se met en marche à un instant connu.
2. Les transitions explicites commandées sont un cas particulier où l'instant de commutation est déterminé par une entrée commandée du système. Par exemple une vanne commandée qui était fermée est ouverte.
3. Les transitions implicites dépendent de l'état du système et ne peuvent résulter qu'implicitement des commandes appliquées au système. On peut citer l'exemple d'un thermostat déclenchant un chauffage lorsque la température $T < T_{on}$ et l'arrêtant lorsque $T > T_{off}$.
4. Les transitions stochastiques sont déclenchées à un instant de transition qui suit une loi aléatoire. Par exemple un composant devient défectueux après un certain temps.

3.1.2.3.2 Fonction de transition

Une façon d'établir la transition d'un mode au suivant est d'utiliser une fonction de transition. Soit $\phi^{a,b}(x, z, u, t)$ la fonction de transition du mode a au mode b. Supposons que le mode a soit actif pour $t \in [t_{k-1}, t_k]$ et que la transition de a à b ait lieu à l'instant t_k. Alors :

$$\begin{cases} \phi^{a,b}(x(t), z(t), u(t), t) > 0 & t \in [t_{k-1}, t_k] \\ \phi^{a,b}(x(t_k), z(t_k), u(t_k), t_k) = 0 \end{cases} \tag{3.12}$$

L'instant de transition correspond à l'instant où la fonction de transition s'annule. Les différents types de transition présentés ci-dessus correspondent à différentes fonctions de transition :
Les fonctions de transition explicites sont de la forme

$$\phi^{a,b}(u(t),t)$$

Les fonctions de transitions explicites commandées sont de la forme

$$\phi^{a,b}(u(t))$$

Les fonctions de transition implicites sont de la forme

$$\phi^{a,b}(x(t),z(t),u(t),t)$$

Le cas des transitions stochastiques sort de notre cas d'étude déterministe.

Dans la méthode d'optimisation développée, on considèrera uniquement le cas des transitions explicites commandées, et l'on supposera que les transitions implicites peuvent quant à elles être reformulées de façon continue par lissage, un point qui a déjà abordé pour les inversions de débit. Des reformulations similaires pour un grand nombre de composants thermo-hydrauliques sont par exemple données dans la librairie Modelica *ThermoOpt* détaillée dans [8].
On doit donc introduire une commande pour déclencher les transitions. La commande introduite est une commande logique codant le mode courant, que l'on note par la suite ω pour la différentier des commandes continues u. On adopte pour ω la Forme Normale Conjonctive : en d'autres termes, cela suppose que le mode du système peut être décrit comme la réunion des modes des différents composants.

3.1.3 Problème d'optimisation dynamique hybride

On formalise maintenant le problème d'optimisation dynamique hybride que l'on cherche à résoudre. Comme on s'intéresse aux systèmes hybrides avec transitions commandées, le mode actif fait partie des variables de décision au même titre que les commandes continues u. Les restrictions suivantes sont faites sur les DAE considérés :

Hypothèse H2. *La dimension des variables différentielles x et algébriques z est invariante. De plus l'état x reste continu aux instants de transition.*

Le point suivant introduit une séparation entre les équations du modèle valables pour toutes les configurations, et les équations spécifiques à la configuration courante. Cela évite la répétition des équations globales dans chaque terme de la disjonction. On suppose donc que les variables différentielles x et algébriques z peuvent être décomposées en plusieurs sous ensembles : les variables "globales" décrites par des équations générales à tous les modes notées x^g, z^g et des variables décrites par les équations spécifiques à chaque disjonction x^j, z^j, où j fait référence à la $j^{\text{ème}}$ disjonction D_j, $\quad j \in \overline{1..n_d}$. On utilise la notation $\setminus$ pour décrire le complément des variables d'un certain type. Ainsi, les variables spécifiques aux modes x^m et z^m (non "globales") sont définies $x^m = x \setminus x^g$ et $z^m = z \setminus z^g$. $u(t)$ est le vecteur des commandes continues. On introduit des commandes binaires $\omega^{ji}(t) \in \{0;1\} \quad i \in D_j$ qui

indiquent si la branche i de la disjonction D_j est active à l'instant t. La notation $\veebar$ empruntée à la programmation disjonctive [31] est un **OU EXCLUSIF n-aire** : les équations listées sous ω^{ji} ne sont ajoutées au système que pour la variable valant $\omega^{ji}(t) = 1$. On considère les systèmes de la forme :

$$\begin{aligned} \dot{x}^g(t) &= f(x(t), z(t), u(t)) \\ x(t_0) &= x_0 \\ z^g &= \bar{g}^g(x(t), z(t), u(t)) \\ 0 &\geq h_I^g(x(t), z(t), u(t)) \end{aligned} \tag{DAEg}$$

$$\bigcup_{j \in \overline{1..n_d}} \veebar_{i \in D_j} \omega^{ji}(t) \begin{pmatrix} \omega^{ji}(t) \\ \dot{x}^j(t) = f_i^j(x(t), z(t), u(t)) \\ z^j(t) = \bar{g}_i^j(x(t), z(t), u(t)) \\ 0 \geq h_{I,i}^j(x(t), z(t), u(t)) \end{pmatrix} \tag{DAEm}$$

Hypothèse H3. *Le système (DAEg-DAEm) est un DAE d'index au plus 1 pour tous les modes.*

Le problème d'optimisation dynamique hybride consiste donc à déterminer sur un horizon $[t_0, t_f]$ la séquence optimale d'activation des modes $(i(t))$, ainsi que les commandes continues $u(t)$ à appliquer au système, tout en respectant les contraintes h_I^g, $h_{I,i}^j$.

$$\begin{aligned} \min_{u(t),\omega(t)} \quad & J = \Phi(x(t_f)) + \textstyle\int_{t_0}^{t_f} L(x,u)dt \\ \text{s.c.} \quad & (DAEg), (DAEm) \end{aligned} \tag{H-OCP}$$

Avec J une fonction scalaire, Φ la fonction coût terminal, L la fonction coût intégral, $x = \left[x^{gT}; x^{mT}\right]^T$ et $z = \left[z^{gT}; z^{mT}\right]^T$. On note cependant qu'il est toujours possible de se ramener uniquement à un terme de Mayer ou de Lagrange en introduisant un état supplémentaire x_J tel que $\frac{dx_J}{dt} = L(x,u,t) + \frac{d\Phi(x(t)}{dt}, \quad x_J(0) = 0$ [32]. En effet, $J = \int_{t_0}^{t_f} [L(x,u,t) + \frac{d\Phi(x(t)}{dt}]dt = x_J(t_f)$.

La dimension combinatoire de ce problème a été réduite en considérant la forme conjonctive plutôt que disjonctive (il n'est pas nécessaire d'énumérer toutes les combinaisons possibles de ω^{ji} pour décrire le mode courant). Cependant, un autre aspect combinatoire provient de la dimension temporelle du problème lorsqu'on cherche à le résoudre numériquement : une grille temporelle de dimension finie est alors souvent utilisée pour définir les instants possibles de commutation. Dans le cas où toutes les transitions entre modes sont autorisées, le nombre de séquences possibles sur une grille de n_e éléments s'élève ainsi à $N_s = (n_d \sum_j n_\omega^j)^{n_e}$. On note aussi par ailleurs que la discrétisation temporelle retenue pour les aspects hybrides contraint les instants de commutation du problème.

3.2 Méthodes directes en optimisation dynamique

Les paragraphes précédents ont présenté le problème d'optimisation dynamique hybride que l'on cherche à résoudre. Celui-ci diffère du problème d'optimisation dynamique introduit

en 3.1.1 par l'ajout de commandes ω^{ji} qui n'ont pas valeur dans un sous ensemble de $\mathbb{R}$ mais dans l'ensemble discret $\{0;1\}$. Pour cette raison, les conditions d'optimalité présentées en 3.1.1.2 ne s'appliquent plus. Cependant, on verra dans les prochaines parties comment il est possible de relaxer le problème hybride afin de se ramener à un problème d'optimisation continu. On pourra alors employer des méthodes de résolution classiques pour le problème relaxé. Dans cette partie, on introduit la classe de méthodes numériques que l'on utilise par la suite pour la résolution du problème de commande optimale : les méthodes dites directes. Les méthodes directes procèdent en discrétisant problème d'optimisation avant sa résolution. L'horizon est segmenté en n_e éléments de durée $T_k = t_k - t_{k-1}$ (eux même contenant n_c points intermédiaires dans certaines méthodes). Les commandes sont discrétisées par projection sur une base de fonctions, on parle de paramétrisation du vecteur de commande (*Control Vector Parametrization*).

Dans les méthodes dites de tir direct, la fonction de commande est envoyée à un simulateur, qui renvoie alors la trajectoire associée et les informations de sensibilité nécessaires à l'optimiseur. On parle pour cela de méthodes séquentielles. En revanche, dans la méthode dite de collocation, on se passe du simulateur en intégrant au problème d'optimisation le schéma d'intégration numérique des variables d'état. On parle pour cela de méthodes simultanées.

3.2.1 Choix de la paramétrisation (grille et base de fonctions)

Le choix de la grille et de la base de fonctions utilisées pour paramétrer les trajectoires des commandes et des états a une grande influence sur les résultats des méthodes directes. La paramétrisation utilisée dans le problème discrétisé doit en effet laisser assez de degrés de liberté pour permettre de reproduire la trajectoire optimale que l'on aurait obtenue pour le problème en dimension infinie. On commence donc par s'intéresser aux caractéristiques de ces trajectoires de commande.
On nomme arc un intervalle de temps pendant lequel un certain jeu de contraintes est actif (la définition des contraintes actives est donnée en Annexe A). Une première remarque est que la trajectoire optimale consiste en une succession d'arcs qui dépendent des contraintes actives [33][34]. En effet, l'activation d'une contrainte équivaut à ajouter une contrainte égalité au problème pendant une certaine durée, ce qui peut amener à des couplages entre variables susceptibles de modifier l'index du système. Ainsi, le nombre de commandes libres du système, c'est-à-dire de commandes non déterminées par une contrainte active, peut varier dans le temps. On peut ainsi identifier quatre types d'arcs pour les commandes :

- la commande est au maximum : $u^i = u^i{}_{max}$ (arc maximal)
- la commande est au minimum : $u^i = u^i{}_{min}$ (arc minimal)
- la commande est déterminée par une contrainte d'état active : $u^i = u^i{}_x$ (arc contraint)
- la commande n'est pas saturée. L'optimisation recherche la stationnarité de l'Hamiltonien, $\frac{\partial \mathcal{H}}{\partial u} = 0$ d'où le nom de *Sensitivity-seeking arc* : $u^i = u^i{}_{sens}$ (arc singulier)

Cette dépendance entre les contraintes sur les états et la commande est particulièrement problématique dans le cas des méthodes séquentielles, mais moins dans le cas de la méthode simultanées, car les états et les commandes sont discrétisées dans le même problème.
Par ailleurs, l'instant où se produit le passage d'un arc au suivant doit pouvoir être décelé par la méthode d'optimisation retenue. Il existe par exemple dans certains cas particuliers des contrôles optimaux dits bang-bang, où la commande passe directement d'un arc minimal à un

arc maximal. La grille de discrétisation utilisée doit alors permettre de localiser ces instants de commutation.
Pour cela, une première méthode est d'utiliser une adaptation de la grille temporelle. L'idée sous-jacente aux méthodes de raffinement de la grille de discrétisation est de détecter la séquence des arcs sur l'horizon, ainsi que les instants de commutation entre ces arcs.
On appliquera dans cette thèse un raisonnement similaire lors de la résolution du problème de commande optimale hybride : en effet, les trajectoires optimales des commandes binaires $\omega^{ji}(t)$ peuvent en être considérées une succession d'arcs minimaux ($\omega^{ji} = 0$) et maximaux ($\omega^{ji} = 1$). Cependant, on résout en réalité un problème relaxé où $\omega^{ji}(t) \in [0; 1]$, il peut exister des arcs où la commande n'est pas saturée. Le raffinement de la grille temporelle pourra ainsi être utilisé pour localiser les instants de commutation dans ce cas. Cette méthode sera décrite en 3.4.3.3.

3.2.2 Algorithme de tir simple direct (*Single Shooting*)

On présente maintenant la méthode d'optimisation directe la plus facile à implémenter : la méthode de tir simple direct. Dans cette méthode, on recherche les commandes permettant d'atteindre les conditions de stationnarité du Lagrangien rappelées en Annexe A pour le problème discrétisé. L'optimiseur définit une fonction de commande paramétrée $u(p, t)$, que le simulateur applique au système, ce qui permet d'évaluer la fonction objectif au temps final. La sensibilité de la fonction objectif aux paramètres p de la commande, qu'il conviendra d'évaluer numériquement ou par intégration de fonctions de sensibilité, est alors utilisée pour remettre à jour la fonction de commande.

3.2.3 Algorithme de tir multiple direct (*Direct Multiple Shooting*)

La méthode de tir simple direct présente un inconvénient : la valeur finale de la fonction objectif est d'autant plus sensible aux variations des paramètres (de la commande) que l'horizon de prédiction est long. Si le modèle est non linéaire, la fonction objectif peut avoir une dépendance non linéaire forte aux paramètres de la commande u, ce qui peut ralentir la convergence des algorithmes d'optimisation. La méthode de tir simple nécessite ainsi une très bonne initialisation. De plus, seuls les points initiaux et finaux sont accessibles pour l'optimiseur, et il n'est pas possible de considérer des points intermédiaires lors de l'initialisation. Pour réduire ces inconvénients, l'idée du tir multiple [35][36][37][13] est d'appliquer la méthode de tir simple sur chacun des éléments de la grille temporelle plutôt que sur tout l'horizon de contrôle. On introduit pour cela des variables d'état locales $s_k(t)$ sur chaque élément (correspondant à $x(t)$ sur l'intervalle $[t_{k-1}, t_k]$) comme montré en Figure 3.1. La dépendance de la fonction objectif à la commande sur chaque élément est ainsi moins non linéaire. Les valeurs s_k de l'état au début de chaque élément sont traitées comme de nouvelles inconnues et des équations de continuité sont ajoutées au problème d'optimisation afin de garantir la continuité de l'état : $s_k(t_k) = s_0^{k+1}$. On obtient ainsi un problème d'optimisation de taille plus importante (on a ajouté $n_e * n_x$ variables s_0^k et autant de contraintes de continuité), mais moins non linéaire. Un autre avantage de cette formulation est qu'elle rend le calcul parallélisable, les n_e simulations des trajectoires s_k pouvant être réalisées simultanément.

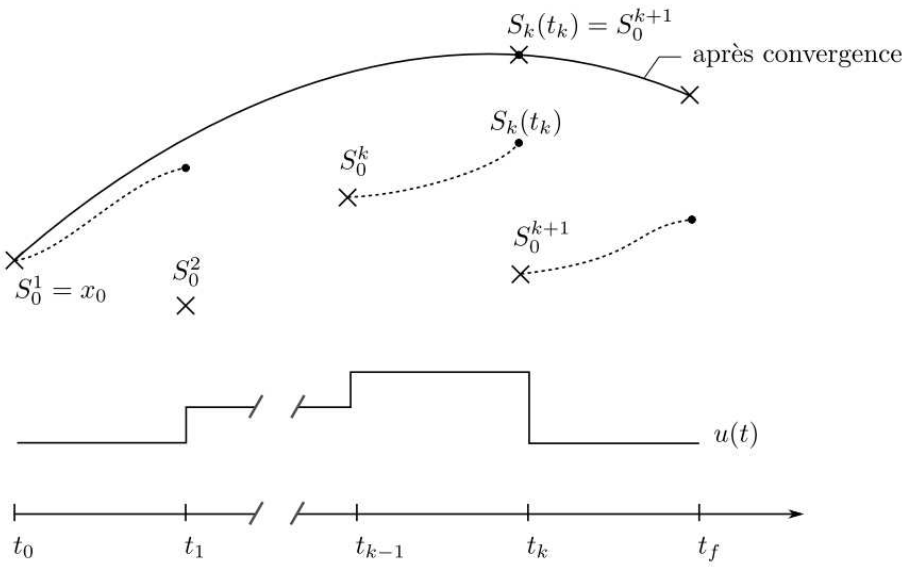


FIGURE 3.1 – Algorithme de tir direct multiple

3.2.4 Méthode de collocation

3.2.4.1 Notations utilisées pour les grilles temporelles

On présente maintenant plus en détail la méthode qui sert de support pour le développement de l'algorithme d'optimisation dynamique hybride développé dans cette thèse : la méthode de collocation. L'intérêt principal de la méthode de collocation est la prise en compte des contraintes sur l'état directement dans le problème d'optimisation, et c'est pour cela que nous l'avons retenue. Cette méthode fait appel à une discrétisation du problème de commande optimale décrite en 3.2.4.2. Dans un souci de clarté, on commence ici par redéfinir les diverses façons de représenter une même grille temporelle de N éléments. On utilise pour cela le même nom mais des polices de caractère différentes :

- Une grille peut être représentée par les instants qui limitent ses éléments. Ces grilles contiennent $N+1$ termes et sont notées avec une police calligraphiée :

$$\mathcal{T} = \{t_0; \dots; t_k; \dots t_{N+1} = t_f\}$$

- Une grille peut être représentée par la durée de ses éléments. Ces grilles contiennent N termes et sont notées en lettres capitales :

$$T = \{T_1; \dots; T_k; \dots T_N\}, \text{ avec } T_n = t_n - t_{n-1}$$

- Une grille peut être donnée comme un sous-ensemble d'une autre grille. Ce type de grille peut par exemple être utilisé si une commande constante par morceaux n'est pas autorisée à varier à tous les pas de temps. La grille $\mathcal{S}$ de $M \leq N$ termes et définie comme un sous-ensembles de $\mathcal{T}$ peut ainsi être représentée comme un vecteur contenant les indices des instants communs aux deux grilles. Ces grilles d'indices sont notées en gras :

$$\mathbf{s} = \{s_l, l \in \overline{1..M}\}, \quad \text{avec } s_l \in \overline{1..N+1} \tag{3.13}$$

Dans le cas où la grille $\mathcal{S}$ couvre l'horizon $[t_0, t_f]$, on aura ainsi $s_1 = 0$ et $s_M = N + 1$ car le dernier instant t_f a pour indice $N + 1$ dans la grille support $\mathcal{T}$.

3.2.4.2 Discrétisation du problème de commande optimale

Dans la méthode de collocation, en plus d'utiliser une paramétrisation du vecteur de commande, les trajectoires des états sont approchées sur chaque élément par un polynôme. Le schéma d'intégration numérique, qui permet d'obtenir les trajectoires d'état en fonction des évaluations de la dérivée des états en certains points, est ici inclus directement sous forme de contraintes dans le problème d'optimisation [38]. On peut ainsi montrer que les méthodes de collocation implémentent sous forme de contraintes dans le NLP un schéma d'intégration de Runge Kutta implicite [39]. L'appel à un simulateur pour fournir les trajectoires d'état n'est alors plus nécessaire. Le problème d'optimisation obtenu est un problème de plus grande taille mais présente une structure qui peut être exploitée pour résoudre le problème efficacement. Dans la méthode de collocation locale, l'horizon est scindé en n_e éléments. Les limites des éléments définissent la grille temporelle

$$\mathcal{T}_0 = \{t_0, t_1, \ldots, t_{n_e} = t_f\}$$

Chacun de ces éléments peut avoir une longueur $T_k = t_k - t_{k-1}$ différente, le changement en variables locales suivant est introduit afin de pouvoir se généraliser sur tous les éléments :

$$t^k(\tau) = t_{k-1} + T_k\tau, \quad \tau \in [0, 1], \forall k \in \overline{1..n_e}$$

Au sein de chaque élément, on définit n_c points de passage nommés points de collocation. On suppose que le nombre de points de collocation n_c ainsi que leurs positions τ_c au sein de chaque élément sont identiques dans tous les éléments. Les coordonnées en temps local des points d'interpolation sont $(\tau_c, x_{kc}, \dot{x}_{kc})$ pour les états et leurs dérivées, (τ_c, z_{kc}) pour les variables algébriques et (τ_c, u^c_{kc}) pour les entrées continues. Certaines entrées peuvent être maintenues constantes sur un élément, on les note u^b (entrées bloquées). Leurs coordonnées sont $(\tau_c, u^b_k, \forall c \in \overline{0..n_c})$. La méthode de collocation assure que la dynamique du système est respectée ponctuellement aux points de collocation par l'ajout de contraintes d'égalité reliant les variables que l'on vient d'introduire. La grille temporelle définissant les instants de collocation est définie comme :

$$\mathcal{T}_1 = \{t_{1,0} = t_0, t_{1,1}, t_{1,2}, \ldots, t_{1,n_c} \ldots, t_{n_e,n_c-1}, t_{n_e,n_c} = t_f\}$$

On illustre en Figure 3.2 les grilles $\mathcal{T}_0$, $\mathcal{T}_1$, ainsi qu'une grille $\mathcal{S} \subset \mathcal{T}_0$, que l'on représente aussi sous forme d'un tableau d'indices $\mathbf{s}$. La méthode de collocation est ainsi une transcription du système original sous forme d'un problème NLP de dimension finie. Le vecteur des variables d'optimisation est ici

$$V_{NLP} = \left(T_k, x_{kc}, \dot{x}_{kc}, z_{kc}, u^c_{kc}, u^b_k\right), k \in [1..n_e], c \in [0..n_c] \tag{3.14}$$

Cependant, on utilisera la plupart du temps une grille temporelle fixée par l'utilisateur (ce qui ne veut pas nécessairement dire que tous les éléments ont la même longueur). Dans ce cas,

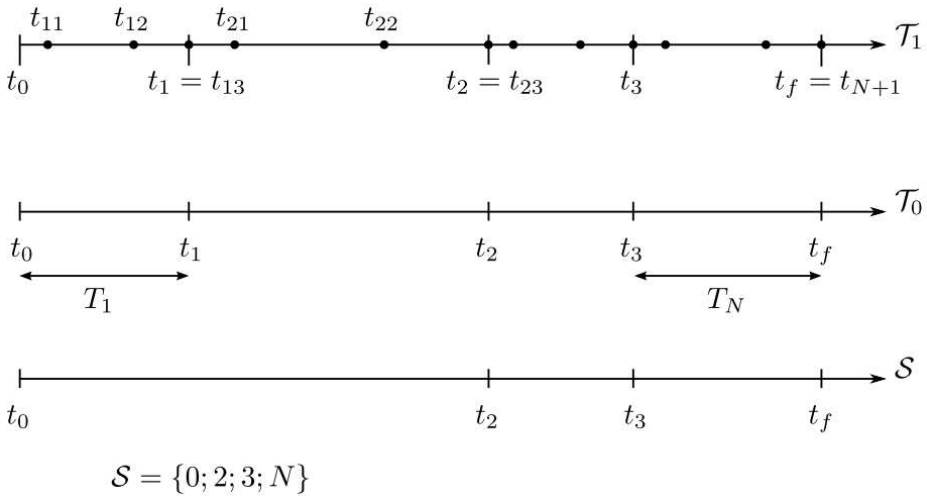


FIGURE 3.2 – Intervalles de temps utilisés dans les méthodes directes

les T_k seront retirés de V_{NLP}. On note V_{NLP}^f ce nouveau vecteur. Grâce à ces vecteurs, les équations de la dynamique aux points de collocation, les contraintes de continuité de l'état entre deux éléments consécutifs, ainsi que le schéma d'intégration numérique, peuvent alors facilement être intégrés comme des contraintes dans le problème NLP.

Choix des instants de collocation :
Les instants des points de collocation τ_c ne font pas partie des variables d'optimisation car en les choisissant d'une certaine manière (points de Gauss, de Radau ou de Lobatto), il est possible d'obtenir une précision optimale, ou certaines propriétés utiles. Afin d'expliquer les propriétés des points de Gauss, Radau et Lobatto, on rappelle que les schémas d'intégration numérique de Runge Kutta consistent à approcher la solution exacte de l'équation différentielle, donnée par :

$$x(t_k) = x(t_{k-1}) + \int_{t_{k-1}}^{t_k} \dot{x} dt \tag{3.15}$$

par la formule de quadrature suivante sur chaque élément

$$x(t_k) = x(t_{k-1}) + T_k \sum_{c=1}^{n_c} w_c dx_{kc}, \quad \text{avec } dx_{kc} = \dot{x}(t_{k-1} + T_k \tau_c) \tag{3.16}$$

Où dx_{kc} est défini par :
- $dx_{kc} = f(x_{kc}, [u_{kc}^c, u_k^b])$ dans le cas d'une ODE ou d'un DAE d'index au plus un
- la solution de la fonction implicite $F(x_{kc}, dx_{kc}, z_{kc}, [u_{kc}^c, u_k^b]) = 0$ dans le cas d'un DAE implicite

Dans la méthode de collocation, les trajectoires de l'état et de sa dérivée sont supposées être des polynômes $P(t) = x(t)$ et $P_d(t) = \dot{x}(t)$ qui vérifient $\frac{dP(t)}{dt} = P_d(t)$ aux points de collocation. Par la suite, on utilise la notation "tilde" pour le polynôme représentant la trajectoire d'une variable ($\tilde{x}$ pour les états, $\tilde{z}$ pour les variables algébriques, etc). Comme l'intégrale d'un polynôme peut être calculée de façon analytique, le choix des points de collocation et des

pondérations w_c peut ainsi être fait de telle sorte à maximiser la précision de la quadrature (3.16). L'intégrale sera même exacte si la trajectoire réelle $\dot{x}$ est un polynôme de degré au plus N, avec $N = 2 * n_c - 1$ pour les points de Radau et $N = 2 * n_c$ pour les points de Gauss.

On utilise par la suite les points dits de Radau car ces points ont la propriété intéressante de définir un instant de collocation à la fin de chaque élément : $\tau_{n_c} = 1$, ce qui permet d'écrire simplement les contraintes de continuité de l'état entre deux éléments consécutifs et les contraintes ou coûts terminaux.

Modélisation des dérivées $\dot{x}$:
Il est à noter que selon les implémentations, dx_{kc} peut être soit traité comme une variable d'optimisation, soit calculé (et dans ce cas éliminé du vecteur V_{NLP}). Les variables dx_{kc} peuvent en effet être calculées d'après le polynôme $\tilde{x}(t)$, en supposant que $\tilde{dx}$ n'est autre que la dérivée du polynôme $\tilde{x}$, ou aussi être définies comme $n_x * n_e * n_c$ variables du problème d'optimisation. Dans ce cas, il faudra également ajouter au problème les contraintes

$$dx_{kc} = \left.\frac{d\tilde{x}(t)}{dt}\right|_{t_{kc}}$$

L'intérêt de cette formulation est que des contraintes ou pénalisations sur $\dot{x}_{kc}$ peuvent être introduites dans le problème d'optimisation, pour limiter les variations brusques de l'état par exemple.

Représentation des variables continues :
Les polynômes utilisés pour approcher la trajectoire d'une variable ζ sur un élément k peuvent se représenter de plusieurs façons. On retiendra ici la formulation sous forme de polynômes de Lagrange : $\zeta_k^s(\tau)$ (3.17). Les polynômes représentés sous la forme de Lagrange présentent en effet un intérêt pratique : ils sont paramétrés par la valeur des points de passage $\zeta_{k,i}$, ce qui permet d'utiliser de façon directe les variables définies précédemment.

$$\begin{cases} \zeta_k^s(\tau) = & \displaystyle\sum_{c=s}^{n_{c,k}} \zeta_{kc} \ell_c^s(\tau) \\ \ell_c^s(\tau) = & \displaystyle\prod_{p \in [s..n_c] \backslash c} \frac{\tau - \tau_p}{\tau_c - \tau_p} \end{cases} \tag{3.17}$$

On justifie maintenant l'utilisation de l'indice de départ s dans l'expression (3.17), qui correspond à l'indice du premier point utilisé dans les polynômes de Lagrange. On a vu que l'intégrale de $\dot{x}$ était exacte avec les points de Radau pour des polynômes jusqu'à un degré $N = 2 * n_c - 1$. Comme le polynôme P_d représentant la trajectoire de $\dot{x}$ est de degré $n_c - 1$, le polynôme P représentant la trajectoire de x peut être choisi de degré n_c sans perte de précision [38], ce qui autorise l'ajout d'un point d'interpolation supplémentaire pour celui-ci. Le choix est fait ici d'ajouter ce point d'interpolation au début de chaque élément, aux instants t_{k0} (correspondant à $\tau_0 = 0$ en variables locales). Il sera ainsi aisé d'écrire la continuité des variables d'état x entre deux éléments consécutifs. Dans (3.17), on a ainsi $s = 0$ pour les états, mais $s = 1$ pour les dérivées dx. Comme par ailleurs on n'impose pas la continuité entre deux éléments pour les commandes u et les variables algébriques z, il n'est pas nécessaire d'avoir un point d'interpolation à $\tau = 0$; on choisit alors $s = 1$ pour ces variables.

Remarque La continuité des trajectoires entre deux éléments ne s'applique qu'aux états. Les variables algébriques z, les commandes u et les valeurs des dérivées dx peuvent être discontinues à la frontière de deux éléments. Le polynôme représentant la trajectoire de x doit ainsi avoir un degré de liberté supplémentaire pour garantir la continuité. C'est pour cela que dans la formule 3.18, un point à $\tau_0 = 0$ est ajouté pour les états, et que la somme impliquée dans le polynôme de Lagrange (3.17) commence à $s = 0$ pour x et à $s = 1$ pour u et z et dx.

Pour simplifier la notation, seul un coût terminal $J = \tilde{\Phi}(x(t_f))$ est considéré ici. On rappelle ici qu'il est toujours possible de ramener un critère de Bolza sous cette forme comme évoqué en 3.1.3.

$$
\begin{aligned}
\min_V J &= \Phi(x_{n_e,nc_{n_e}}) \quad \text{s.c.} \\
0 &= g(x_{kc}, z_{kc}, [u^c_{kc}, u^b_k], \tau_c) && \text{Equations algébriques} \\
dx_{kc} &= \frac{1}{T_k} \sum_{c=0}^{n_c} x_{kc} \dot{\ell}^0_c(\tau_c) && \text{Equations de collocation} \\
dx_{kc} &= f(x_{kc}, z_{kc}, [u^c_{kc}, u^b_k], \tau_c) && \text{Equations dynamiques} \\
x_{k,0} &= x_{k-1,nc_k} && \text{Equations de continuité} \\
0 &\geq h_I(x_{kc}, z_{kc}, u_{kc}) && \text{Contraintes inégalité} \\
\begin{pmatrix} \underline{x} \\ \underline{dx} \\ \underline{z} \\ \underline{u} \end{pmatrix} &\leq \begin{pmatrix} x_{kc} \\ dx_{kc} \\ z_{kc} \\ u_{kc} \end{pmatrix} \leq \begin{pmatrix} \overline{x} \\ \overline{dx} \\ \overline{z} \\ \overline{u} \end{pmatrix} && \text{Contraintes de bornes}
\end{aligned}
\tag{3.18}
$$

Une implémentation de la méthode de collocation est présente dans le logiciel d'optimisation JModelica [40], et utilise le logiciel CasADi pour l'obtention de dérivées symboliques [24]. Les développements proposés ici ont été réalisés avec JModelica et sont des extensions de cette implémentation.

3.3 Méthodes pour les problèmes MINLP

Le problème d'optimisation dynamique hybride (H-OCP) fait intervenir des variables continues et des variables logiques. Dans la partie précédente, on a montré comment un problème d'optimisation dynamique peut être transformé en un programme non linéaire à l'aide des méthodes directes (tir, tir multiple, collocation). Le couplage de ces deux aspects résulte en un problème de dimension finie mêlant variables continues $x \in \mathbb{R}^{n_x}$ et logiques $y \in \{0;1\}^{n_y}$ faisant partie de la classe *Mixed Integer Nonlinear Programming*, abrégé MINLP (on fait la remarque que dans cette partie x et y font référence aux variables continues et logiques et non plus aux états et sorties définies précédemment). Dans le problème considéré, les variables y_i sont utilisées pour représenter le mode courant ou l'état d'activation d'un composant. Certaines contraintes ne sont ainsi ajoutées au problème que lorsque la variable y_i correspondante vaut 1. On s'intéresse donc au problème MINLP suivant :

$$\begin{array}{lll} \min\limits_{x,y} & J(x,y) & \\ \text{s.c.} & & G(x,y) \leq 0 \\ & & H(x,y) = 0 \\ & & \veebar_{i \in \overline{1..n_y}} \begin{pmatrix} y^i \\ G^i(x,y) \leq 0 \\ H^i(x,y) = 0 \end{pmatrix} \end{array} \tag{3.19}$$

La propriété "*Un seul mode actif à chaque instant*" que l'on a appelée disjonction est notée $\veebar$ et peut être modélisée par l'utilisation de la propriété SOS1 (empruntée à la programmation linéaire en nombres entiers) :

Définition (Propriété SOS1 *Special Ordered Set 1*, SOS1) Les variables logiques $y_1, \ldots, y_{n_y}$ vérifient la propriété SOS1 si

$$\sum_{i=1}^{n_y} y^i = 1, \quad y^i \in \{0;1\} \quad \forall i \in \overline{1..n_y} \tag{3.20}$$

On présente ici certaines des méthodes pouvant être employées pour résoudre les problèmes MINLP du type (3.19). La plupart d'entre elles s'appuient sur des problèmes relaxés :

Définition (Problème relaxé) On appelle problème relaxé un problème qui admet les solutions du problème initial comme un cas particulier. Dans le cas des problèmes MINLP (3.19), des problèmes relaxés peuvent être obtenus en remplaçant des contraintes $y^i \in \{0;1\}$ par $y^i \in [0,1]$

3.3.1 Décomposition continu/logique, algorithmes Branch & Bound

L'algorithme Branch & Bound est inspiré de la PLNE, et peut sous certaines conditions être étendu au cas non linéaire. On suppose pour l'instant que le problème est non linéaire mais convexe (la convexité était garantie dans le cas linéaire), puis on montre les modifications nécessaires dans le cas non convexe. Dans la version originale de l'algorithme Branch & Bound, la partie logique du problème est représentée par un arbre de décision 2-aire (c'est à dire composé de nœuds ayant chacun deux branches filles), où chaque nœud représente une variable binaire y^i et ses deux branches filles le cas $y^i = 0$ et $y^i = 1$. Il y a donc au plus $N = 2^{n_\omega}$ nœuds dans l'arbre (ce nombre peut être restreint par les contraintes logiques). On donne l'exemple d'un système pouvant être dans une des trois configurations montrées en Figure 3.3, chacune de ces configurations étant définie sur $\mathbb{R}^2$ par $G_i(x_1, x_2) \leq 0$. La construction de l'arbre correspondant est montré sur la droite. Comme la disjonction contraint les variables logiques a être mutuellement exclusives, cet arbre n'a que 3 nœuds terminaux (et non pas $2^3 = 8$).

Des problèmes relaxés (NLP) peuvent alors être construits en parcourant les branches de l'arbre jusqu'à un nœud k, c'est-à-dire en fixant certaines des variables y_f soit à 0, soit à 1 et en utilisant la relaxation $[0,1]$ pour les autres. Par abus de notation, les variables y^i de y_f fixées à 1 sont notées y^i, celles fixées à 0 $\neg y^i$. On note $J^{bin,*}$ la valeur optimale de la fonction objectif pour le problème original et $J^{rel,*}_{y_f}$ la valeur de la fonction objectif pour le problème relaxé courant. La valeur de l'optimum lorsque toutes les variables binaires sont fixées à 0 ou 1, correspondant à un vecteur $Y_f \in \{0;1\}^N$, c'est-à-dire lorsqu'on a atteint un

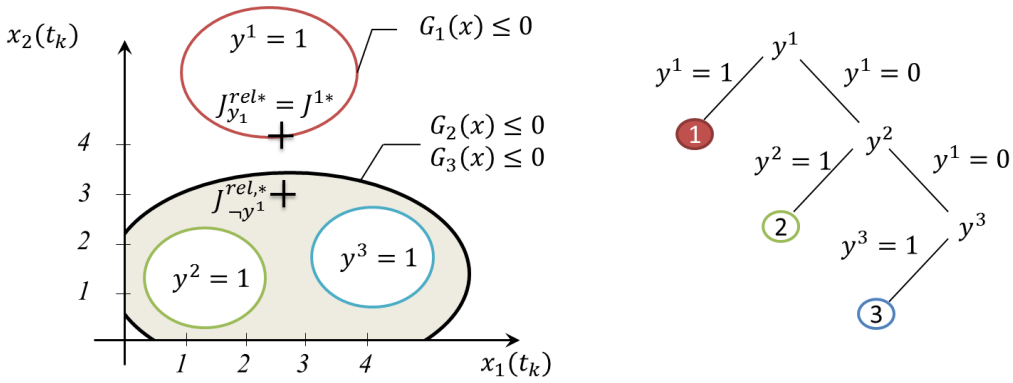


FIGURE 3.3 – Algorithme de Branch & Bound sur une disjonction de 3 modes

nœud terminal k dans l'arbre de décision, est notée $J^{k,*}$. On a supposé pour l'instant que les contraintes du problème relaxé sont convexes, ce qui garantit que ces minima sont globaux. Dans l'algorithme Branch & Bound, on recherche des bornes inférieures et supérieures J_{min} et J_{max} pour $J^{bin,*}$. Les propriétés suivantes sont utilisées, en conservant à chaque fois en mémoire les meilleures bornes J^*_{min} et J^*_{max} trouvées jusqu'alors :

- $J^{bin,*}_k$ est une solution faisable mais pas forcément optimale pour le problème initial car elle correspond à un choix arbitraire des variables logiques. Pour cette raison, $J^{bin,*} \leq J^{bin,*}_k$ donne une borne supérieure $J_{max} = J^{k,*}$ pour la fonction objectif.
- Les problèmes relaxés vérifient $J^{rel,*} \leq J^{bin,*}$. $J^{rel,*}$ fournit ainsi une borne inférieure J_{min} pour la fonction objectif.
- Si le problème relaxé fournit une solution binaire pour les variables relaxées y_r, alors cette solution fournit une borne supérieure $J_{max} = J^{rel,*}$.
- Si le problème relaxé est infaisable, alors le choix des variables y_f fixées n'est pas faisable, et toutes les branches sous ce sous-arbre peuvent être éliminées.
- Si le problème relaxé sur un sous arbre donne une valeur de l'objectif moins bonne que la meilleure borne supérieure ($J^{rel,*} > J^*_{max}$), alors le sous-arbre peut être écarté des recherches.

Dans la version de l'algorithme Branch & Bound proposée par [41] le problème entièrement relaxé est résolu, puis l'on fixe de façon récursive des noeuds (branch). Pour chaque nouveau noeud y_f fixé, deux problèmes relaxés sont créés correspondant à la branche 0 et 1. Cependant, comme les problèmes relaxés doivent fournir des bornes inférieures de l'objectif, il est important que ceux-ci soient des minima globaux. Dans le cas non linéaire, cela peut être fait en convexifiant le problème relaxé :

- On surestime le domaine des contraintes par un ensemble convexe (ensemble gris sur la Figure 3.3)
- On crée un sous-estimateur convexe pour la fonction objectif

Le parcours de l'arbre se poursuit ensuite en éliminant si besoin des sous-arbres comme décrit précédemment, jusqu'à atteindre un nœud terminal, ou obtenir une solution relaxée entière. Ces deux cas fournissent en effet une borne supérieure J_{max}. L'algorithme s'arrête lorsque la différence entre J^*_{max} et J^*_{min} est sous une tolérance définie par l'utilisateur.

Des améliorations du Branch & Bound dans le cas non linéaire ont été proposées, on citera notamment l'algorithme *Logic Based Outer Approximation* initialement proposé par [42] et raffiné dans [30] puis [43], et dont l'application première est l'optimisation structurelle de procédés. Dans cette méthode, une superstructure du procédé permettant d'obtenir toutes les configurations possibles est créée. Le modèle est donc potentiellement de grande taille. L'idée principale de l'algorithme est alors de réduire la dimension des problèmes relaxés en ne conservant que les composants correspondant au choix des variables logiques données par le problème maître (c'est de là que vient l'appellation *Logic Based*). Les problèmes relaxés sont ainsi des problèmes réduits. Une méthode nommée *Outer Approximation* (OA) est alors employée pour convexifier ces problèmes réduits.

3.3.2 Reformulations continues

Une autre approche consiste à reformuler les contraintes logiques à l'aide de variables continues. Dans cette approche, les variables binaires sont relaxées, le plus souvent par $y \in [0,1]$. L'utilisation de contraintes spécifiques ou de pénalisation de la fonction objectif permet par la suite de respecter le caractère binaire des solutions. Ce type d'approche permet d'éviter l'aspect combinatoire des précédentes méthodes et la résolution d'un grand nombre de sous-problèmes mais peut engendrer des difficultés numériques lors de la résolution du problème. Il convient de faire la distinction entre les deux types de contraintes suivants qui nécessitent un traitement différent :

- La convexification des contraintes spécifiques aux modes, qui donne une surestimation de l'ensemble admissible
- La reformulation continue des contraintes disjonctives de type SOS1

3.3.2.1 Reformulation continue des contraintes spécifiques aux modes

La modélisation hybride peut être utilisée pour modéliser des contraintes spécifiques à chaque mode. Certaines contraintes $G^i(x) \leq 0$ ne seront ainsi activées que lorsque la variable logique y^i correspondante sera active. On peut alors faire appel à une des reformulations continues suivantes, qui fournissent une définition exacte des contraintes spécifiques au mode actif dans les cas limites où toutes les variables sont binaires.

- Par multiplication binaire :

$$y^i G^i(x) \leq 0, \quad \forall i \in \overline{1..n_y}, \quad \sum_{l=1}^{n_y} y_l = 1 \tag{3.21}$$

- Par la méthode *Big-M* :

$$G^i(x) \leq M \sum_{l \neq i} y^l, \quad \sum_{l=1}^{n_y} y_l = 1 \tag{3.22}$$

M est ici une constante suffisamment grande pour que $G^i(x) \leq M$ soit toujours vérifiée.

– Par l'enveloppe convexe du domaine des contraintes [31]. On introduit ici n_ω vecteurs $\nu^i \geq 0$ et θ tels que :

$$\begin{aligned} \sum_{i=1}^{n_\omega} y^i = 1, \quad \theta = \sum_{i=1}^{n_\omega} \nu^i \\ y^i G^i(\frac{\nu^i}{y^i}, t) \leq 0 \\ 0 \leq \nu^i \leq y^i \nu^i_{max} \end{aligned} \tag{3.23}$$

Ces méthodes peuvent également s'appliquer aux contraintes égalité, dans le cas de la méthode Big-M en reformulant les contraintes égalité comme deux inégalités. En pratique, on peut être amenés à utiliser des relaxations de ces formulations : dans ce cas, les y^i ne sont plus des variables binaires mais définies par $y^i \in [0, 1]$. Dans ce cas relaxé, toutes les formulations ne sont alors plus équivalentes : l'enveloppe convexe donne la relaxation la plus resserrée de l'ensemble admissible mais est plus compliquée à mettre en pratique que la méthode Big-M, notamment car les contraintes doivent être reformulées avec les nouvelles variables ν^i. Par ailleurs, une régularisation doit être utilisée pour la troisième contrainte de (3.23) (division par 0). Enfin, la multiplication binaire doit être elle aussi régularisée pour pouvoir être traitée numériquement, la façon la plus simple étant d'introduire une variable $\epsilon > 0$ (que l'on peut fixer ou minimiser) :

$$y^i G^i(x) \leq \epsilon$$

Ces relaxations seront utilisées pour reformuler les contraintes spécifiques aux modes du problème de commande optimale hybride H-OCP et seront détaillées pour ce cas spécifique en partie 3.4.1.4.

3.3.2.2 Traitement des disjonctions par l'ajout de contraintes continues

On a vu pour l'instant la reformulation des contraintes continues spécifiques à chaque mode. Cependant, les problèmes hybrides peuvent aussi faire intervenir des contraintes sur les variables logiques y^i elles mêmes. Dans notre cas, on supposera que les disjonctions sont les seules contraintes de ce type.
Afin de reformuler de façon continue les contraintes de disjonction, on utilise des fonctions auxiliaires Φ qui ont la particularité de s'annuler pour tous les points admissibles. En remplaçant les contraintes logiques par des contraintes du type $\Phi(y) = 0$ le problème peut alors être résolu sur les variables relaxées. Cependant, le domaine admissible décrit par une disjonction est disjoint et avec un intérieur vide. Or ce type de contrainte, connu sous le nom de contraintes de complémentarité pose typiquement des problèmes aux algorithmes d'optimisation comme nous allons le voir ci-dessous.

Définition Un problème de complémentarité (MPCC, *Mathematical Problem with Complementarity Constraints*) contenant m_c contraintes de complémentarité est décrit par le jeu de contraintes suivant :

$$\begin{aligned} C_1^i(x) \geq 0 \\ C_2^i(x) \geq 0 \\ C_1^i(x) C_2^i(x) = 0, \quad \forall i \in [1..m_c] \end{aligned}$$

On l'abrègera parfois en $C_1^i(x) \perp C_2^i$.

L'approche la plus directe pour reformuler une disjonction de façon continue est sous forme du problème MPCC suivant :

$$\sum_{i \in D_j} y^{ji} = 1 \tag{3.24a}$$

$$y^{ji} y^{j,l \neq i} = 0 \qquad \forall i \in D_j \tag{3.24b}$$

$$0 \leq y^{i,j} \tag{3.24c}$$

Cependant, cette formulation est peu souhaitable car elle enfreint la plupart des critères de qualification des contraintes en chaque point admissible comme détaillé ci-dessous. Or la qualification des contraintes est un pré-requis pour le succès des algorithmes que nous allons utiliser.

Qualification des contraintes pour les problèmes MPCC :
Le critère LICQ défini en Annexe A est un des critères de qualification des contraintes qui garantit que les conditions KKT sont des conditions nécessaires d'optimalité. C'est un critère assez restrictif mais facile à tester (test sur le rang du gradient des contraintes actives). On montre ici que la formulation (3.24) viole le critère LICQ en tout point admissible. Tout d'abord, la condition SOS1 garantit qu'au moins une des conditions $y^1 > 0$ ou $y^2 > 0$ est vérifiée. Supposons que ce soit y^1. En raison de la complémentarité, les contraintes actives incluent $y^2 = 0$ et $y^1 y^2 = 0$. Les lignes correspondantes dans le gradient des contraintes actives sont ainsi linéairement dépendantes :

$$\frac{\partial}{\partial (y^1, y^2, X)} \begin{pmatrix} y^1 y^2 \\ y^2 \end{pmatrix} = \begin{pmatrix} y^2 = 0 & y^1 & \dots & 0 \\ 0 & 1 & \dots & 0 \end{pmatrix}$$

Le même problème s'observe lorsque l'on utilise la méthode de multiplication binaire décrite précédemment : le critère LICQ est mis en défaut en chaque point qui vérifie $y^i = 0, h^m_{I,i}(x) = 0$ [44]. De plus, cette formulation est problématique pour les algorithmes de points intérieurs tels qu'IPOPT [45], qui nécessitent d'être initialisés avec une solution admissible strictement à l'intérieur du domaine des contraintes. Or ici toute solution admissible n'est pas intérieure au domaine des contraintes : en effet, une solution admissible implique que pour au moins un des i, $y^i = 0$. On montre également que le critère MFCQ (présenté en Annexe A), moins restrictif, est également enfreint en tout point admissible [46]. La reformulation continue des contraintes de complémentarité sous la forme (3.24) semble ainsi inappropriée pour ce type d'algorithme qui est malheureusement celui que nous allons utiliser par la suite. Une première solution pour contrer ce problème est d'utiliser des algorithmes adaptés à ce type de problèmes. On peut notamment citer IPOPT-C [47], une version modifiée d'IPOPT, qui utilise une régularisation des contraintes de complémentarité : les contraintes de complémentarité sont ainsi relaxées à l'aide d'un paramètre δ et durcies à chaque itération (μ est ici le paramètre de la fonction barrière de l'algorithme) :

$$y^i y^l = 0 \quad \text{est relaxé par } y^i y^l \leq \delta\mu, \quad \mu \text{ le paramètre de barrière, } \mu \to 0 \text{ et } \delta > 0$$

Des variables d'écart sont ensuite ajoutées sur cette nouvelle formulation :

$$y^i y^l + e^{il} = \delta\mu, \quad e^{il} \geq 0$$

Cela permet de traiter la contrainte de complémentarité relaxée sans les problèmes précités, et à la limite $\mu = 0$, la condition de complémentarité est respectée.

Traitement de la complémentarité par les fonctions NCP
Une alternative à l'utilisation des contraintes (3.24) utilise les fonctions dites NCP (*Nonlinear Complementarity Problem*), qui sont des applications $\phi_{NCP} : \mathbb{R}^2 \to \mathbb{R}$ possédant la propriété suivante :

$$\phi_{NCP}(x, y) = 0 \Leftrightarrow x \geq 0, y \geq 0, xy = 0 \tag{3.25}$$

Une fonction NCP connue est la fonction de Fischer Burmeister définie par

$$\phi_{FB} = (x + y) - \sqrt{x^2 + y^2} \tag{3.26}$$

Cette fonction a certaines propriétés intéressantes en optimisation. Elle est notamment différentiable partout sauf en $(0, 0)$ (qui n'est par ailleurs pas un point admissible dans le cas d'une disjonction)

$$\nabla\phi_{FB} = \begin{pmatrix} 1 - \frac{2x}{\sqrt{x^2+y^2}} \\ 1 - \frac{2y}{\sqrt{x^2+y^2}} \end{pmatrix} \tag{3.27}$$

Une disjonction $\underline{\vee}_{i \in \overline{1..n_\omega}} \omega^i$ peut ainsi se reformuler en utilisant des fonctions NCP comme suggéré par [48] :

$$\Phi_{FB}\left(\omega^i, \sum_{j \neq i} \omega^j\right) = 0 \tag{3.28}$$

$$\sum_{k=1}^{n_\omega} \omega^k \geq 1 \tag{3.29}$$

$$\omega^k \geq 0 \qquad \forall k \in \overline{1..n_\omega} \tag{3.30}$$

La présence de variables logiques est ainsi remplacée par l'ajout de contraintes égalité non linéaires, qui ajoutent des non-convexités au problème. Ces non convexités sont souvent problématiques lors de la résolution du problème, même si cet inconvénient peut être tempéré en utilisant une relaxation pour la contrainte 3.28 :$\Phi_{FB}\left(\omega^i, \sum_{j \neq i} \omega^j\right) \leq \mu^{(n)}$, avec $\mu^{(n)}$ une constante que l'on fait décroître à chaque itération n de l'optimisation.
Une autre alternative est d'utiliser des fonctions NCP régularisées comme celles décrites dans [46]. L'utilisation de la fonction suivante garantit par exemple une convergence vers des points M-stationnaires (définis en Annexe A) :

$$\Phi_K(a, b; t) = \begin{cases} (a - t)(b - t) & , a + b \geq 2t \\ -\frac{1}{2}((a - t)^2 + (b - t)^2) & , \text{sinon} \end{cases}$$

Où $t \geq 0$ est un paramètre que l'on fait tendre progressivement vers 0. Cette fonction peut ainsi être utilisée pour représenter une disjonction en ajoutant au problème continu la contrainte :

$$\Phi_K\left(\omega^i, \sum_{j \neq i} \omega^j; t\right) = 0, \quad \forall i \in \overline{1..n_\omega}$$

3.3.2.3 Traitement des disjonctions par pénalisation

Afin d'éviter les difficultés induites par la complémentarité, une autre approche est de relaxer la contrainte de complémentarité mais de pénaliser toute déviation [49],[50]. Le problème d'optimisation devient :

$$\min_{x,\epsilon} \quad J(x) + \rho\epsilon \tag{3.31}$$

$$t.q. G_1^i(x)G_2^i(x) \leq \epsilon \tag{3.32}$$

$$\epsilon > 0, \quad G_1(x) \geq 0, \quad G_2(x) \geq 0, \quad \forall i \in [1..m_c] \tag{3.33}$$

La convergence est alors garantie pour ρ suffisamment grand. Comme tous les $G_1^i(x)$ et $G_2(x)^i$ sont positifs, il est également possible de remplacer (3.32) par la formulation aggrégée suivante (produit scalaire) :

$$G_1(x)G_2(x) \leq \epsilon \tag{3.34}$$

Il est également possible de définir ϵ comme une constante et non comme une variable d'optimisation. La méthode la plus courante est de résoudre le problème (3.31-3.33) de manière itérative en réduisant progressivement la valeur de ϵ.

Le domaine décrit est non convexe mais n'est plus disjoint, ce qui autorise à nouveau l'utilisation d'algorithmes de points intérieurs.

Dans la pratique, ces méthodes se sont révélées très sensibles à l'initialisation. Pour cette raison, elles n'ont pas été retenues dans la méthode d'optimisation développée. Celle-ci utilise néanmoins certains des aspects présentés dans cette partie :

- On forme un problème relaxé qui utilise les convexifications décrites en 3.3.2.1.
- Branch & Bound : la résolution alterne entre la résolution du problème relaxé et d'un problème où l'on fixe les variables logiques jusqu'à ce que l'écart entre les deux solutions passe en dessous d'un seuil.

Le respect des contraintes de disjonction est en revanche traité d'une manière qui n'a pas encore été abordée. Dans la solution proposée, les trajectoires des commandes binaires sont construites d'après le résultat du problème relaxé à l'aide d'une méthode spécifique que l'on présente en 3.4.2.

3.4 Développement d'une méthode d'optimisation dynamique hybride

Dans cette partie, on présente une extension de la méthode de collocation au problème d'optimisation dynamique pour les systèmes hybrides avec transitions explicites décrit en (H-OCP). La solution proposée combine les méthodes d'optimisation dynamique et les méthodes MINLP présentées dans les deux parties précédentes. Le problème de commande optimale initial de dimension infinie est tout d'abord discrétisé dans le temps, et en particulier les trajectoires des entrées logiques $\omega(t)$. Le problème discrétisé ainsi obtenu est un problème MINLP, mais qui a la particularité de représenter une dimension temporelle. Cette particularité peut être exploitée à l'aide de techniques de résolution spécialisée. On peut ainsi identifier trois grands types de méthodes pour le traitement des aspects logiques dans le problème de commande optimale.

La première approche utilise une décomposition continu/logique du problème. La résolution fait appel à deux problèmes hiérarchisés : un problème "maître" optimise les variables logiques, puis des sous-problèmes de commande optimale continus sont résolus avec la trajectoire des variables logiques fixées. C'est une approche similaire à celle utilisée dans les algorithmes Branch & Bound ou Outer Approximation. Ici, les variables logiques du problème représentent l'état des composants sur chaque élément.La difficulté de l'utilisation de ce type de méthodes pour l'optimisation dynamique est liée à sa nature combinatoire : le nombre de séquences de commutations possible croit exponentiellement avec le nombre d'éléments de la grille de discrétisation. Pour pallier cet inconvénient [31] suggère de ne considérer qu'un nombre restreint de séquences probables en utilisant la programmation disjonctive [51][52]. Cependant cette phase de modélisation est très spécifique au problème considéré.

Pour éviter cet aspect combinatoire, une seconde approche passe par la reformulation continue des aspects disjonctifs comme évoqué en 3.3.2.2. L'ajout de contraintes continues au problème relaxé remplace les contraintes logiques du problème initial, et notamment les disjonctions qui déterminent le mode courant. Une approche par pénalisation, où l'on durcit progressivement la pénalité sur le caractère non binaire des variables logiques peut aussi être envisagée, comme mentionné en 3.3.2.3. Le problème à résoudre est ainsi ramené à un simple NLP, mais cette formulation amène les mêmes problèmes numériques que ceux évoqués en 3.3.2.2 : elle ajoute des non linéarités fortes dans les contraintes ou la fonction objectif qui rendent le problème très sensible à l'initialisation des variables. Cette approche a été expérimentée mais donnait des résultats non satisfaisants même pour des cas simples et n'a pas été retenue par la suite.
La troisième est une méthode qui prend en compte la dimension temporelle de ce problème MINLP. Son principe est de construire les trajectoires des variables logiques d'après les trajectoires relaxées de ces mêmes variables. Cette dernière approche a été retenue dans la thèse pour ses performances numériques et les garanties théoriques de la méthode employée. Son fondement s'appuie sur des résultats initialement présentés par [53] pour les systèmes commutés à deux modes : soit les trajectoires des variables logiques obtenues sur le problème relaxé sont déjà des solutions du problème hybride (solutions de type *Bang Bang* où les commandes relaxées "sautent" entre 0 et 1), ou soit les variables relaxées prennent une valeur intermédiaire, mais il est dans ce cas possible de construire sur cette grille temporelle une trajectoire binaire sous-optimale telle que la différence entre les deux trajectoires résultantes soit bornée. La méthode utilisée dans cette thèse introduite par [54] et présentée en 3.4.2 permet de minimiser cette borne.

On formule maintenant les exigences pour la méthode de résolution numérique du problème de commande optimale hybride auxquelles la méthode développée devra répondre.

- La méthode doit fournir des trajectoires binaires pour les entrées logiques
- La solution du problème discrétisé doit être admissible pour le problème réel en temps continu. En particulier, les contraintes doivent être respectées en dehors des instants de discrétisation. Cette exigence implique que la granularité de la discrétisation doit être assez fine pour obtenir la précision requise pour les trajectoires calculées (par rapport aux trajectoires réelles).
- Les instants de commutation optimaux doivent pouvoir être localisés. Il est important de noter à ce stade que la discrétisation temporelle choisie influence le nombre et les

instants de commutations possibles pour le système. La grille temporelle doit ainsi pouvoir être adaptée par la méthode d'optimisation.

Afin de mieux appréhender les concepts développés dans la suite de ce chapitre, on présente à titre introductif les résultats obtenus par la méthode itérative d'optimisation dynamique hybride sur un exemple simple : l'oscillateur de Van Der Pol (VDP), régi par les équations suivantes :

$$\begin{aligned} \dot{x}_1 &= (1 - x_2^2)x_1 - x_2 + u \\ \dot{x}_2 &= x_1 \\ \dot{J} &= e^{p_1}(x_1^2 + x_2^2 + u^2) \\ x_1(t_0) = 0, \quad x_2(t_0) &= 1, \quad J(t_0) = 0 \\ 0 \leq u &\leq 0,75 \end{aligned}$$

Avec x_1, x_2 les états du système, u la commande, p_1 un paramètre et $J(t_f)$ la fonction objectif visant à amener le système en $(x_s, u_s) = ((0,0),0)$. Les deux graphes sur la Figure 3.4 montrent de haut en bas les trajectoires des états x_1 et x_2, de la commande u et de la fonction objectif J à la première et à la dernière itération (3) de la méthode proposée. La grille temporelle courante est matérialisée par les traits verticaux. L'optimisation est initialement réalisée sur une grille temporelle à grosse maille (figure du haut). Les trajectoires des sorties prédites par la méhtode de collocation sont en trait pointillé. Le système est ensuite simulé avec les trajectoires de commande optimales données par la méthode de collocation (trait plein pour les sorties). On constate qu'en raison de la discrétisation trop grossière utilisée, les trajectoires simulées du système s'écartent de celles prédites par l'optimisation. Afin d'obtenir une bonne cohérence entre ces deux trajectoires, la grille est alors raffinée aux endroits où la précision de la méthode de collocation est insuffisante. L'erreur est ici calculée d'après une méthode qui sera présentée au paragraphe 3.4.3.1 On constate sur le deuxième graphe de la Figure 3.4 que le raffinement de la grille d'optimisation a permis d'améliorer la stabilisation de l'oscillateur tout en conservant un nombre de pas de temps adapté (ce qui permet de restreindre la taille du problème discrétisé sans altérer la précision des solutions).

Dans le cas des systèmes hybrides, la grille est également raffinée dans un autre but : celui de détecter les instants de commutation. La méthode d'optimisation proposée alterne entre la résolution de problèmes relaxés où $\omega(t) \in [0,1]$ et de problèmes où les trajectoires $\omega(t)$ sont fixées et binaires ($\omega(t) \in \{0;1\}$), en raffinant à chaque étape la grille temporelle. Pour illustrer la méthode d'optimisation sur un système hybride, on introduit une contrainte hybride dans l'oscillateur de Van der Pol : on considère que l'entrée u présente désormais une bande morte et ne peut pas être activée en dessous d'un seuil $u_m = 0,1$. L'activation de u est ainsi modélisée par l'ajout d'une variable logique ON telle que :

$$\begin{cases} u \in [0,1;0,75] & , ON = 1 \\ u = 0 & , ON = 0 \end{cases}$$

Les résultats de la méthode après 8 itérations sont présentés dans la Figure 3.5. La trajectoire de la variable binaire ON est insérée sur le quatrième graphique. Sur le graphiques 4, la trajectoire de la commande logique ON (trait plein bleu) est construite d'après la trajectoire relaxée de cette variable (trait pointillé vert). Sur les autres graphiques, les trajectoires en trait pointillé vert représentent l'optimisation sur le problème relaxé et celles en trait bleu continu l'optimisation sur le problème où l'on a fixé la trajectoire de la variable ON. On

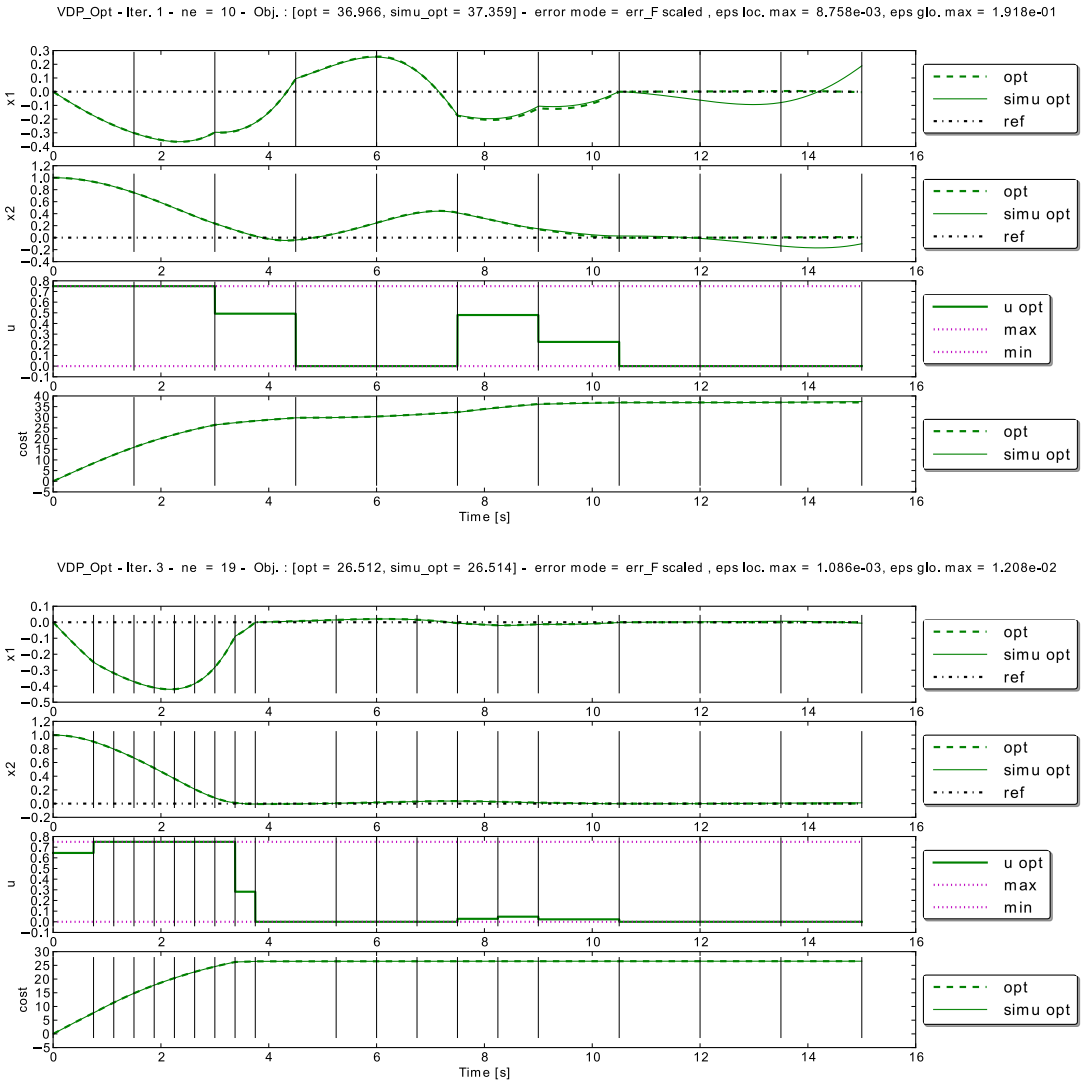


FIGURE 3.4 – Optimisation dynamique de l'oscillateur de VDP - Itération 1 et 3

constate que la grille n'est pas raffinée de manière homogène. L'entrée u est activée au début de l'horizon (de 0 à 4s), puis brièvement après 8s. Autour de cet instant, on remarque que la trajectoire relaxée pour ON n'est pas à 0 ni 1. Le raffinement de la grille permet de localiser précisément l'instant d'activation de l'entrée u, et d'obtenir une fonction objectif similaire sur le problème hybride (24.510) et le problème relaxé (24.468).
Comme on a pu le constater dans ces deux exemples introductifs, la méthode développée fait intervenir les aspects suivants :

- La mise en forme des modèles hybrides et la reformulation des contraintes hybrides selon un formalisme particulier. Cette partie sera détaillée en 3.4.1.
- La génération de trajectoires binaires d'après des trajectoires relaxées pour les entrées logiques telles que la variable ON. Cette partie est développée en 3.4.2.
- Le calcul de l'erreur sur la grille d'optimisation que l'on présente en 3.4.3.1.

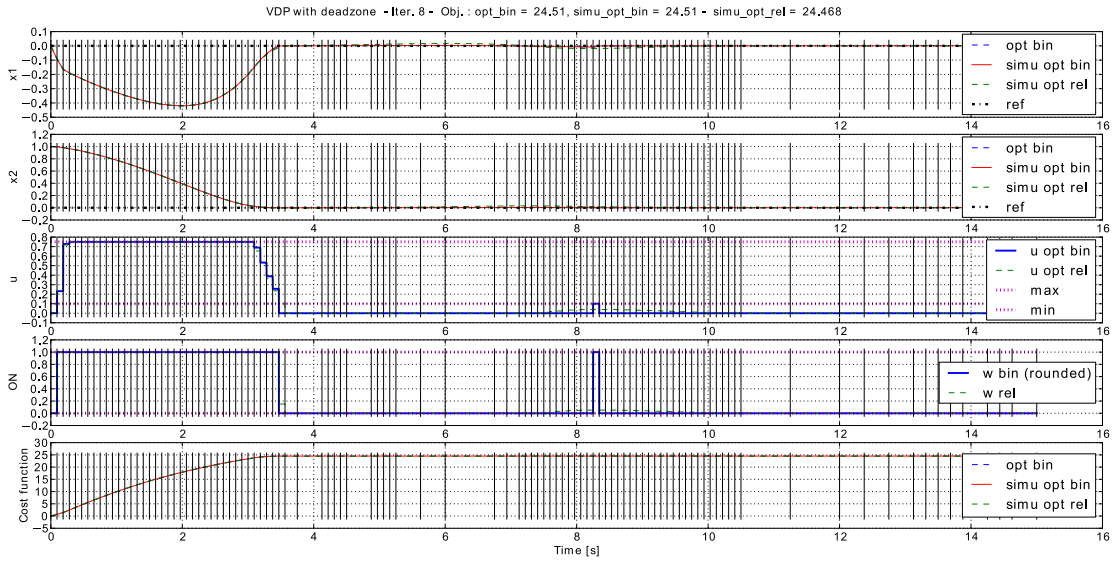


FIGURE 3.5 – Optimisation dynamique de l'oscillateur de VDP hybride - Itération 8

– L'adaptation itérative de cette grille en fonction de l'erreur et de la trajectoire des variables logiques relaxées. Cette partie est détaillée en 3.4.3.3.

On présentera enfin en 3.4.5 certaines améliorations de la méthode afin de pouvoir prendre en compte des coûts de transition, ainsi que des durées minimales pour chacun des modes.

3.4.1 Reformulations continues des problèmes dynamiques hybrides

Dans cette partie, on présente une méthode pour mettre les systèmes (DAEg-DAEm) sous une forme adaptée à la méthode d'optimisation proposée. Les modèles obtenus sont des modèles continus relaxés, qui sont affines en la commande ω. Cette mise en forme des modèles fait appel à :

– Une représentation spécifique des commandes logiques ω et continues u
– Une reformulation des équations dynamiques spécifiques aux modes dans (DAEm)
– Une reformulation des contraintes spécifiques aux modes dans (DAEm). On s'attachera en particulier aux contraintes représentant une plage de fonctionnement en marche.

La reformulation continue des aspects disjonctifs évoquée en 3.3.2.2 n'est pas utilisée ici. A la place, on propose d'utiliser une heuristique nommée *Sum Up Rounding* permettant de construire *a posteriori* des trajectoires pour les variables binaires tout en respectant les contraintes de disjonction, et ce d'après les trajectoires du problème relaxé.
Par ailleurs, la modélisation des fonctions de transition n'est pas nécessaire dans notre cas : en effet, on a fait l'hypothèse que les transitions implicites étaient reformulées de façon continue par lissage, et l'on n'a formulé aucune restriction sur les transitions explicites commandées.

3.4.1.1 Modélisation des modes

On retient la modélisation du système hybride sous la forme normale conjonctive, où le mode courant est la réunion des modes de chaque composant, plutôt qu'un unique mode défini

à l'échelle du système. Les raisons pour cela sont les suivantes :

- La forme disjonctive demande au modélisateur d'énumérer les différentes combinaisons valables des variables logiques. Cela ne va pas dans le sens d'une modélisation orientée objet, où les propriétés d'ensemble du système sont générées automatiquement par assemblage de composants élémentaires.
- La forme CNF permet également de définir plus facilement les contraintes spécifiques à l'état de fonctionnement d'un composant (par exemple un débit minimal si le composant est actif). En effet, dans le cas de la forme disjonctive, ces contraintes devraient être dupliquées pour chacun des modes où le composant est actif.

Le modèle contient un nombre n_d de disjonctions notées $D_j, \quad j \in \overline{1..n_d}$, ayant chacune un nombre n_ω^j de termes. Des variables logiques $\omega^{ji}(t) \in \{0;1\}$ sont introduites pour décrire l'activation de la branche i de la disjonction j. Cependant, dans le cas $n_\omega^j = 2$ (codant par exemple les états ON/OFF), on se limite à l'introduction d'une variable $\omega^j(t)$, l'autre terme pouvant se déduire par $1 - \omega^j(t)$. On appelle disjonction binaire cette première catégorie de disjonction où $n_\omega^j = 2$. Les disjonctions avec $n_\omega^j > 2$ sont quant à elles nommées disjonctions SOS1, et utilisent les variables logiques à deux indices $\omega^{ji}(t)$. On note que l'élimination d'une variable serait également possible pour ce type de disjonction en utilisant la propriété SOS1 (3.35c), mais induit une formulation plus compliquée des contraintes spécifiques à chaque mode.

Le mode actif du système à l'instant t est décrit par un vecteur $\mathbf{m}(t)$ de longueur n_d. Comme celui-ci est maintenu constant sur chaque élément, on note $\mathbf{m}_k$ sa valeur sur le pas de temps k. Les composantes m_k^j de ce vecteur décrivent la branche active de la disjonction j. Dans le cas des disjonctions binaires on a ainsi $m_k^j = \omega_k^j$, et dans le cas des disjonctions SOS1 $m_k^j = i$ t.q. $\omega_k^{ji} = 1$.

Durées minimales : Dans les systèmes énergétiques, il est très fréquent que la durée des modes d'un composant soit contrainte (temps minimum de marche, d'arrêt, etc.). Les méthodes développées doivent donc tenter de prendre en compte ce facteur. On peut considérer deux façons de modéliser les durées minimales :

- On considère une durée minimale par disjonction, valable pour tous les termes de la disjonction (durée minimale ON égale à la durée minimale OFF)
- On considère une durée minimale pour chacun des termes de la disjonction (durée minimale de marche différente de la durée minimale d'arrêt). Cette deuxième option est plus réaliste mais présente une implémentation beaucoup plus compliquée dans la méthode proposée. Pour cette raison, on se limitera à la première formulation.

L'interface avec l'algorithme d'optimisation et le modèle contient ainsi les éléments suivants :

- Une liste de n_d termes contenant les noms des variables ω^{ji} ou ω^j des différentes disjonctions
- Une liste des durées minimales T_{min}^j de chaque disjonction

Par ailleurs, le respect des durées minimales rend nécessaire la prise en compte des modes actifs dans chaque disjonction au début de l'horizon d'optimisation. Ainsi, aucune commutation ne doit être autorisée pour la disjonction j tant que le mode courant n'est pas actif depuis une durée au moins égale à T_{min}^j. On a ainsi besoin de connaître pour chaque disjonction j :

- Le mode $m_{pre}^j = m^j(t_0^-)$ actif dans chaque disjonction avant le début de l'horizon
- La durée T_{pre}^j depuis laquelle les modes m^j ont été activés avant le début de l'horizon

A titre d'exemple voici les informations D_j, T_{min}^j, m_{pre}^j et T_{pre}^j communiquées à l'algorithme d'optimisation pour un modèle contenant 3 disjonctions (2 disjonctions binaires et une dis-

jonction SOS1), de durées minimales différentes :

$$([(\omega^1), 10s, 1, 5s], [(\omega^{21}, \omega^{22}, \omega^{23}), 2s, 3, 2s], [(\omega^3), 5s, 0, 7s])$$

A l'état initial, le système est ainsi dans le mode $m = \begin{pmatrix}1 & 3 & 0\end{pmatrix}^T$. La commutation n'est pas permise dès l'instant initial pour la première disjonction car le mode 1 n'est actif que depuis 5 s alors que la durée minimale est de 10 s.

3.4.1.2 Modélisation des entrées

3.4.1.2.1 La notion d'entrée bloquée

On utilisera par la suite des entrées dites bloquées, c'est-à-dire des entrées constantes par morceaux maintenues (bloquées) à une même valeur sur un ou plusieurs éléments consécutifs, comme évoqué en 3.2.4.2. Dans le cadre du MPC, les entrées bloquées sont souvent utilisées pour réduire la dimension du vecteur de commande tout en maintenant un horizon de prédiction suffisamment long (par exemple on peut considérer que les entrées varient uniquement sur les premiers pas de temps, ou qu'elles ne sont autorisées à varier qu'à certains instants). Les instants où u est autorisé à varier sont donnés par leurs indices dans la grille de collocation $\mathcal{T}_0$. Cette liste d'indices est définie comme

$$\mathbf{b}_u = \{b_l \quad l \in \overline{1..n_{b_u}}\}$$

Elle correspond aux instants

$$\mathcal{B}_u = \{t_0; t_{b_1}; \ldots; t_{b_l}; \ldots; t_{b_{n_{b_u}}}\}$$

3.4.1.2.2 Modélisation des commandes logiques

Comme évoqué précédemment, la discrétisation retenue pour les variables doit être à même de représenter les trajectoires optimales du problème OCP initial, qui est de dimension infinie. Les trajectoires des variables binaires sont composées d'arcs où elles prennent les valeurs 0 ou 1 ; pour cette raison, des fonctions constantes par morceaux sont utilisées pour les modéliser. Avec n_s^j le nombre maximal d'arcs pour la disjonction D_j, on note $\mathcal{S}^j = \{s_1, \ldots, s_{n_{s+1}} = t_f\}$ la grille des $n_s^j + 1$ instants de commutation possibles (même si les commutations à t_f ne sont pas considérées). Dans la méthode de collocation, l'implémentation est simplifiée si l'on suppose que les commutations ne sont possibles qu'à la frontière des éléments. Les $\mathcal{S}^j$ sont donc des sous-ensembles de la grille $\mathcal{T}_0$ et peuvent être caractérisées pour chaque entrée par une liste d'indices $\mathbf{b}_\omega$. On note que comme les entrées binaires sont constantes sur chaque élément, il n'est pas nécessaire d'indexer ici les points de collocation. Ainsi ω_k^{ji} est la valeur de $\omega^{ji}(t)$ sur le segment T_k : $\omega^{ji}(t) = \omega_k^{ji}, \quad \forall t \in [t_{k-1}, t_k], \forall i \in D_j$.

On note $S_k^j = s_l - s_{l-1}, \forall f \in \overline{1..n_{n_s}}$ la durée des éléments des grilles $\mathcal{S}^j$. Ils correspondent à la durée minimale possible pour chaque mode. Idéalement, la discrétisation retenue pour les entrées binaires doit permettre de localiser les instants de commutation optimaux. Deux approches sont alors possibles :

- on peut traiter la grille $\mathcal{T}_0$ comme une variable d'optimisation. Dans la pratique, ce sont les durées T_k des éléments qui sont introduites comme n_e nouvelles variables. Cette approche a l'avantage de considérer les instants de commutation directement dans le

problème d'optimisation. En revanche, la retranscription des équations de collocation avec ces nouvelles variables introduit certaines non linéarités qui n'existaient pas dans le cas d'une grille fixée, rendant la résolution du problème plus complexe.
- on peut adapter la grille de façon itérative et *a posteriori* afin de localiser les commutations. On conserve dans ce cas pour chaque problème une retranscription sur une grille fixée, mais plusieurs itérations peuvent être requises pour localiser l'instant de transition optimal.

On notera que pour ces deux approches le nombre maximal de commutations reste limité par le nombre de pas de temps de la grille, même si l'on ne peut pas savoir *a priori* le nombre optimal de commutations du problème en dimension infinie.

Un autre aspect concerne le respect des durées minimales de chaque mode. La façon la plus simple pour le garantir est de choisir $S_k^j = T_{min}^j$. Cette solution n'est pas très satisfaisante car elle contraint les instants de commutation à être également répartis. Il est donc plus intéressant de garantir $S_k^j \geq T_{min}^j$ mais en localisant le début de chaque arc. On présentera deux méthodes permettant de garantir cette contrainte lors de la construction des trajectoires binaires. Ces deux méthodes implémentent des stratégies différentes pour localiser les instants de commutation au pas le plus fin dont nous disposons, en l'occurrence la grille $\mathcal{T}_0$.

3.4.1.2.3 Modélisation des entrées continues

Dans la méthode de collocation, les entrées continues $u(t)$ peuvent être modélisées de plusieurs façons. Comme ce choix est laissé à l'utilisateur, on présente ici ces différentes représentations.

Entrées polynomiales par morceaux : La première option est d'utiliser les mêmes points de collocation que pour les états et variables algébriques. A la différence des états cependant, les entrées peuvent être discontinues à la jonction de deux éléments. Les entrées sont ainsi des fonctions polynomiales par morceaux de degré $n_c - 1$ sur chaque élément.

Continuité des entrées : Parfois, il est cependant souhaitable de maintenir la continuité de u. Une méthode couramment employée est d'introduire dans le problème la commande $du = \dot{u}$. u est alors éliminée des entrées et devient un état du système. Cette méthode présente d'autres avantages : l'introduction explicite de du comme nouvelle variable dans le problème d'optimisation permet d'ajouter une pénalité sur ce terme dans la fonction objectif, ou de le restreindre à un certain domaine. Ces deux manipulations ont tendance à limiter les variations brusques de la commande qui ne sont pas souhaitées en général.

Entrées constantes par morceaux : Une troisième option pour la modélisation des entrées est d'utiliser les facteurs bloquants, ce qui permet d'obtenir des fonctions constantes par morceaux.

Entrées affines continues : Enfin, la dernière approche, qui mêle les deux précédentes, permet d'obtenir une commande continue affine par morceaux. Les facteurs bloquants sont utilisés sur la commande du nouvellement introduite afin d'obtenir du constant sur chaque élément et par conséquent u affine. Les commandes affines par morceaux présentent un grand intérêt pratique : elles garantissent le respect des contraintes de bornes sur u sur tout l'horizon. En effet, un problème souvent rencontré lorsque l'on utilise les polynômes de Lagrange

pour modéliser les entrées est la nature oscillatoire des trajectoires interpolées, qui peuvent amener à des violations des contraintes de bornes en dehors des points de collocation. Les entrées affines par morceaux semblent ainsi un bon compromis entre un nombre de degrés de liberté suffisant et des trajectoires moins susceptibles d'osciller ou de violer des contraintes. La modélisation des entrées affines par morceaux est la suivante : $\dot{u} = du$ sur la grille $\mathbf{B}_{du} = \mathcal{T}_0$ où la commande est autorisée à changer.

3.4.1.3 Formulation convexe des DAE

On cherche maintenant à disposer d'un modèle adapté pour l'optimisation par la méthode de collocation, c'est-à-dire un modèle continu soumis à des contraintes continues elles aussi. Pour cela, on présente une méthode pour relaxer le système hybride initial. Les modifications à apporter au modèle concernent les équations et contraintes spécifiques aux modes. On se place dans le cadre des systèmes modélisés par (DAEg)-(DAEm). Ce problème hybride peut être embarqué dans un même modèle à l'aide de la formulation suivante, parfois nommée *Outer Convexification* ou *Embedding*.

$$\dot{x}^j = \sum_{i \in D_j} \omega^{ji} f_i^j(x, z^g, u) \tag{3.35a}$$

$$z^j = \sum_{i \in D_j} \omega^{ji} \bar{g}_i^j(x, z, u) \tag{3.35b}$$

$$\sum_{i \in D_j} \omega^{ji} = 1, \quad 0 \leq \omega^{ji} \leq 1, \quad \forall j \in \overline{1..n_d} \tag{3.35c}$$

Avec $D_j = \overline{1..n_\omega^j}$. Une reformulation continue est possible en relaxant les commandes binaires $\omega^{ji}(t) \in \{0;1\}$ (selon les méthodes, cette relaxation se fera sur l'intervalle $[0,1]$, ou $[0,\infty[$). Le modèle obtenu à ce stade est alors un DAE sous forme de modèle d'état (ODE) pouvant être traité par la méthode de collocation. Il reste néanmoins à reformuler les contraintes spécifiques aux modes.

Dans la suite de ce chapitre on utilisera parfois la notation plus précise $x(t) = x(x_0, u, \omega; t)$ et $z(t) = z(x_0, u, \omega; t)$ pour les trajectoires de x et z correspondant à l'intégration du système (3.35) pour les entrées $[u(t), \omega(t)]$, avec $\omega(t) = \{\omega^{ji}(t), \quad j \in \overline{1..n_d}, i \in \overline{1..n_\omega^j}\}$

3.4.1.4 Reformulation des contraintes de plage de fonctionnement

On présente ici la reformulation des contraintes spécifiques aux modes pour un cas particulier qui se retrouve fréquemment dans les systèmes énergétiques : les contraintes de plage de fonctionnement en marche. Autrement dit, un composant peut être soit éteint, soit manœuvré entre un minimum et un maximum technique. On suppose que les composants en question sont commandables et que l'état (allumé/éteint) est une variable de décision. Une telle hypothèse s'intègre facilement avec une description du mode sous forme CNF (où une variable logique ω peut coder l'état d'un composant). La contrainte originale est la suivante :

$$\begin{cases} u & = 0 & ,\omega = 0 \\ u & \in [\underline{u}; \overline{u}] & ,\omega = 1 \end{cases}$$

La première formulation est la méthode Big-M décrite en (3.22) :

$$0 \leq u \leq \omega \overline{u} \tag{3.36}$$
$$\underline{u} - u \leq (1-\omega)M \tag{3.37}$$
$$\omega \in [0,1] \tag{3.38}$$

Avec $M \geq \underline{u}$. Cependant, dans la pratique, la reformulation Big-M semble tomber facilement dans des minima locaux. On propose donc la formulation suivante : On propose de relaxer ce type de contrainte en introduisant deux commandes : une commande continue u_{mod} et une commande logique ω. La commande ω est modélisée comme une entrée constante par morceaux. La commande u_{mod} peut être polynomiale, affine ou constante par morceaux. La commande u réellement appliquée au système devient une variable algébrique :

$$u = u_{mod}\omega, \quad u_{mod} \in [\underline{u}; \overline{u}], \omega \in [0,1] \tag{3.39}$$

Avec cette formulation, la valeur de u_{mod} n'a pas d'impact sur la solution tant que $\omega = 0$.

3.4.2 Méthode *Sum Up Rounding*

On a vu jusqu'alors comment transformer le modèle hybride (DAEg-DAEm) sous une forme convexifiée (3.35), en introduisant des commandes binaires ω codant les modes commandés, et en reformulant de façon continue les modes soumis à des transitions implicites. En relaxant les commandes ω par $\omega \in [0,1]$, le modèle obtenu est un modèle continu et différentiable, sur lequel on peut résoudre le problème de commande optimale (ici avec la méthode de collocation). Cependant, en raison de la relaxation de la contrainte $\omega \in \{0;1\}$, il est possible que les trajectoires calculées lors de l'optimisation ne soient pas des trajectoires du système hybride initial. On propose donc une phase de restauration permettant de générer des trajectoires binaires pour les commandes ω, en s'appuyant sur une méthode nommée *Sum Up Rounding*, initialement présentée dans [54]. On commence par revenir sur certains résultats théoriques justifiant l'emploi de cette méthode.

3.4.2.1 Génération de trajectoires binaires

Dans [53] apparaît un résultat important pour les systèmes commutés à deux modes relaxés selon (3.35). Le système original (commuté) est donné par :

$$\dot{x}(t) = f_{\omega(t)}(x(t), u(t)), \quad \omega(t) \in \{0;1\} \tag{3.40}$$

Le système relaxé (*embedded*) :

$$\dot{x}(t) = (1-\omega(t))f_0(x(t),u(t)) + \omega(t)f_1(x(t),u(t)), \quad \omega(t) \in [0,1] \tag{3.41}$$

Proposition 3.2. *[53] L'ensemble des trajectoires du système commuté (3.40) est dense dans l'ensemble des trajectoires du système (3.41).*

Deux cas se présentent alors : soit le problème relaxé (3.35) admet une solution binaire pour ω, la trajectoire obtenue est alors aussi solution du système commuté, soit il existe des intervalles où $\omega \in]0,1[$ et d'après la Proposition 3.2 il est possible de construire une

trajectoire sous-optimale admissible pour le système commuté arbitrairement proche de la solution relaxée. Ce résultat est par la suite généralisé aux systèmes à n_ω modes dans [55]. Par extension à notre cas, les trajectoires du système hybride (DAEg)-(DAEm) sont denses dans l'ensemble des trajectoires du système DAE convexifié (3.35). Il reste maintenant à déterminer la façon de construire cette trajectoire sous-optimale admissible pour le système commuté.
La méthode Sum Up Rounding (SUR) proposée par [54][56] est une méthode permettant de générer sur une grille temporelle une trajectoire binaire

$$\omega_{bin} : [t_0, t_f] \to \{0;1\}^{n_\omega}$$

d'après une trajectoire relaxée

$$\omega_{rel} : [t_0, t_f] \to [0,1]^{n_\omega}$$

La méthode s'applique aux systèmes hybrides convexifiés selon (3.35). On commence par introduire un premier résultat sur l'évolution d'un système (3.35) lorsqu'on lui applique les mêmes entrées continues $\tilde{u}$ mais deux entrées binaires $\omega_a(t)$ et $\omega_b(t)$ différentes. Soient donc

$$x_a(t) = x_a(x_0, \tilde{u}, \omega_a; t) \text{ et } x_b(t) = x_b(x_0, \tilde{u}, \omega_b; t)$$

les deux trajectoires d'état correspondantes, qui satisfont :

$$\begin{aligned} \dot{x}_a(t) &= \omega_a(t) f(x_a(t), \tilde{u}(t)) \\ \dot{x}_b(t) &= \omega_b(t) f(x_b(t), \tilde{u}(t)) \end{aligned}$$

.

Hypothèse H4. *On fait les hypothèses suivantes :*
- $\|\omega_a(t)\| \leq 1$, $\|\omega_b(t)\| \leq 1$
- *f est Lipschitz en x :* $\|f(x,\tilde{u}) - f(y,\tilde{u})\|_\infty \leq \mathcal{L}_x \|x - y\|_\infty$ *(voir Annexe B)*
- *f et sa dérivée sont bornées :*

$$\|f(x,\tilde{u})\|_\infty \leq M, \quad \left\| \frac{df(x,\tilde{u})}{dt} \right\|_\infty \leq C$$

- *Les trajectoires ω_a et ω_b vérifient $\forall t \in [t_0, t_f]$:*

$$\left\| \int_{t_0}^{t} [\omega_a(\tau) - \omega_b(\tau)]\, d\tau \right\|_\infty = \max_{j \in \overline{1..n_\omega}} \left| \int_{t_0}^{t} \left[\omega_a^j(\tau) - \omega_b^j(\tau) \right] d\tau \right| \leq \epsilon \tag{3.42}$$

La dernière hypothèse sera vérifiée par construction comme détaillé ensuite. Le théorème 3.3 de [56] donne une borne sur l'écart entre les trajectoires $x_a(t)$ et $x_b(t)$:

Théorème 3.3. *([56]) Sous les hypothèses (H4), l'erreur entre les trajectoires x_a et x_b est bornée par*

$$\|x_a(t) - x_b(t)\|_\infty \leq (\|x_a(0) - x_b(0)\|_\infty + (M + Ct)\epsilon)\, e^{\mathcal{L}_x t}, \quad \forall t \in [t_0, t_f]$$

Ou, si l'on considère que $x_a(0) = x_b(0)$

$$\|x_a(t) - x_b(t)\|_\infty \leq e^{\mathcal{L}_x t}(M + Ct)\epsilon$$

La démonstration se fait en utilisant un cas particulier du Lemme de Grönwall introduit en Annexe B. Comme l'on peut le constater, l'écart maximal entre les trajectoires $x_a(t)$ et $x_b(t)$ dépend linéairement de la borne ϵ introduite en (3.42).
Un cas particulier du Théorème 3.4 est le suivant : $\omega_a \leftarrow \omega_{rel} \in [0;1]$ et $\omega_b \leftarrow \omega_{bin} \in \{0;1\}$ (admissible pour le système hybride). Partant de ce constat, et étant donnée une trajectoire relaxée ω_{rel}, on peut chercher à construire une trajectoire binaire ω_{bin} qui minimise l'intégrale (3.42) (et donc la borne ϵ), et par conséquent minimise aussi la borne sur l'écart entre les trajectoires des états x_{rel} et x_{bin} correspondantes.
On a vu par ailleurs que la forme de Bolza de la fonction objectif peut toujours se mettre sous forme $x_J(t_f)$ avec $\frac{dx_J}{dt} = \Phi(x(t))+$ Cette stratégie est l'essence même de la méthode Sum Up Rounding. Deux versions de cette méthode sont données, la première dans le cas où les composantes de ω vérifient la propriété SOS1 (3.35c), la deuxième sans cette restriction.

Pour le cas des disjonctions de type binaire, la méthode SUR originale est donnée dans l'Algorithme 4 et illustrée en Figure 3.6. La trajectoire de la variable relaxée ω_{rel} est représentée sur la ligne du haut. Sur la ligne du bas, on présente le déroulement de l'algorithme sur les 4 premiers pas de temps pour produire la trajectoire ω_{bin}. L'algorithme SUR consiste à comparer l'intégrale de la solution relaxée jusqu'au pas de temps t_{k+1} (aire grise) avec l'intégrale de la solution binaire jusqu'à l'instant t_k (aire noire) afin de définir progressivement la valeur $\omega_{bin,k}$ de ω_{bin} pour le pas de temps k (trait plein noir sur la deuxième ligne). $\omega_{bin,k}$ est ainsi fixé à 1 sur le pas k si la différence entre les deux intégrales est supérieure à $0,5T_k$, à 0 sinon. Cette stratégie permet de minimiser la différence entre les aires grises et noires comme mentionné dans le Théorème 3.4. Un autre algorithme et une version similaire de ce théorème sont également donnés pour les disjonctions SOS1.

Algorithme 4 Sum Up Rounding binaire

Entrées: Grille $\mathcal{W}_{rel} = [\omega_{rel,k}]$, durée des éléments T_k : $T = [T_k]$, $\quad i \in \overline{1..n_\omega}, k \in \overline{1..n_e}$
Pour $k = 1$ **à** n_e **Faire**
$q_k \leftarrow \sum_{n=1}^{k} \omega_{rel,n} T_n - \sum_{n=1}^{k-1} \omega_{bin,n} T_n$
$\omega_{bin,k} \leftarrow$ 1 **Si** $q_k \geq 0,5T_k$ **Sinon** 0
Fin Pour
Retourner $\mathcal{W}_{bin} = [\omega_{bin,k}], \quad k \in \overline{1..n_e}$

Théorème 3.4. *([56]) Si $\omega_{rel} : [t_0, t_f] \to [0,1]$ est une fonction mesurable et $\omega_{bin} : [t_0, t_f] \to \{0;1\}$ est construit selon l'Algorithme 4, alors*

$$\left| \int_{t_0}^{t_f} \omega_{bin}(t) - \omega_{rel}(t) dt \right| \leq 0,5 \max_{\overline{1..n_e}} T_k$$

Dans le cas où $\omega(t)$ est un vecteur, la méthode SUR est appliquée à chacune des composantes $\omega^i(t)$ et l'on remplace la valeur absolue de l'intégrale par la norme infinie.

3.4.2.2 Méthode SUR sur une grille bloquée

La méthode SUR décrite précédemment autorise la commutation à tous les instants de la grille. Or comme mentionné précédemment, certains modes peuvent avoir des contraintes

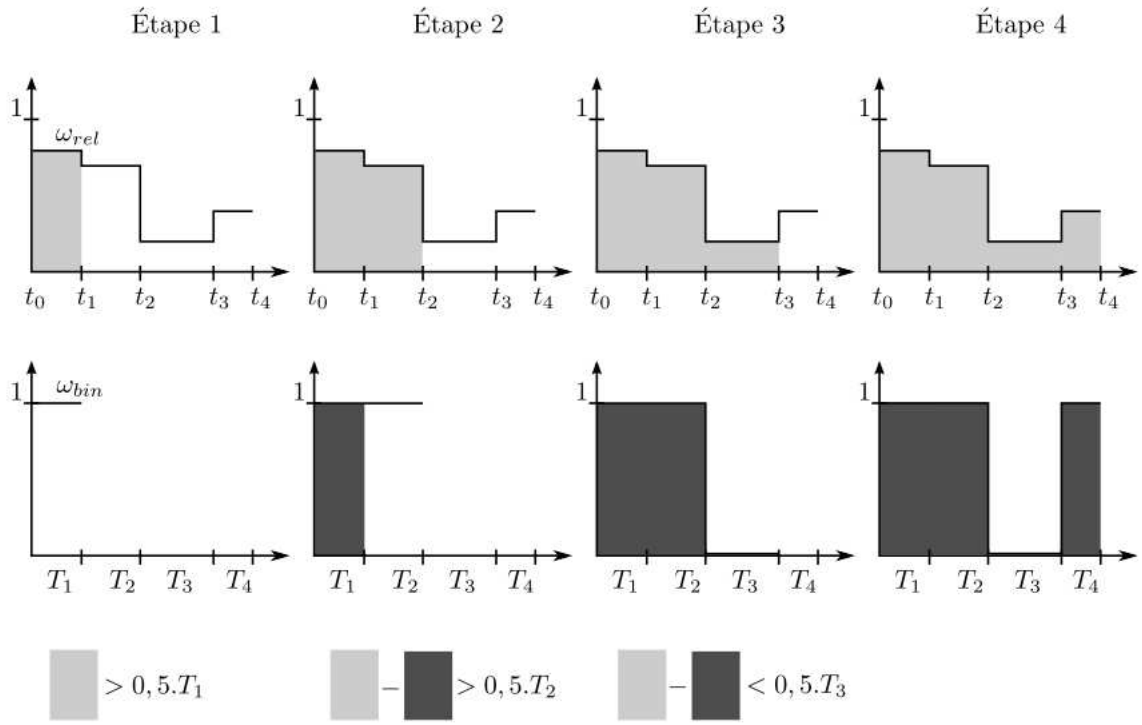


FIGURE 3.6 – Illustration de la méthode SUR sur une disjonction binaire

de durées minimales. On propose donc ici une version adaptée de la méthode Sum Up Rounding. La modification par rapport à [54] concerne la restriction des instants de commutation possibles à ceux d'une grille $\mathcal{S}^j \subseteq \mathcal{T}_0$ spécifique à chaque disjonction j : $\mathcal{S}^j = \{t_{s_1}, \ldots, t_{s_l}, \ldots, t_{s_{n_s[j]+1}} = t_f\}, \quad l \in \overline{1..n_s[j]}$. Dans cette thèse, on propose une méthode permettant de créer ces grilles de façon séquentielle afin de localiser les commutations sur la grille la plus fine dont on dispose ($\mathcal{T}_0$) et de garantir la durée minimale T_{min} des modes sans toutefois imposer à l'avance une grille de commutation à la maille de T_{min}.

Plusieurs des grilles temporelles définies en 3.2.4.1 vont être utilisées : la grille des éléments $\mathcal{T}_0$, celle de leur durée T, ainsi que des grilles d'indices **s** correspondant aux indices de certains instant t_k de $\mathcal{T}_0$. Les grilles sont représentées sous forme de tableaux, avec les indices entre crochets. Les indices suivants sont utilisés :

- $k \in \overline{1..n_e}$: les éléments de la grille de collocation $\mathcal{T}_0$
- $j \in \overline{1..n_d}$: les disjonctions du système
- $i \in \overline{1..n_\omega[j]}$: les termes de la disjonction j
- $l \in \overline{1..n_s[j]}$: les éléments S^j des grilles de commutation

Pour simplifier les notations dans les algorithmes, on introduit maintenant la fonction *indice(condition,var)* qui retourne le premier indice de la grille vérifiant la *condition* en faisant varier la variable *var*.

Dans la méthode SUR, les $\omega^i_{bin,k}$ sont définis séquentiellement de la façon suivante : sur chaque pas de temps $[t_k, t_{k+1}[$, l'algorithme évalue pour chaque mode la valeur de (3.42) attendue si on ne le sélectionne pas sur cet intervalle, c'est-à-dire l'augmentation de ϵ cor-

respondante sur ce pas de temps. Cette dégradation est calculée par les variables auxiliaires q_k^i. On sélectionne alors le mode correspondant à la plus grande valeur q_k^i. Si une ambiguïté existe entre deux modes, priorité est donnée au mode ayant le plus petit index k.

Algorithme 5 Sum Up Rounding sur une grille de commutation

Entrées: Grille $\mathcal{W}_{rel} = [\omega_{rel,k}^i]$, durées des éléments $T_k : T = [T_k]$, $\quad i \in \overline{1..n_\omega}, k \in \overline{1..n_e}$, grille de commutation $\mathbf{s}[s_l], l \in \overline{1..n_s}$, mode initial ω_0 à t_0^-

$\omega_{bin,k} \leftarrow 0, \quad \forall k \in \overline{1..n_e}$
$\omega_{bin,k} \leftarrow \omega_0, \quad \forall k \in \overline{0, s_1}$
Pour $f = 1$ **à** n_s **Faire**
  $q_s^i \leftarrow \sum_{n=1}^{s_l} \omega_{rel,n}^i T_n - \sum_{n=1}^{s_{l-1}} \omega_{bin,n}^i T_n, \quad \forall i \in \overline{1..n_\omega}$
  Si $n_\omega > 1$ **alors**
    $m \leftarrow indice(q_k^i \geq q_k^j \quad \forall j \neq i, i)$
    $\omega_{bin,k}^m \leftarrow 1, \quad \forall k \in \overline{s_l..s_{l+1}}$
  Sinon
    $\omega_{bin,k}^1 \leftarrow$ 1 **Si** $q_s^1 \geq 0,5 T_k$ **Sinon** 0 , $\quad \forall k \in \overline{s_l..s_{l+1}}$
  Fin Si
Fin Pour
Retourner $\mathcal{W}_{bin} = [\omega_{bin,k}^i], \quad \forall k \in \overline{1..n_e}$

3.4.3 Raffinement de la grille

On propose ici un algorithme de raffinement de la grille temporelle ayant deux objectifs. Le premier est la localisation des instants optimaux des commutations, celles-ci n'étant permises qu'à la jonction de deux éléments. On a ainsi $\mathcal{S}^j \subseteq \mathcal{T}_0, \quad \forall j \in \overline{1..n_d}$. Le second est de garantir la précision de la méthode de collocation, du fait que les trajectoires des variables $\tilde{X}(t) = [\tilde{x}(t), \dot{\tilde{x}}(t), \tilde{z}(t)]$ calculées par la méthode ne sont que des approximations polynomiales des trajectoires réelles $X(t) = [x(t), \dot{x}(t), z(t)]$, obtenues pour la même trajectoire de commande optimale $\tilde{u}(t)$. La précision de $\tilde{X}(t)$ dépend ainsi du nombre d'éléments et de points de collocation choisis, qui à leur tour influent sur la dimension du problème. Enfin, il existe souvent un compromis entre la proximité de la solution du problème original et relaxé et un nombre élevé de commutations. Dans le cas des systèmes énergétiques, les commutations fréquentes sont souvent peu souhaitables. Pour pallier ce problème, une solution est d'imposer pour chaque entrée logique une durée minimale pour chaque mode et donc une certaine "granularité" des grilles de commutation $\mathcal{S}^j$ utilisées. Le prix à payer est un écart potentiellement plus important entre les trajectoires binaires et relaxées (donc également un écart plus grand entre J_{rel} et J_{bin}) comme suggéré par le Théorème 3.4.

La difficulté posée ici est due à la quantification des instants de commutation possibles imposée par la grille $\mathcal{T}_0$, et le fait que celle-ci doive être utilisée comme support pour toutes les disjonctions j, qui n'ont *a priori* aucune raison de partager les mêmes instants de commutation. On a fait le choix ici de raffiner la grille temporelle en adaptant uniquement le nombre et la durée des éléments, c'est-à-dire en modifiant la grille $\mathcal{T}_0$. On aurait pu également adapter

la grille de collocation $\mathcal{T}_1$, en ajustant le nombre de points de collocation au sein de chaque élément. Ajouter des points de collocation sur un élément permet en effet d'accroitre l'ordre de la méthode d'intégration et d'améliorer localement la précision aux endroits nécessaires. Cependant, l'implémentation actuelle de la méthode de collocation dans JModelica impose un nombre de points de collocation similaire sur tous les éléments. L'adaptation du nombre de points de collocation n'est donc pas considérée pour l'instant.
La première étape est maintenant de quantifier l'erreur due à la méthode de collocation, avant d'exposer la stratégie mise en place pour la contrôler.

3.4.3.1 Estimation de l'erreur de la méthode de collocation

Deux notions sont importantes pour quantifier la précision de la méthode de collocation, que l'on apparente à une méthode d'intégration numérique à un pas : l'erreur locale, et l'erreur globale. L'erreur globale tout d'abord correspond simplement à l'erreur entre la trajectoire réelle et simulée. C'est celle qui nous intéresse en pratique. On note $\tilde{X}^T(t) = [\tilde{x}^T(t), \dot{\tilde{x}}^T(t), \tilde{z}^T(t)]$ les trajectoires obtenues par optimisation et $X(t) = [x(t), \dot{x}(t), z(t)]$ celles obtenues par simulation avec la trajectoire de commande optimale $\tilde{u}(t)$. En supposant que les trajectoires $X(t)$ et $\tilde{X}(t)$ sont correctement normalisées, on peut retenir la définition suivante pour l'erreur globale sur un élément k :

$$E_k = \left\| \int_{t_{k-1}}^{t_k} \left| X(t) - \tilde{X}(t) \right| dt \right\|_\infty \tag{3.43}$$

Cette mesure de l'erreur n'est pas vraiment adaptée pour arbitrer le raffinement de la grille car elle ne tient pas compte du fait que l'erreur tend à s'accumuler au cours du temps. Le raffinement de la grille est ainsi susceptible de diviser inutilement la grille en fin d'horizon alors qu'il aurait parfois été nécessaire de le faire avant que la tolérance sur l'erreur globale ne soit dépassée. On s'intéresse donc à une autre mesure de l'erreur : l'erreur locale.
Pour les méthodes d'intégration numérique à un pas, l'erreur locale sur un élément correspond à l'erreur d'intégration sur un pas, *partant des mêmes conditions initiales* $\bar{X}_k$. Or comme mentionné précédemment, les trajectoires X et $\tilde{X}$ diffèrent, et ne partagent pas les mêmes valeurs aux instants t_k : la condition $X(t_k) = \tilde{X}(t_k) = \bar{X}_k$ n'est alors pas vérifiée. On introduit donc sur chaque pas de temps k une trajectoire estimée $\hat{X}^k$ ayant pour condition initiale $\hat{X}^k = \bar{X}_k$ pour servir de référence au calcul de l'erreur locale. Deux choix sont alors possibles pour l'initialisation des $\hat{X}^k$: on peut soit poser $\bar{X}_k = \hat{X}^k(t_k) = X(t_k)$, soit $\bar{X}_k = \hat{X}^k(t_k) = \tilde{X}(t_k)$. On retiendra la deuxième solution, c'est-à-dire que le résultat issu de la méthode de collocation nous donne les valeurs d'initialisation pour les trajectoires estimées $\hat{X}^k$. L'erreur globale peut alors être vue comme l'accumulation des erreurs locales et leur propagation à travers le système simulé. On propose maintenant deux façons de définir l'erreur locale.

Erreur locale sur les variables :

La première méthode est basée sur l'erreur par rapport à $\hat{X}^k$ et est calculée comme suit : on initialise l'état estimé d'après le résultat d'optimisation $\hat{X}^k(t_k) = \tilde{X}(t_k)$, puis on simule le système sur un pas avec la trajectoire $\tilde{u}$, en utilisant une tolérance très faible. L'erreur locale peut alors être définie sur la grille $\mathcal{T}_0$ comme :

$$\epsilon_k = \|e(t_{k+1})\|_\infty , \quad \text{avec} \quad e(t) = \hat{X}^k(t) - \tilde{X}(t) \tag{3.44}$$

On peut aussi considérer la forme intégrale suivante.

$$\epsilon_k = \left\| \int_{t_{k-1}}^{t_k} |e(t)| dt \right\|_\infty \tag{3.45}$$

Cependant, les trajectoires $\hat{X}^k$ obtenues par simulation ne sont données que par un ensemble discret de points. Il est donc nécessaire de les interpoler dans un premier temps afin d'obtenir ces trajectoires comme une fonction continue $\hat{X}^k(t)$ et pouvoir évaluer l'intégrale.

Erreur locale sur la dynamique :

Dans la deuxième méthode, on évalue plutôt le respect de la dynamique le long de la trajectoire calculée. Dans l'implémentation de la méthode de collocation de JModelica, le DAE est considéré sous forme implicite (DAEi), et le respect de cette équation n'est garanti qu'aux points de collocation t_{kc} :

$$F(\tilde{X}(t_{kc}), \tilde{u}(t_{kc})) = F(x_{kc}, \dot{x}_{kc}, z_{kc}, u_{kc}) = 0, \quad \forall k \in \overline{1..n_e}, c \in \overline{1..n_c} \tag{3.46}$$

On définit ainsi l'erreur instantanée sur la dynamique comme : $\eta(t) = F(\tilde{X}(t), \tilde{u}(t))$. Cependant $\eta(t)$ qui vaut exactement 0 aux points de collocation est susceptible d'osciller entre ces points, générant des compensations dans l'intégrale. Pour cette raison, on retient comme mesure l'*erreur locale absolue* ζ_k [57] définie par :

$$\zeta_k = \left\| \int_{t_{k-1}}^{t_k} |\eta(t)| dt \right\|_\infty \tag{3.47}$$

Cette méthode est intéressante car elle ne nécessite pas l'appel à un algorithme d'intégration numérique. Son implémentation nécessite les éléments suivants :

- les trajectoires $\tilde{X}(t)$ et $\tilde{u}(t)$ qui sont les polynômes d'interpolation utilisés dans la méthode de collocation et peuvent être déduits facilement des points $\tilde{X}_{ik}$, $\tilde{u}_{ik}$
- l'intégrale de la fonction dans (3.47) est calculée par la méthode de quadrature de Romberg comme suggéré par [58]. Cette méthode permet d'évaluer l'intégrale d'une fonction (ici $|\eta(t)|$) en évaluant celle-ci en certains points de l'intervalle. La quadrature de Romberg peut aussi être utilisée pour évaluer l'intégrale dans (3.45).

Erreur locale de l'optimalité des solutions :

Une hypothèse implicite que l'on a faite pour les deux méthodes précédentes est l'exactitude de la trajectoire de commande calculée : $u(t) = u^*(t) = \tilde{u}(t)$, mais comme souligné par [57], les trajectoires optimales des entrées doivent normalement aussi vérifier les équations adjointes. Récemment, les auteurs [59] ont fait une proposition pour prendre en compte l'erreur sur les conditions d'optimalité des variables duales dans le raffinement de la grille. Cependant, les états adjoints ne sont pas considérés explicitement dans les méthodes directes telles que la

méthode de collocation (même si certaines correspondances existent entre les états adjoints et les multiplicateurs de Lagrange associés dans le NLP [39]), et nous ne les considèrerons pas ici. Dans notre cas, le raffinement de la grille utilisé contrôle donc seulement la précision des résultats pour une trajectoire $u(t)$ donnée mais ne prend pas en compte son optimalité.

3.4.3.2 Normalisation de l'erreur

On cherche à garantir une erreur locale relative sur la grille d'optimisation. Pour cette raison, la mesure de l'erreur doit être normalisée. Cela est crucial pour les modèles physiques qui font intervenir des variables ayant des ordres de grandeur très différents (impact sur le calcul de (3.44)), et peuvent mêler des dynamiques lentes et rapides (impact sur le calcul de (3.47) . Pour la méthode (3.44), les valeurs nominales des différentes composantes sont utilisées (l'indice temporel est ici laissé car il est possible de définir une trajectoire nominale plutôt qu'une valeur nominale dans l'algorithme de collocation). Pour chaque composante i du vecteur X, l'erreur est donnée par

$$\epsilon_{rel,k}^{(i)} = \frac{\epsilon_k^{(i)}}{\max(X_{nom,k}^{(i)}, \delta)} \tag{3.48}$$

Avec δ un petit nombre servant à régulariser l'expression. Dans JModelica, il est également possible de travailler directement avec des variables normalisées. L'implémentation est détaillée dans [60].
Pour la deuxième méthode (3.47), la façon de normaliser est moins évidente. Un premier commentaire que l'on peut faire est que la trajectoire $\tilde{X}$ calculée par la méthode de collocation peut être considérée comme une trajectoire perturbée autour de la trajectoire réelle $\hat{X}$, qui elle satisfait en tout point $F(\hat{X}(t), \tilde{u}(t)) = 0$. Pour cette raison, on peut utiliser pour $\eta(t)$ une approximation du premier ordre autour de $\hat{X}$:

$$\begin{aligned}\eta(t) = F(\tilde{X}(t), \tilde{u}(t))| = |F(\hat{X}(t) + e(t), \tilde{u} + \delta_u) \approx F(\hat{X}, \tilde{u}) + \nabla F(\hat{X}, \tilde{u}) \begin{bmatrix} e(t) & \delta u(t) \end{bmatrix}^T \\ \approx \nabla F(\hat{X}, \tilde{u}) \begin{bmatrix} e(t) & 0 \end{bmatrix}^T\end{aligned}$$

Ici, $\delta u(t) = 0$ car la même trajectoire des entrées $\tilde{u}$ est utilisée pour générer $\hat{X}$ et $\tilde{X}$.

De cette expression, il peut sembler raisonnable d'utiliser une normalisation s'appuyant sur les valeurs de ∇F, évalué le long de la trajectoire $(\hat{X}(t), \tilde{u}(t))$. Un point intéressant à noter est qu'il s'agit d'une quantité calculée pour d'autres raisons par le solveur IPOPT : les équations de collocation (3.46) sont en effet des contraintes du NLP, et sont linéarisées au moment de la résolution par IPOPT. Or IPOPT effectue également une normalisation des contraintes. On utilisera donc ici les facteurs de normalisation donnés par IPOPT plutôt que de chercher à en créer de nouveau. La normalisation utilisée par IPOPT pour $\nabla F_j(X_{ik}, u_{ik})$ (c'est-à-dire pour chaque composante j de F_j) est la suivante :

$$(\nabla F_j(X_{ik}, u_{ik}))_{nom} = d_{ijk} \nabla F_j(X_{ik}, u_{ik}), \quad d_{ijk} = \min \left\{ 1, \frac{g_{max}}{\left\| \nabla F_j(X_{ik}^0, u_{ik}^0) \right\|_\infty} \right\}$$

Avec g_{max} un paramètre d'IPOPT (100 par défaut) et $[X_{ik}^0, u_{ik}^0]$ les valeurs initiales données à l'algoritme. Dans JModelica, ces valeurs proviennent des trajectoires d'initialisation fournies

à l'algorithme (option *init_ traj*) ou des attributs *InitialGuess* si ces trajectoires ne sont pas renseignées. On souligne au passage l'importance d'une bonne initialisation des variables du problème discrétisé (3.18) résolu par IPOPT, qui est un algorithme d'optimisation locale.

3.4.3.3 Stratégies d'adaptation de la grille

L'erreur locale, qui conditionne la précision des résultats obtenus par simulation dépend principalement de la méthode d'intégration employée (même s'il existe d'autres facteurs d'erreur tels que les erreurs d'arrondi). La précision d'une méthode d'intégration peut être caractérisée par son ordre :

Définition (Ordre d'une méthode d'intégration numérique) Une méthode d'intégration numérique est d'ordre n si l'erreur locale est en $\mathcal{O}(T^n)$, T étant la durée du pas. On appelle aussi cette erreur "erreur de troncature", en référence à l'approximation de f (dans le cas d'une ODE) par son développement en série de Taylor.

Les méthodes de collocation introduites brièvement en 2.2.3.1 sont des méthodes de Runge Kutta implicites. Leur ordre p dépend des points de collocation employés : les points de Gauss Legendre donnent une méthode d'ordre $2 * n_c$, et ceux de Gauss-Radau (un point de collocation en $\tau = 1$) une méthode d'ordre $2 * n_c - 1$ [39]. On peut ainsi approcher l'erreur locale comme :

$$\epsilon_k \approx c_k T_k^p$$

Avec c_k une constante. Cependant, il est noté dans [58] qu'en présence d'un DAE d'ordre supérieur à 1 ou de contraintes actives, l'ordre de la méthode peut être dégradé. Un entier r est ainsi introduit pour quantifier cette réduction d'ordre, mais dont la détermination reste empirique.

$$\epsilon_k \approx c_k T_k^{p-r}$$

En première approximation et sans considérer r, il est possible d'évaluer l'erreur prédite à l'itération $n+1$ si l'on scinde ou fusionne deux éléments par :

$$\epsilon_{p,k}^{(n+1)} = \epsilon^{(n)} \left(\frac{T_k^{(n)}}{T_k^{(n+1)}} \right)^p \tag{3.49}$$

Le raffinement de la grille peut ainsi être vu comme un moyen de contrôler indirectement l'erreur globale en agissant sur la taille des éléments, et donc l'erreur locale. Cependant, et notamment dans le cas de systèmes instables, il est possible que le raffinement de la grille basé sur l'erreur locale uniquement soit insuffisant : par exemple, dans le problème de stabilisation de l'oscillateur de Van Der Pol (instable), le résultat de la Figure 3.7. Ici, on constate que l'optimisation amène bien les états à l'origine. Cependant, la simulation du système montre que ceux-ci divergent en réalité. La raison est la suivante : les erreurs locales sont calculées en prenant comme point de départ la trajectoire d'optimisation, qui atteint l'état stationnaire $(x_s, u_s) = (0, 0)$ à un instant t_s. Or la trajectoire simulée diffère quelque peu de cette trajectoire, et n'atteint pas exactement ce point stationnaire mais $(x_\epsilon, 0)$ à t_s. S'agissant d'un système instable, cette erreur se propage et devient de plus en plus grande.

On a présenté jusqu'alors une mesure de l'erreur locale sur un élément, que l'on souhaite garder en dessous d'une certaine tolérance tol_ϵ. On peut alors choisir de scinder les éléments

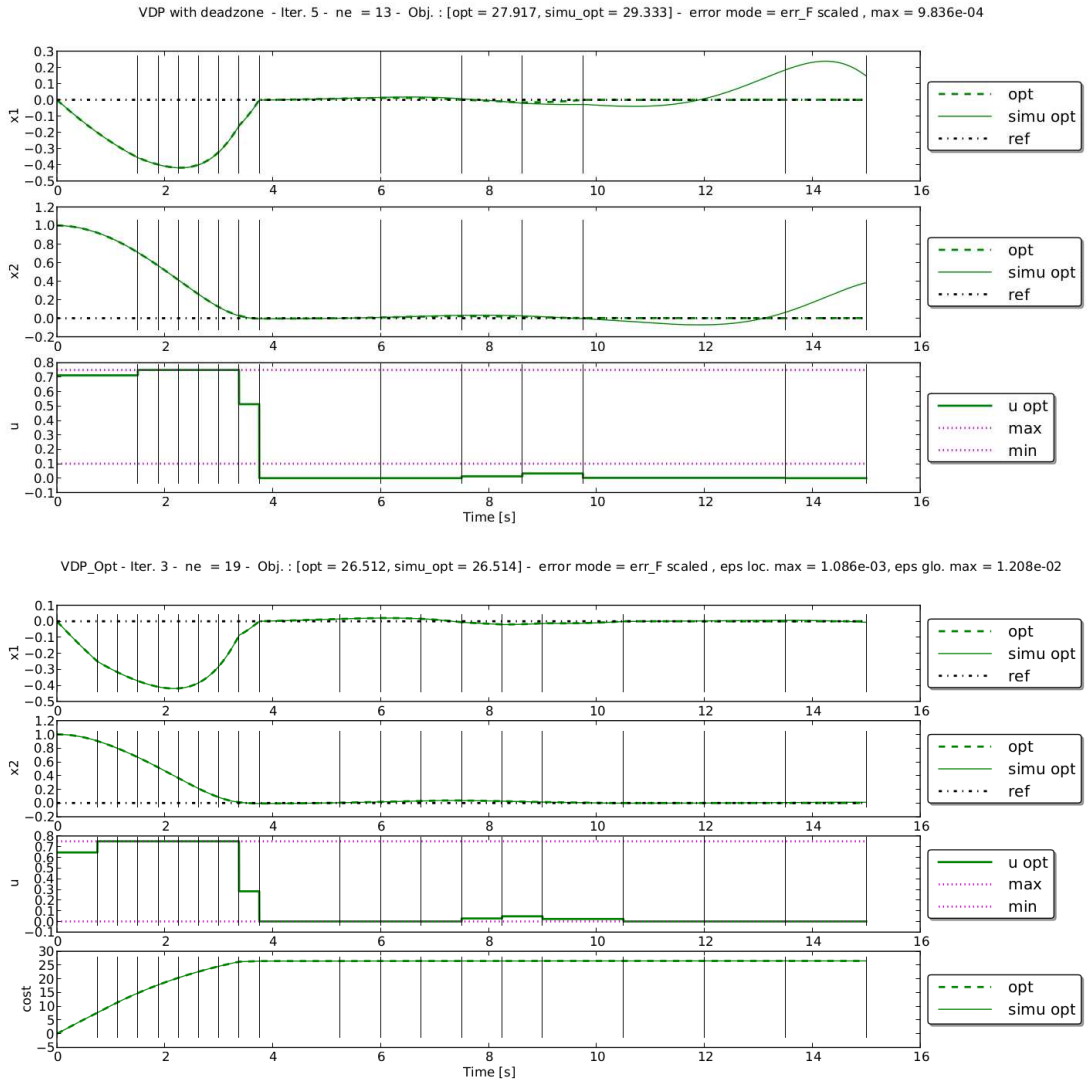


FIGURE 3.7 – Echec de stabilisation de l'oscillateur VDP avec un raffinement de grille basé sur l'erreur locale (haut) - Amélioration avec le contrôle de l'erreur globale (bas)

ne respectant pas ce critère, car comme évoqué précédemment, réduire le pas d'intégration augmente la précision numérique.
La grille peut aussi être raffinée dans l'objectif de localiser les instants de commutation, et l'on doit disposer également d'un critère pour décider si un élément doit être scindé dans cette optique. On commence par définir l'ensemble des modes admissibles pour une disjonction contenant n_ω variables, qui est noté $\mathcal{M}(n_\omega)$. Ainsi, $\mathcal{M}(1) = \{0; 1\}$ et $\mathcal{M}(3) = \{[1,0,0],[0,1,0],[0,0,1]\}$. On peut alors choisir de scinder les éléments où les variables logiques relaxées n'appartiennent pas à ces ensembles. Dans la pratique, une certaine tolérance tol_b est accordée sur le caractère binaire des solutions. On note $\mathcal{M}_{\mathrm{tol}_b}(n_\omega)$ l'ensemble des valeurs relaxées admissibles pour les commandes binaires. On autorise ainsi $\omega \in [0; \mathrm{tol}_b] \cup [1 - \mathrm{tol}_b; 1]$.

On présente maintenant l'Algorithme 6 utilisé pour le raffinement de la grille courante, et pour la génération de trajectoires binaires respectant les durées minimales des modes. La grille courante $\mathcal{T}_0^n$ de l'itération n est adaptée en modifiant le nombre d'éléments n_e et leur durée T_k, afin de préparer la future grille $\mathcal{T}_0^{n+1}$. L'ancienne grille est parcourue progressivement pour préparer la suivante et l'on scinde les éléments sur lesquels la précision est insuffisante, ou sur lesquels la solution relaxée n'est pas binaire pour au moins une des disjonctions.
Par ailleurs, comme les modes doivent respecter des contraintes de durées minimales, les transitions ne sont pas permises pour tous les instants de la grille $\mathcal{T}_0^{n+1}$. L'algorithme construit donc progressivement des grilles de commutation $\mathcal{S}^{n+1}[j]$ pour toutes les disjonctions. Pour des raisons de commodité, on utilise pour les grilles de commutation la notation sous forme de tableau d'indices $\mathbf{s}$ introduite en (3.13).

Les trajectoires binaires sont alors générées au fur et à mesure de la construction de la grille $\mathcal{S}^{n+1}[j]$ en appliquant la méthode SUR de l'Algorithme 5. Le dernier indice s_M contenu dans $\mathbf{s}[j]$ détermine jusqu'à quel pas de temps $t_{s[j][M]}$ la méthode SUR est appliquée. Les valeurs des variables binaires relaxées sont stockées dans $\mathcal{W}_{rel}^n[j]$, et les durées des pas de temps de cette grille sont données dans $T[k]^n$. Le mode courant est déterminé par $m[j] = m^j$ qui représente la branche active pour chaque disjonction j (on a $\omega^{jm^j} = 1$). La variable $T_{pre}[j]$ indique depuis combien de temps $\mathbf{m}_{pre}$ est le mode actif à t_0. On utilise également au sein de l'algorithme une variable $T_a[j]$ qui stocke la durée depuis laquelle le mode $\mathbf{m}_k$ est actif à l'instant k.

L'Algorithme 6, nommé SURminT procède selon les étapes suivantes :

1. l.1 à l.9 : La grille courante T^n est tout d'abord raffinée aux endroits où la précision numérique est insuffisante et/ou aux endroits où les solutions relaxées ne sont pas binaires. On obtient ainsi la grille T^{n+1} sur lequel l'algorithme SURminT est appliqué.

2. l.10 à l.11 : initialisation et parcours des disjonctions

3. l.12 à l.17 : à l'instant initial, le mode m_{pre}^j est actif depuis une durée T_{pre}^j et ne peut changer tant qu'il n'a pas atteint sa durée minimale T_{min}^j. On parcourt donc les indices k de la grille en maintenant $m_k^j = m_{pre}^j$ et en comptabilisant la durée totale d'activation dans la variable D_s jusqu'à ce que la condition $T_{pre}^j + D_s > T_{min}^j$ soit vérifiée.

4. l.18 à 40 : parcours des indices k de la grille

5. l.19 à 24 : recherche de l'indice k_{next} jusqu'auquel le prochain mode devra être maintenu si une commutation se produit à l'instant k (tel que $t_{k_{\text{next}}} - t_k \geq T_{min}^j$).

6. l.25 à 31 : détermination du mode courant, le terme auxiliaire q de la méthode SURminT utilise l'intervalle $[t_k; t_{k_{\text{next}}}]$ comme horizon de comparaison (et non pas $[t_k; t_{k+1}]$ comme dans la méthode SUR)

7. l.32 à 39 : détermination du mode courant m_k^j. Le prochain instant de commutation possible est k_{next} si une commutation a eu lieu, $k+1$ sinon.

Algorithme 6 Sum Up Rounding avec durée minimale des modes (SURminT)

Entrées: Grille temporelle T^n, trajectoires relaxées $\mathcal{W}^n_{rel} = \left[\omega^{ji}_{rel,k} \ / \ \omega^{j}_{\mathrm{rel,k}}\right]$ et erreur locale ϵ sur T^n, modes initiaux $\mathbf{m}_{pre}$ et leur durée d'activité $\mathbf{T}_{pre}$ à l'instant initial, tolérances tol_b et tol_ϵ

Pour $k = 1$ **à** n_e **Faire**
  $y[j] \leftarrow \mathcal{W}^n_{rel}[j][k], \quad \forall j \in \overline{1..n_d}$
  Si $\epsilon[k] \geq \mathrm{tol}_\epsilon$ **OU** $\exists j \in \overline{1..n_d}$ t.q. $|y[j] \notin \mathcal{M}_{\mathrm{tol}_b}(n_\omega[j])$ **alors**
    $\mathcal{W}^{n+1}_{rel} \triangleleft [y, y]$; $T^{n+1} \triangleleft [\frac{T^n[k]}{2}, \frac{T^n[k]}{2}]$
  Sinon
    $\mathcal{W}^{n+1}_{rel} \triangleleft y$; $T^{n+1} \triangleleft T^n[k]$
  Fin Si
Fin Pour
$n_e \leftarrow longueur(T^{n+1})$
$\mathcal{W}_{bin}[j][k] \leftarrow \mathbf{0}, \forall j \in \overline{1, n_d}, \ k \in \overline{1, n_e}$
Pour $j = 1$ **à** n_d **Faire**
  $m \leftarrow m^j_{pre}$, $D_s \leftarrow T^j_{pre}$
  $k \leftarrow 0$
  Tant que $D_s < T^j_{min}$ **ET** $k < n_e - 1$ **Faire**
    $k \leftarrow k + 1$, $D_s \leftarrow D_s + T_k$
    $\omega^{jm}_{\mathrm{bin,k}} \leftarrow 1$
  Fin Tant que
  Tant que $k < n_e$ **Faire**
    $k_{\mathrm{next}} \leftarrow k + 1$
    $D_s \leftarrow T_{k_{\mathrm{next}}}$
    Tant que $D_s < T^j_{min}$ **ET** $k_{\mathrm{next}} < n_e$ **Faire**
      $D_s \leftarrow D_s + T_{k_{\mathrm{next}}}$
      $k_{\mathrm{next}} \leftarrow k_{\mathrm{next}} + 1$
    Fin Tant que
    Si $n_\omega > 2$ **alors**
      $q^i \leftarrow \sum_{n=1}^{k_{\mathrm{next}}} \omega^{ji}_{\mathrm{rel,n}} T_n - \sum_{n=1}^{k} \omega^{ji}_{\mathrm{bin,n}} T_n, \quad \forall i \in \overline{1, n^j_\omega}$
      $m_{new} \leftarrow index(\{q^i \geq q^l \quad \forall l \neq i\}, i)$
    Sinon
      $q \leftarrow \sum_{n=1}^{k_{\mathrm{next}}} \omega^{j}_{\mathrm{rel,n}} T_n - \sum_{n=1}^{k} \omega^{j}_{\mathrm{bin,n}} T_n$
      $m_{new} \leftarrow 1$ **Si** $q \geq 0.5 D_s$ **Sinon** 0
    Fin Si
    Si $m_{new} \neq m$ **alors**
      $m \leftarrow m_{new}$
      $\omega^{jm}_{\mathrm{bin,n}} \leftarrow 1$ **Si** $n_\omega > 2$ **Sinon** $\omega^{j}_{\mathrm{bin,n}} \leftarrow m$, $\forall n \in \overline{k, k_{\mathrm{next}}}$
      $k \leftarrow k_{\mathrm{next}}$
    Sinon
      $\omega^{jm}_{\mathrm{bin,k}} \leftarrow 1$ **Si** $n_\omega > 2$ **Sinon** $\omega^{j}_{\mathrm{bin,k}} \leftarrow m$
      $k \leftarrow k + 1$
    Fin Si
  Fin Tant que
Fin Pour
Retourner $\mathcal{W}_{\mathrm{bin}} = \left[\omega^{ji}_{\mathrm{bin,k}} \ / \ \omega^{j}_{\mathrm{bin,k}}\right], \quad T^{n+1}$

La méthode est illustrée dans la Figure 3.8 pour une disjonction de type ON/OFF. Le graphique du haut montre la trajectoire de la commande binaire relaxée (bleu) et celle de la trajectoire d'état (orange). La trajectoire en trait plein représente la trajectoire prédite par l'optimisation et celle en pointillés la trajectoire réelle du système obtenue par simulation (l'illustration est quelque peu inexacte puisque la trajectoire x^*_{simu} donne une mesure de l'erreur globale alors que la mesure qui est utilisée pour le raffinement de la grille s'appuie sur l'erreur locale). Le résultat du raffinement de la grille courante selon les critères évoqués précédemment (split/keep) est présenté dans le deuxième schéma.

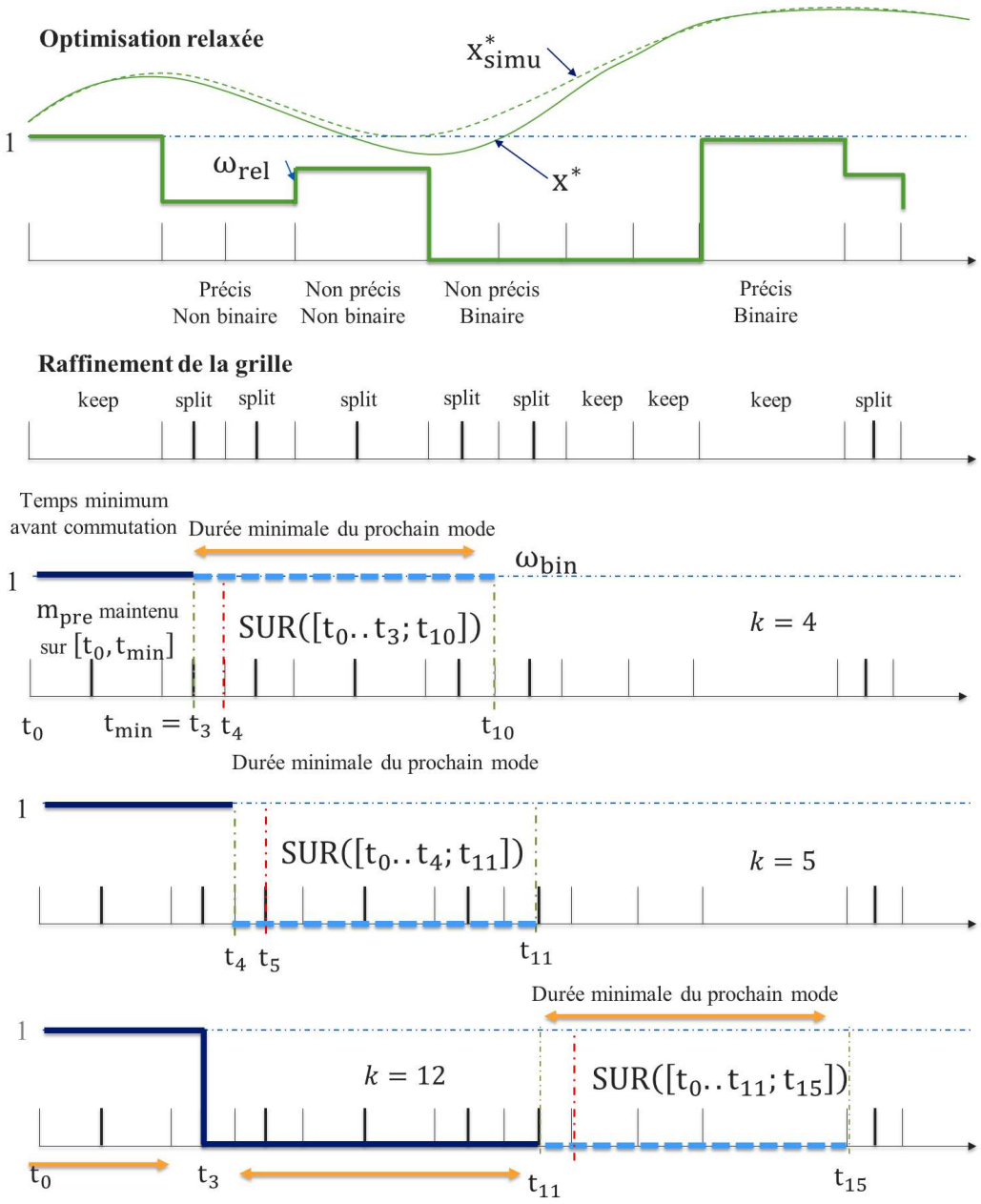


FIGURE 3.8 – Raffinement de la grille et génération de trajectoires binaires

Les graphes 3, 4 et 5 de la Figure 3.8 illustrent l'algorithme SURminT prenant en compte la durée minimale des modes. On suppose que le système est dans l'état ON avant t_0, mais n'a pas encore atteint la durée minimale T_{min}. Pour cette raison, la commutation n'est pas permise avant t_2 et le mode ON est maintenu jusqu'à cet instant (graphique du haut). A partir de cet instant, la commutation est possible et le prochain mode est calculé en s'intéressant à la solution relaxée sur $[t_k; t_s]$. Sur le deuxième graphique, le mode ON est encore sélectionné, et l'algorithme progresse depuis t_{k+1}. Sur le dernier graphique, on observe que la commutation a eu lieu. Le mode OFF est alors maintenu sur une durée suffisante pour garantir le respect de T_{min}, puis la procédure est réitérée.

3.4.4 Algorithme d'optimisation hybride

Dans cette partie, on présente l'algorithme itératif d'optimisation dynamique hybride, qui fait intervenir les différentes briques que l'on vient d'introduire : la méthode de collocation, l'estimation de l'erreur sur la grille courante, et la méthode SURminT qui sert à générer des trajectoires binaires et préparer la grille suivante. Cet algorithme est représenté dans la Figure 3.9. On initialise l'algorithme avec une grille temporelle T_0^0 et n_c points de collocation par élément. Puis les étapes suivantes sont répétées à chaque itération n jusqu'à atteindre l'un des critères d'arrêt de l'algorithme :

1. Le problème de commande optimale hybride relaxé est résolu. On obtient les trajectoires prédites par la méthode de collocation pour l'état x_{rel}^{*n} et les commandes ω_{rel}^{*n} et u_{rel}^{*n}
2. On simule le système avec les commandes ω_{rel}^{*n} et u_{rel}^{*n}. On obtient la trajectoire précise $x_{rel,sim}^{*n}$ pour les états ainsi que la valeur de la fonction objectif J_{rel}^n
3. On calcule l'erreur locale ϵ sur la grille de collocation T_0^n en utilisant la définition de l'erreur (3.44) ou (3.47)
4. On applique l'Algorithme 6 (SURminT) de raffinement de la grille pour T_0^n, ω_{rel}^{*n} , ϵ. On obtient la nouvelle grille de collocation T_0^{n+1} ainsi que les trajectoires binaires ω_{bin}^{n+1} sur cette grille.
5. Le problème de commande optimale où l'on a fixé les trajectoires $\omega_{bin}^{n+1}(t)$ est résolu sur la grille T_0^{n+1} . On obtient les trajectoires prédites par la méthode de collocation pour l'état x_{bin}^{*n} et les commandes continues u_{bin}^{*n}. Ces trajectoires servent pour l'initialisation du problème d'optimisation à l'étape $n+1$.
6. On simule le système avec les commandes ω_{bin}^n et u_{bin}^{*n}. On obtient la trajectoire précise $x_{bin,sim}^{*n}$ pour les états ainsi que la valeur de la fonction objectif J_{bin}^n
7. On incrémente l'itération courante : $n \leftarrow n+1$
8. On teste si l'un des critères d'arrêt défini ci-dessous est vérifié, auquel cas on sort de la boucle. Sinon la procédure est réitérée.

Les critères d'arrêt sont les suivants :

- la solution relaxée a un comportement binaire sur toute la grille avec une tolérance tol_b sur le caractère binaire et la précision relative des solutions est sous une tolérance tol_ϵ
- l'écart entre la fonction objectif relaxée et binaire est en dessous d'une certaine tolérance tol_J et la précision relative des solutions relaxées est sous une tolérance tol_ϵ
- le raffinement de la grille est trop important : soit le nombre d'éléments est trop grand ($n_e > n_{e,max}$), soit la longueur du plus petit élément est inférieure à une limite T_ϵ ($\min_k T_k < T_\epsilon$). On peut conserver seulement la première condition si l'on impose qu'un élément ne peut être scindé lorsque $T_k/2 < T_\epsilon$)

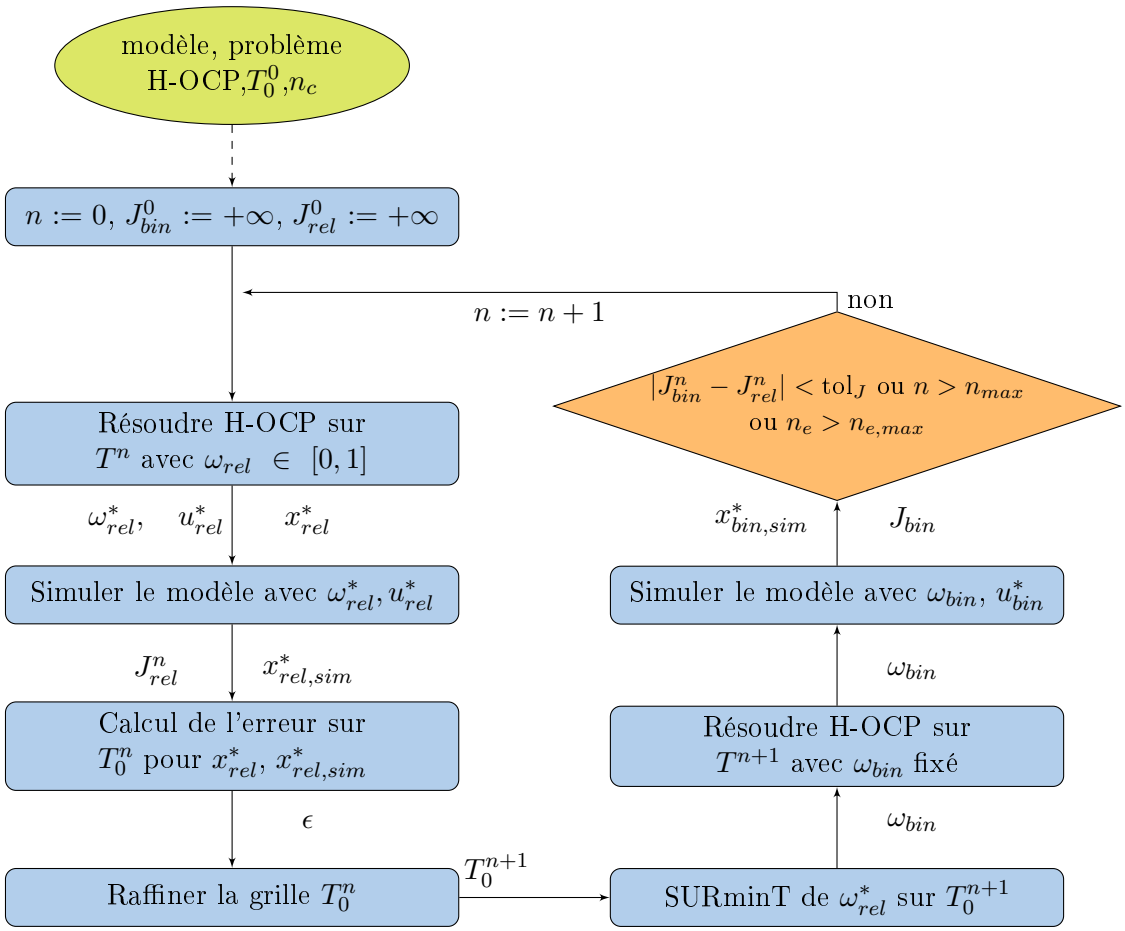


FIGURE 3.9 – Algorithme d'optimisation dynamique hybride

– un nombre maximal d'itérations a été atteint ($n > n_{max}$)

Ces critères permettent à l'algorithme de terminer en un nombre fini d'itérations. Il est important de noter que la procédure d'arrondi modifie la solution optimale du problème relaxé. Il est ainsi possible qu'après arrondi, les commandes continues u_{rel}^{*n} amènent à des violations de contraintes sur les états. C'est pour cette raison que l'on procède à une ré-optimisation des commandes continues après avoir fixé les variables binaires. Cette méthode permet de garantir à nouveau le respect des contraintes si une solution existe, mais ne garantit pas l'existence de cette solution. Cependant, on peut s'attendre à ce que la probabilité d'avoir un problème relaxé faisable et un problème arrondi infaisable diminue avec le pas de temps choisi, le Théorème 3.4 assurant que l'écart entre les états pour le problème relaxé et arrondi soit borné par une borne dépendant linéairement de la durée du plus grand pas de temps.

3.4.5 Prise en compte des coûts de transition et de modulation

Il peut être intéressant dans certaines situations de prendre en compte des coûts de transition lors du passage d'un mode à l'autre. Dans le cas des centrales d'énergie, les coûts de démarrage à froid d'un composant en sont un exemple typique ; ils correspondent souvent au surcout de maintenance associé aux phases transitoires. De même, des coûts de modulation

à la hausse ou à la baisse peuvent parfois être considérés pour certaines commandes de l'installation.
Dans cette partie, on présente une méthode pour reformuler de façon continue la variation à la hausse ou à la baisse de certaines des commandes bloquées (commandes constantes par morceaux) du système. La reformulation est intégrée lors de la génération du NLP par la méthode de collocation (ce qui nécessite de lui apporter quelques modifications alors que les algorithmes précédents lui étaient extérieurs). Il est alors possible de comptabiliser les coûts associés à ces variations dans la fonction objectif. Dans le cas particulier des commandes ω, les modulations autorisées sont le passage de 1 à 0 ou de 0 à 1 uniquement. En constatant que dans la forme conjonctive, l'état d'un composant j peut être décrit par une variable ω^j, il est alors possible de définir un coût d'arrêt c_{down} pour le premier cas (passage de 1 à 0) et un coût de démarrage c_{up} pour le second (passage de 0 à 1). Un démarrage à un instant t_k de la grille correspond alors à $\{\omega^j_{k-1} = 0; \omega^j_k = 1\}$, ou encore $\Delta\omega^j_k = 1$ avec $\Delta\omega^j_k = \omega^j_k - \omega^j_{k-1}$. Dans la suite de ce paragraphe, on s'intéresse uniquement au cas des commandes binaires ω, *i.e.* aux coûts de démarrage/ arrêt mais le même raisonnement peut s'appliquer aux commandes continues bloquées u^b. La méthode est ici présentée pour les coûts de démarrage (variations à la hausse), l'adaptation aux coûts d'arrêts se déduit facilement. On introduit pour chaque variable ω^j associée à un coût de démarrage des variables up^j_k définies de la façon suivante :

$$\mathrm{up}^j_k = \max(\Delta\omega^j_k; 0), \forall k \in \overline{1..n_e} \tag{3.50}$$

La fonction objectif J, que l'on présente ici sous forme intégrale est alors modifiée de la façon suivante :

$$\begin{aligned} J_{up} &= J + \sum_{j=1}^{n_d} c^j_{up} \sum_{k=1}^{n_e} \mathrm{up}^j_k \\ &= \underbrace{\int_{t_0}^{t_f} L(x,u)dt}_{\text{Terme intégral}} + \underbrace{\sum_{j=1}^{n_d} c^j_{up} \sum_{k=1}^{n_e} \mathrm{up}^j_k}_{\text{Somme}} \end{aligned} \tag{3.51}$$

Cette formulation est exacte si $\omega^j(t_k) \in \{0; 1\}$. A ce stade, deux remarques importantes doivent être formulées :

- La méthode SUR donne une borne sur l'écart maximal entre les états x_{bin} du problème original hybride et les états x_{rel} du problème relaxé. Cette propriété s'applique aussi à la valeur de la fonction objectif : la borne sur l'écart entre J_{bin} et J_{rel} est due au fait que J peut s'exprimer comme la valeur finale d'un état x_J défini par l'équation différentielle $\dot{x}_J(t) = L(x(t), u(t))$. Malheureusement, la fonction objectif modifiée J_{up} ne peut plus être mise sous cette forme en raison de la présence du terme *Somme* dans (3.51). Les méthodes SUR et SURminT ne sont donc plus adaptées pour générer les trajectoires binaires en présence de coûts de démarrage.
- La fonction max utilisée dans 3.50 n'est pas différentiable et ne peut être traitée avec un solveur NLP standard.

En réponse à ces deux problématiques, deux solutions sont apportées. La méthode proposée ne présente pas les garanties théoriques de convergence de la méthode SUR mais a fourni des résultats convenables dans la pratique. Concernant la génération de trajectoires binaires, on propose tout d'abord d'utiliser une méthode d'arrondi simple avec respect des temps minimaux à la place de la méthode SURminT pour les variables ω présentant des coûts

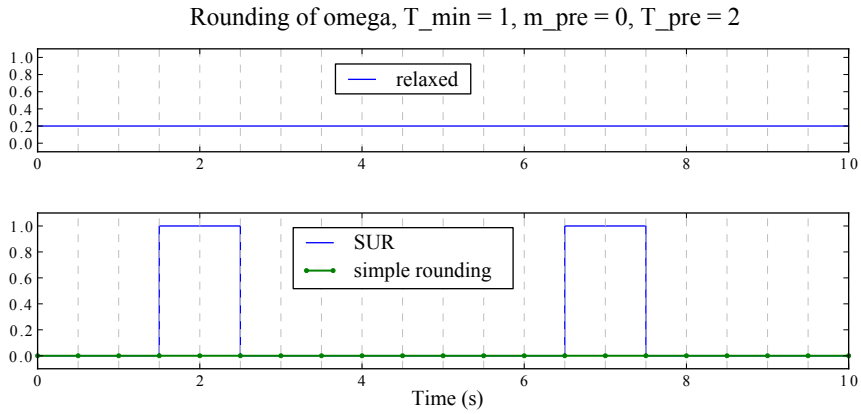


FIGURE 3.10 – Comparaison de l'arrondi donné par la méthode SURminT et un arrondi standard

de démarrage lorsque ceux-ci ont une importance significative dans la fonction objectif. Cette solution recherche la proximité instantanée entre les trajectoires $\omega_{rel}(t)$ et $\omega_{bin}(t)$ plutôt que la proximité de leurs intégrales et évite les commutations induites par la méthode SUR. La Figure 3.10 illustre les problèmes pouvant se produire sinon : ici, la solution ω_{rel} de l'optimisation relaxée avec une valeur initiale imposée à 0 est identiquement à 0.2 (graphique du haut), ce qui correspond à un coût de variation à la hausse de $0,2 * c_{up}$ sur tout l'horizon, alors que l'application de la méthode SURminT génère un coût de variation à la hausse de $2 * c_{up}$ (deux démarrages), ce qui est problématique si ce terme est prépondérant sur le terme intégral. Dans l'exemple, l'arrondi simple produit quant à lui une solution identiquement égale à 0, correspondant à un coût de démarrage nul. La méthode d'arrondi simple avec respect des temps minimaux consiste à remplacer la ligne 4 de l'Algorithme 5 par :

$$q_s^i \leftarrow \sum_{n=1}^{s_l} \omega_{rel,n}^i T_n - \sum_{n=1}^{s_{l-1}} \omega_{rel,n}^i T_n, \quad \forall i \in \overline{1..n_\omega}$$

En ce qui concerne la non-différentiabilité du terme (3.50), on propose de ré-exprimer la fonction max sous forme d'un problème MPCC, s'inspirant de la démarche proposée par [32]. Le principe de cette méthode est de reformuler cette fonction comme le résultat d'un problème d'optimisation sous contrainte. En effet, on a :

$$\mathrm{up}_k^j = \Delta\omega_k^j - y\Delta\omega_k^j \tag{3.52}$$

$$y = arg\left\{\min_{\hat{y}} \hat{y}\Delta\omega_k^j, \quad \text{s.c.} \quad 0 \leq \hat{y} \leq 1\right\} \tag{3.53}$$

Les conditions d'optimalité du problème (3.53) sont alors utilisées comme des contraintes pour (3.52). Le Lagrangien de (3.53) fait intervenir de nouvelles variables duales que l'on nomme a_k^j et b_k^j :

$$\mathcal{L}(y, a^j, b^j) = y\Delta\omega_k^j + ya^j + (1-y)b^j$$

Les conditions d'optimalité s'écrivent alors :

$$\begin{aligned} \Delta\omega_k^j - a_k^j + b_k^j &= 0 \\ 0 \leq a_k^j \perp y &\geq 0 \\ 0 \leq b_k^j \perp (1-y) &\geq 0 \end{aligned}$$

Après simplifications de y, on obtient le jeu d'équations suivant pour déterminer les variables

$$\mathrm{up}_k^j = \Delta\omega_k^j + b_k^j \tag{3.54}$$

$$\Delta\omega_k^j = a_k^j - b_k^j \tag{3.55}$$

$$0 \leq a_k^j \perp b_k^j \geq 0 \tag{3.56}$$

La contrainte (3.56) est une contrainte de complémentarité. La notation $\perp$ signifie que les variables a_k^j et b_k^j ne peuvent être toutes les deux strictement positives (elles peuvent en revanche être toutes les deux nulles). Cette contrainte ne peut être implémentée directement dans IPOPT en raison des problèmes numériques déjà évoqués (elle pourrait l'être uniquement dans une version dérivée nommée IPOPT-C, mais qui n'est pas disponible depuis JModelica). On propose donc de la traiter par pénalisation dans la fonction objectif. Pour synthétiser, le problème d'optimisation original est modifié en y ajoutant les contraintes (3.54- 3.55) et en considérant la fonction objectif suivante.

$$\tilde{J} = J_{up} + \rho \sum_{j=1}^{n_d} \sum_{k=1}^{n_e} a_k^j b_k^j \tag{3.57}$$

Avec ρ une constante positive. Comme on peut le constater, cette méthode nécessite l'ajout dans la méthode de collocation de nouvelles variables, de nouvelles contraintes et modifie la fonction objectif du problème initial.

3.5 Applications de la méthode d'optimisation hybride

On s'intéresse dans cette partie à l'optimisation économique du planning journalier de la centrale de cogénération présentée au chapitre 2. Dans cette installation, les gains proviennent de la revente d'électricité au réseau, à un prix variant dans le temps $C_{elec}(t)$, montré en Figure 2.18. Les coûts concernent quant à eux l'achat de combustible, à un prix fixe C_f. Ces deux prix sont normalisés par rapport au prix maximal de l'électricité $C_{elec,max}$, et leur valeur donnée sur $[0, 1$ p.u.$]$. Un prix de 0.54 p.u. est utilisé pour le combustible. L'objectif est alors de minimiser le coût d'exploitation tout en fournissant au réseau une puissance thermique W_{net} égale à la demande W_{demand}, à une température T_{net} proche de sa référence $T_{net,ref}$. On considère la fonction objectif suivante :

$$\begin{aligned} J = \underbrace{\int_{t_0}^{t_f} \left[\left(W_{fb}(t) + W_{fe}(t) \right) C_f - W_{elec}(t) C_{elec}(t) \right] dt}_{\text{cost}} + \\ + \underbrace{\int_{t_0}^{t_f} \left[\left(\rho_W (W_{net}(t) - W_{demand}(t)) \right)^2 + \left(\rho_T (T_{net}(t) - T_{net,ref}) \right)^2 \right] dt}_{J_{pen}} \end{aligned} \tag{3.58}$$

Ici, le suivi de puissance et de température s'effectue en pénalisant l'écart à la référence. Les pénalités ρ_T et ρ_W ont été choisies après quelques essais afin d'obtenir un bon suivi et une contribution négligeable de J_{pen} dans J après convergence. La revente de chaleur n'a pas été intégrée dans la fonction objectif car la demande en chaleur doit être satisfaite exactement dans tous les cas et n'influence pas l'optimisation.
Certaines contraintes doivent être respectées dans l'installation. Pour les commandes, on impose les contraintes suivantes :

- La charge du moteur $\text{load}_e = W_{fe}/W_{fe,max} \in [0,5;1]$, **si** ON_e, $\quad 0$ **sinon**
- La charge de la chaudière $\text{load}_b = W_{fb}/W_{fb,max} \in [0,6;1]$, **si** ON_b, $\quad 0$ **sinon**
- Le débit au réseau $q_{net} > 0$
- Le débit au moteur $q_e \in [0\ kg/s, 20\ kg/s]$
- Le débit à la chaudière en marche $q_b \in [q_\epsilon\ kg/s, 15\ kg/s]$, **si** ON_b, $\quad q_\epsilon$ **sinon**
- Le débit dans la branche de régulation $q_{ret} \in [0,01; 0,2 q_{net}]$

Cependant, quelques modifications sont apportées à la modélisation des commandes pour démontrer les points évoqués en 3.4.1. Tout d'abord, on souhaite modéliser le débit q_{eng} et la charge load_e du moteur comme des signaux affines par morceaux. On substitue pour cela ces entrées u avec leur dérivée $deru$, puis l'on utilise les facteurs bloquants sur $deru$. Les autres entrées ont été modélisées par des signaux constants par morceaux en utilisant les facteurs bloquants également. Des polynômes d'ordre supérieur n'ont pas été utilisés pour les entrées en raison des oscillations parfois observées (qui peuvent amener à des violations de contraintes entre les points de collocation).
Un autre commentaire sur la modélisation des commandes concerne le débit q_{ret} dans la branche de régulation : celui-ci est une variable d'ajustement pour la température au réseau, et ne doit pas représenter une fraction trop importante du débit q_{net}. C'est pour cela que la contrainte sur q_{ret} n'est pas donnée en valeur absolue, mais comme une fraction de q_{net} (ici limitée à 20% de celui-ci). q_{ret} est donc remplacée par une nouvelle commande $\text{frac}_{ret} \in [0,01; 0,2]$ dans le modèle :

$$\text{frac}_{ret} \in [0; 0,2], \qquad \text{avec } \text{frac}_{ret} = \frac{q_{ret}}{q_{net}}$$

Si l'on avait utilisé directement q_{ret} comme variable de commande, il aurait fallu ajouter la contrainte $0 \leq q_{ret} \leq 0,2 q_{net}$ au problème d'optimisation. Ce choix n'a pas été retenu car l'algorithme de points intérieurs utilisé traite de façon plus efficace les contraintes de bornes sur les variables que les autres contraintes.
Des contraintes sur les températures de l'eau dans l'installation sont également présentes afin d'éviter les phénomènes d'ébullition. On impose par ailleurs que l'état de charge en fin de journée soit supérieur ou égal à celui en début de journée :

$$\text{SOC}(t_f) \geq \text{SOC}(t_0)$$

Enfin, des contraintes hybrides sur les temps minimum de marche sont ajoutées dans la définition du problème : une heure pour le moteur, une demi-heure pour la chaudière. L'état de la chaudière et du moteur avant t_0 sont supposés inconnus pour l'instant. La méthode est initialisée avec $n_e = 12$ éléments et $n_c = 3$ points de collocation par élément.
La Figure 3.11 montre le résultat de l'optimisation pour l'état de charge du stockage et les puissances moteur et chaudière à l'itération 1 et 3. On peut lire dans le titre d'une part la valeur de la fonction objectif (Obj. IPOPT) qui inclut le terme de pénalisation $Jpen$, d'autre

part l'objectif économique (cost) seul, pour le problème relaxé (sur la grille $\mathcal{T}_0^n$) et pour le problème binaire (sur la grille $\mathcal{T}_0^{n+1}$). On constate que le moteur est mis en marche toute la journée alors que la chaudière connaît des démarrages et arrêts. Pour la deuxième et dernière itération, le profil de température dans le stockage ainsi que la température envoyée au réseau sont montrés en Figure 3.13, les commandes ω et la courbe de prix en Figure 3.12.

3.5.1 Comparaison de la méthode de collocation hybride avec la PLNE

Un modèle de l'installation en PLNE, calibré sur les mêmes données que le modèle non linéaire a été réalisé pour comparer les résultats des deux méthodes. Ce modèle a été réalisé avec le logiciel EDF PILOT et est représenté en Figure 3.14. Il s'agit d'un modèle en énergie. Les rendements du moteur et de la chaudière ont été définis comme constants (une modélisation constante par morceaux est également possible), alors qu'ils sont définis par un polynôme dans le modèle non linéaire. Le problème est résolu sur une journée au pas de la demi-heure en prenant pour chaque pas de temps la valeur moyenne de la demande sur cette période. On a ainsi $N = 48$ pas de temps. On demande à ce que la demande soit satisfaite exactement par l'ajout d'une contrainte égalité. Aucune contrainte sur les températures ne peut être ici ajoutée car celles-ci ne sont pas modélisées dans cette formulation. Le problème d'optimisation équivalent est le suivant :

$$J_{lin} = \sum_{k=1}^{N} (W_{fb,k} + W_{fe,k}) C_f - W_{elec,k} C_{elec,k}$$

On souhaite maintenant comparer les résultats des deux méthodes, et l'on utilise pour cela le modèle non linéaire, le plus détaillé comme modèle de validation. Un problème qui se pose pour cette comparaison est que le modèle de PNLE possède des commandes qui ne correspondent pas aux commandes du système réel. Dans le cas de la centrale de cogénération, les commandes sont ainsi :

- La puissance de combustible au moteur W_{fe} que l'on convertit en charge load_e
- La puissance de combustible à la chaudière W_{fb} que l'on convertit en charge load_b

Il est donc nécessaire dans un premier temps de générer des trajectoires pour les autres commandes (débits). Cela peut être fait de plusieurs façons :

1. En intégrant dans le modèle des régulations pour les commandes inconnues de la forme $u(t) = c(u^*(t), y^*(t))$. Ces régulations seront probablement sous optimales mais devront permettre néanmoins le respect des contraintes. Dans la pratique, cette stratégie est très dépendante de l'installation considérée et nécessite une connaissance métier qui ne fait pas l'objet de cette thèse.
2. En utilisant un observateur à entrées inconnues. On estime ici les trajectoires de l'état $x(t)$ et de la commande $u(t)$ d'après les valeurs des sorties $y^*(t)$ données par le modèle simplifié.
3. En posant un problème inverse sur les trajectoires du modèle non linéaire. On résout alors un nouveau problème de commande optimale sur le modèle non linéaire où l'objectif est de minimiser la distance entre les trajectoires des variables $[u^*(t), y^*(t)]$ communes aux deux modèles (ici les trajectoires en puissance et énergie). Cette stratégie se pose sous forme d'un problème d'optimisation et permet de prendre en compte les contraintes du modèle non linéaire.

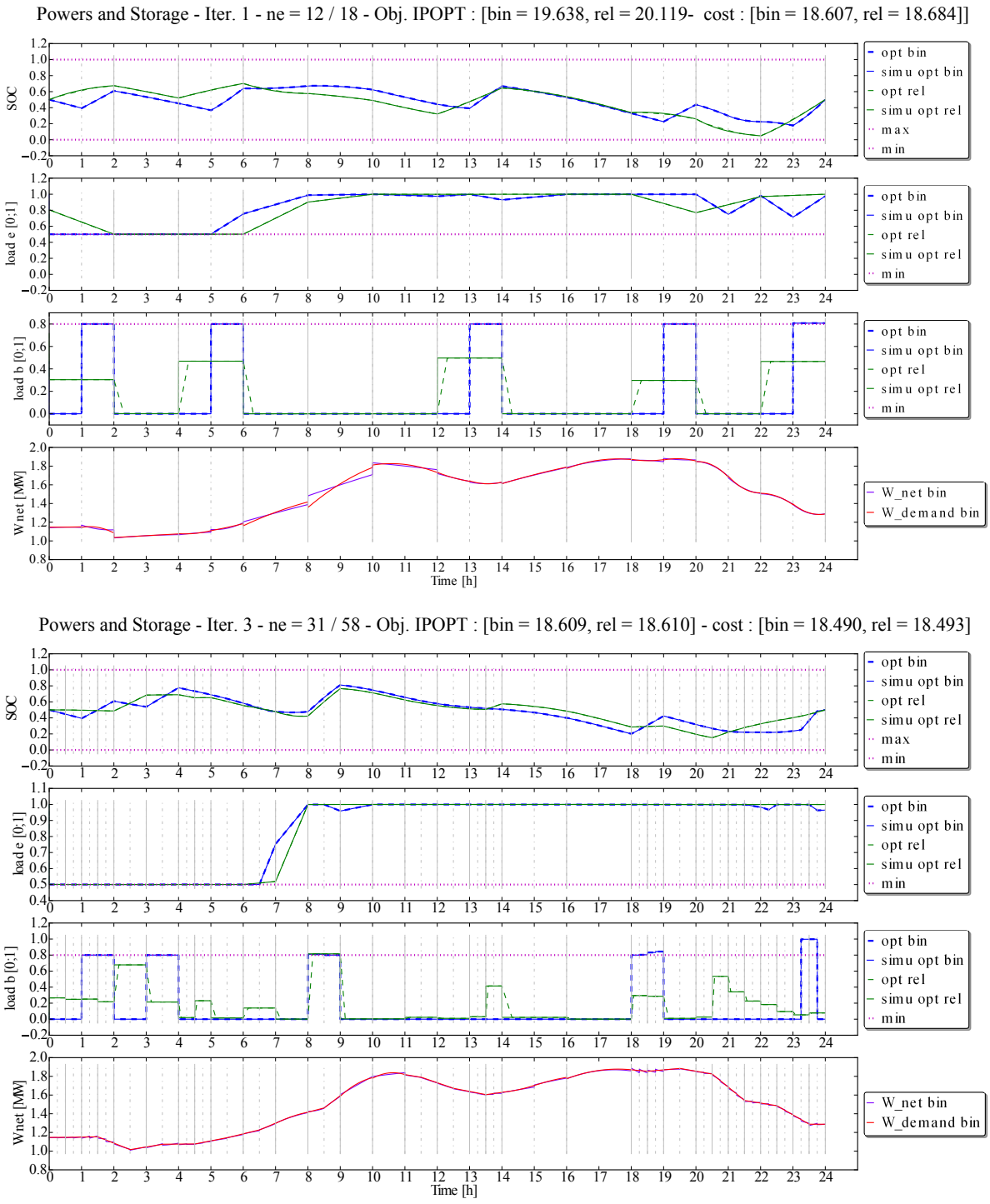


FIGURE 3.11 – Optimisation de la cogénération avec la méthode de collocation hybride - Puissances et énergie à l'itération 1 (haut) et 3 (bas)

La première méthode, qui requiert la synthèse de lois de commande admissible s'est avérée plus ardue que prévu. La difficulté se situe dans la garantie du respect simultané de toutes

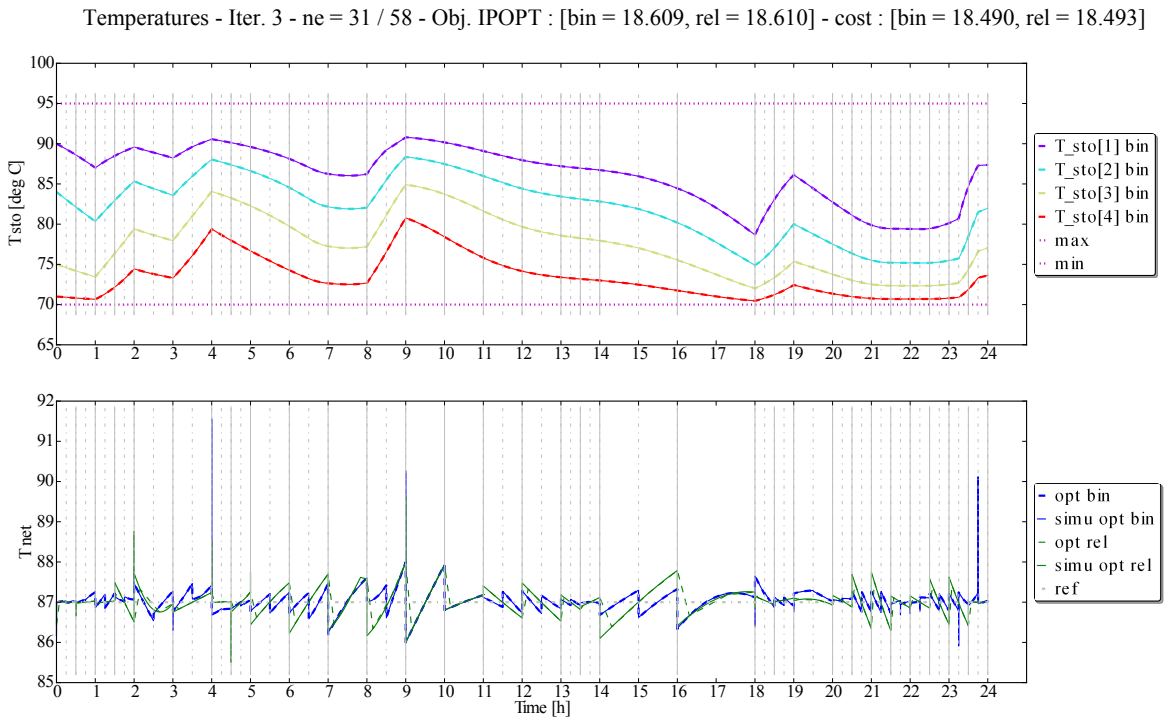


FIGURE 3.12 – Optimisation de la cogénération avec la méthode de collocation hybride - Températures

les contraintes de l'installation. Le respect de ces contraintes est en effet un pré-requis pour pouvoir comparer deux solutions d'optimisation (la fonction objectif valant en théorie l'infini si une contrainte n'est pas respectée). On ne retiendra donc pas cette approche, ni la mise en place d'un observateur à entrées inconnues qui pourrait faire l'objet de futurs développements. On se concentre donc sur la dernière solution qui nous servira à établir une comparaison entre la méthode de collocation hybride et la PLNE. On retient comme variables communes au modèle non linéaire et au modèle de PLNE les variables suivantes : $V = [W_{hb}, W_{he}, \mathrm{SOC}, T_{net}, W_{net}]$. T_{net} n'est pas à proprement parler une variable du modèle de PLNE mais doit suivre une référence constante. Par ailleurs les décisions d'engagement (ON/OFF) de la chaudière et du moteur ne sont pas remises en cause. Le critère quadratique suivant est retenu :

$$J_{ident} = \int_{t_0}^{t_f} [(V(t) - V_{ref}(t))^T Q (V(t) - V_{ref})] dt$$

Avec $Q = diag(1e-7, 2e-7, 3e3, 1e3, 1e-4)$. Les contraintes sur les variables continues sont les mêmes que précédemment. Le résultat de cette méthode est illustré en Figure 3.15. La méthode permet de trouver des commandes admissibles (figure de droite) tout en gardant une grande similitude pour les sorties (figure de gauche).
La comparaison des deux méthodes peut maintenant être faite avec toutefois certaines réserves. En particulier l'identification des trajectoires pour les entrées inconnues se fait sans prendre en compte la fonction objectif (3.58), et en particulier le terme de pénalité. Les commandes identifiées sont également influencées par les pondérations appliquées sur les différentes sorties lors de l'identification (matrice Q). Pour ces raisons, le coût J_{lin} prédit par la

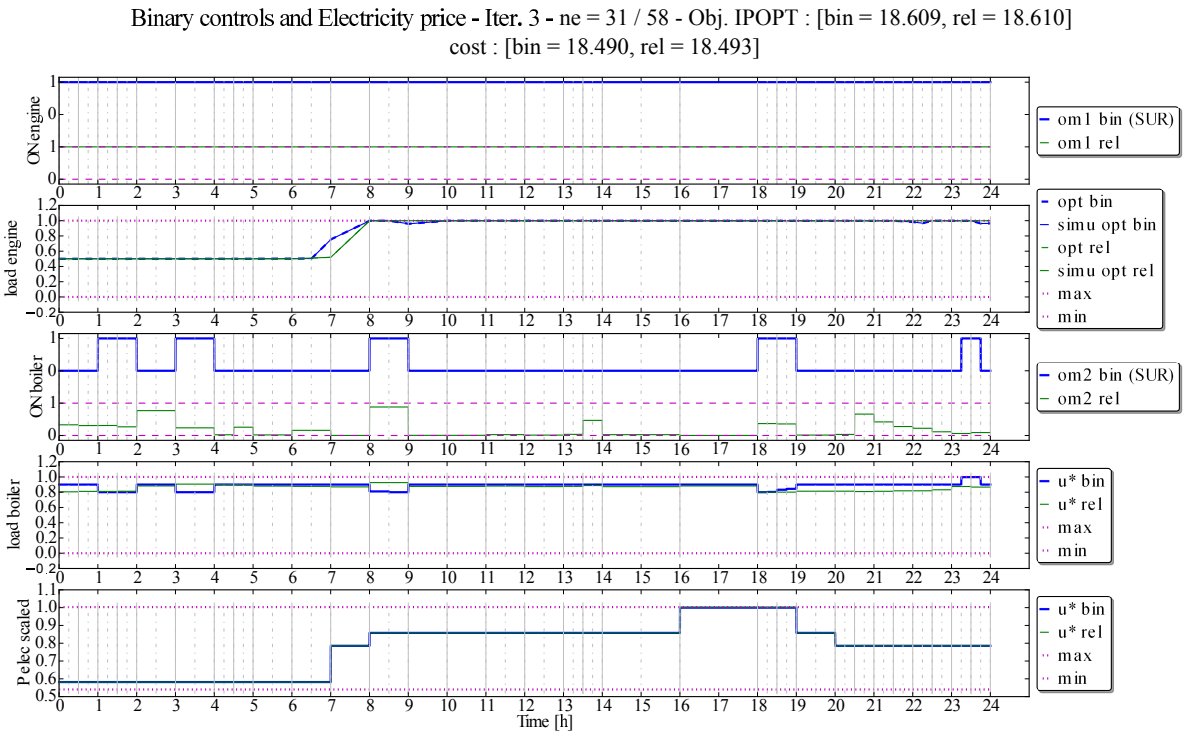


FIGURE 3.13 – Optimisation de la cogénération avec la méthode de collocation hybride - Modes et prix de l'électricité

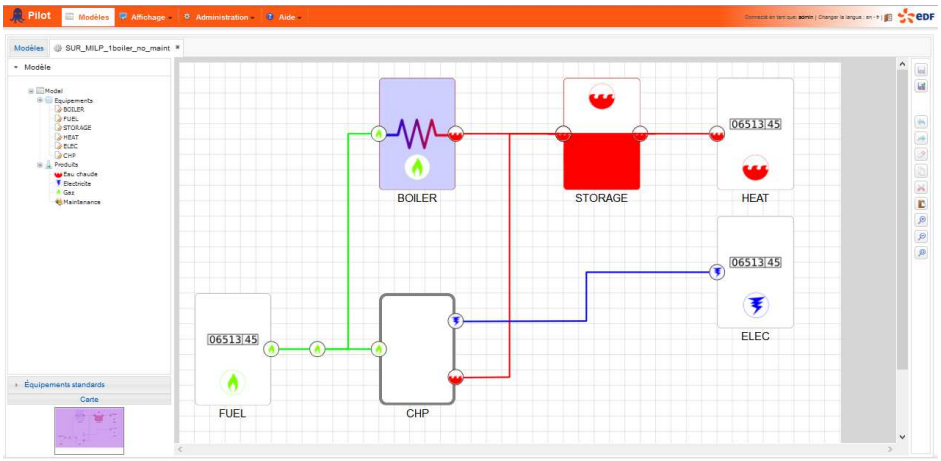


FIGURE 3.14 – Modèle PILOT (PLNE) de l'installation de cogénération

PLNE (18,332) diffère légèrement du coût J correspondant à la trajectoire identifiée (18,206).

La comparaison des deux méthodes est donnée en Figure 3.16. Les résultats obtenus pour la méthode PLNE avec identification des entrées inconnues sont en trait pointillé bleu (MILP) et ceux obtenus par la méthode de collocation hybride sont en trait plein violet (SURminT). On constate que les deux méthodes fournissent des plannings d'engagement différents mais

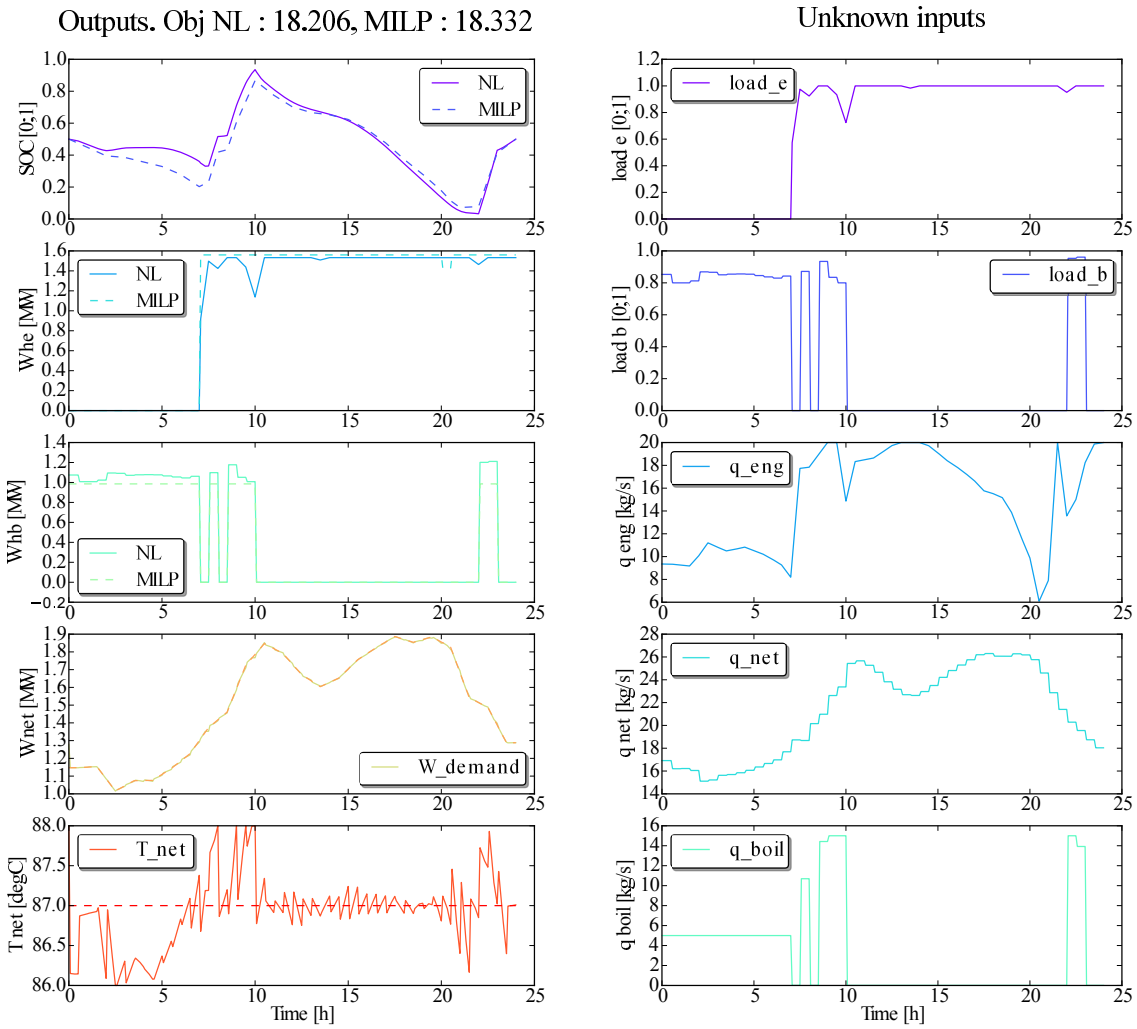


FIGURE 3.15 – Identification des variables du modèle non linéaire avec celles du modèle PLNE (gauche) - Estimation des entrées inconnues (droite)

une valeur similaire pour la fonction objectif.

En conclusion, les deux méthodes donnent des résultats assez similaires sur ce modèle en ce qui concerne la valeur de la fonction objectif même si les trajectoires de l'installation diffèrent. La méthode de collocation hybride a permis de traiter correctement les aspects logiques d'engagement des unités tout comme la PLNE. On note cependant que dans le cas de la PLNE, les résultats présentés ont nécessité la résolution supplémentaire d'un problème d'optimisation non linéaire afin de reconstruire les trajectoires des commandes manquantes (ici les débits), faute de quoi aucune garantie n'était donnée sur le respect des contraintes physiques dans le modèle non linéaire.
Par ailleurs, dans ce cas d'étude, l'hypothèse de propriétés constantes de l'eau permet de réduire fortement la non linéarité du modèle, et explique que l'on arrive à trouver une aussi bonne concordance entre le modèle non linéaire et le modèle de PLNE. Malheureusement,

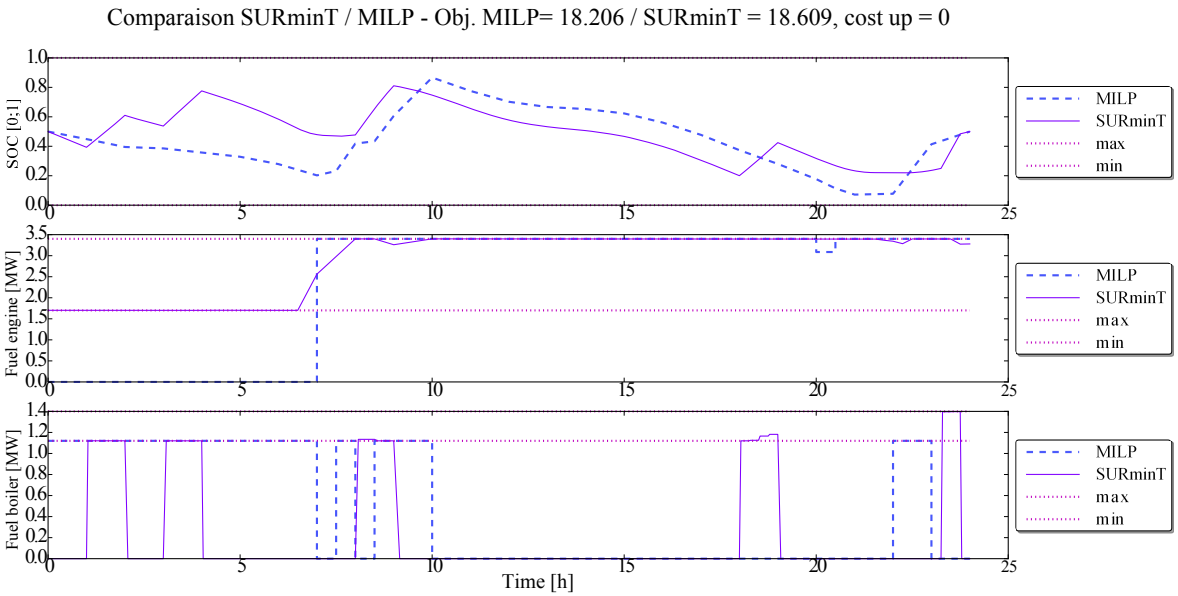


FIGURE 3.16 – Comparaison de la méthode de collocation hybride avec la PLNE - stock, puissance combustible à la chaudière et au moteur

cette hypothèse ne peut plus être formulée lorsqu'on considère des centrales utilisant de l'eau/vapeur comme fluide de travail. L'optimisation sur le modèle physique prend alors toute son importance. Un tel exemple a été traité et est présenté en Annexe F. On s'intéresse maintenant à un autre aspect de la méthode proposée : la prise en compte de coûts de démarrage dans la fonction objectif.

3.5.2 Coûts de démarrage

On teste ici l'optimisation en présence de coûts de démarrage et l'on constate les différences avec le cas de référence (sans coût de démarrage) de la Figure 3.11. Dans ce cas de référence, on constate des démarrages et arrêts fréquents de la chaudière, mais qui respectent néanmoins la durée minimale autorisée (une demi-heure). On introduit alors pour la chaudière un coût de démarrage élevé de 3 p.u. mais sans faire de restriction sur le mode dans lequel elle se trouve avant le début de l'horizon. Les résultats sont présentés en Figure 3.17. Comme évoqué en 3.4.5, la génération de la trajectoire binaire correspondant à l'état de la chaudière est générée en utilisant un arrondi simple respectant la durée minimale des modes et non pas la méthode SURminT. On constate maintenant que la chaudière est à l'état ON sur tout l'horizon, ce qui évite de payer des coûts de démarrage. La comparaison des résultats donnés par la méthode de collocation hybride et la PLNE sur ce cas est donné en Figure 3.18. Enfin, dans le dernier cas, on impose en plus à la chaudière d'être arrêtée avant le début de l'horizon comme montré en Figure 3.19. On constate que la chaudière n'est plus jamais allumée. La conséquence est que le système ne parvient plus à satisfaire aussi bien la demande en chaleur (l'optimisation juge qu'il est préférable de payer le prix de la pénalité J_{pen} dans (3.58)).

La méthode de collocation hybride a par ailleurs été testée sur un système de plus grande taille : il s'agit d'un réseau de chaleur alimenté par une cogénération à cycle vapeur issu de

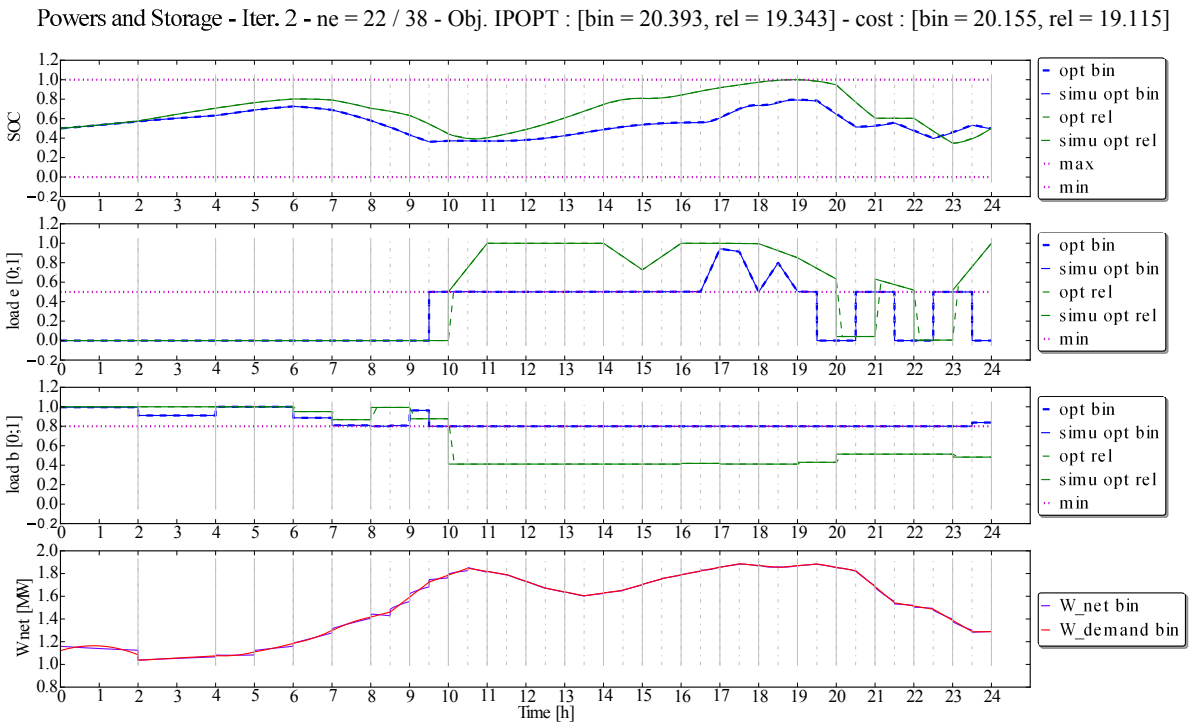


FIGURE 3.17 – Optimisation de la cogénération : ajout d'un coût de démarrage de 3 u pour la chaudière, mode initial libre

l'étude [9]. Le gain en complexité est significatif puisque le modèle en question inclut une représentation lissée des tables de l'eau/vapeur ainsi qu'une modélisation des retards dans le réseau. Ces résultats sont présentés en Annexe F.

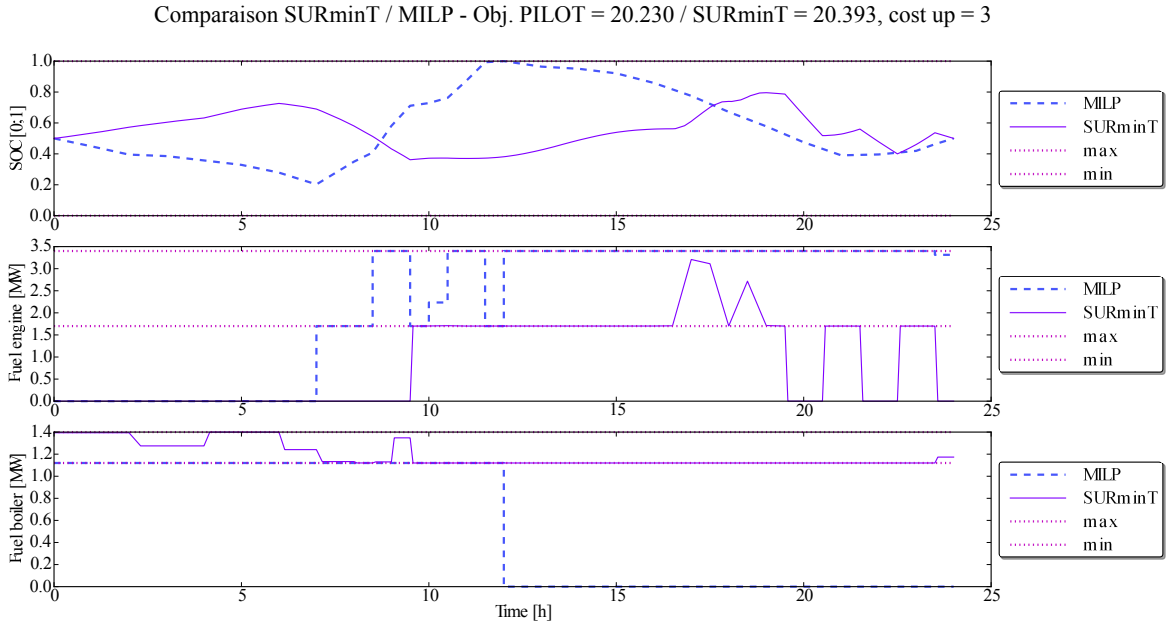


FIGURE 3.18 – Comparaison de l'optimisation de la cogénération avec un coût de démarrage de 3 u

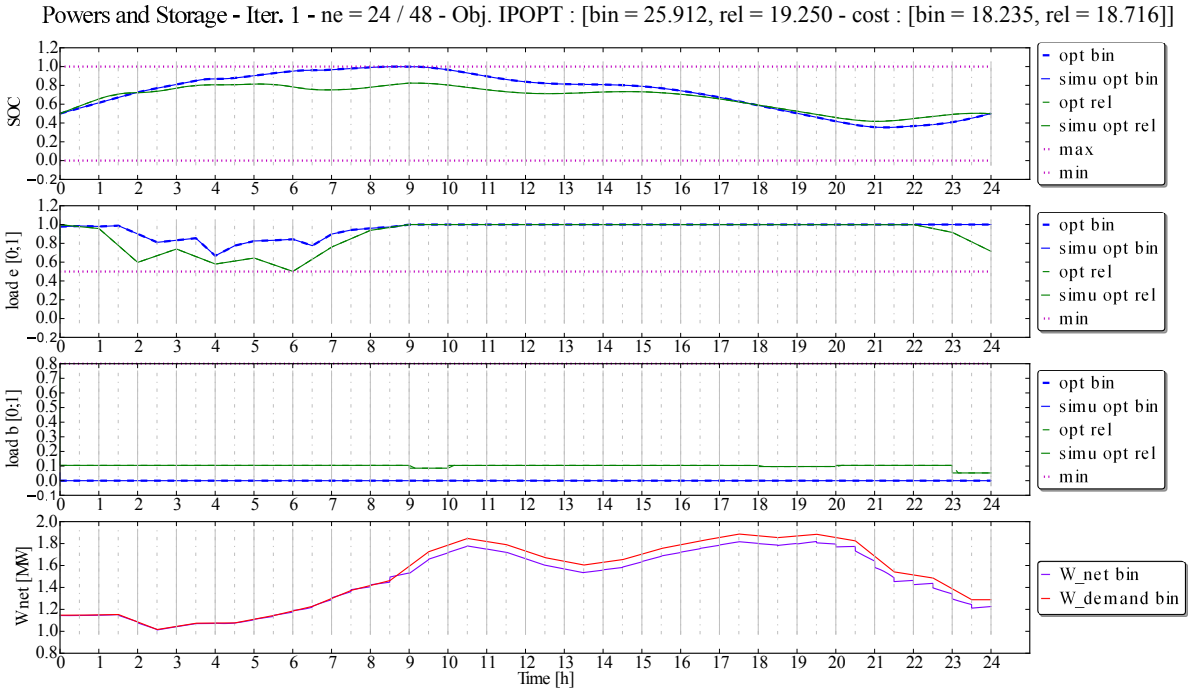


FIGURE 3.19 – Optimisation de la cogénération avec un coût de démarrage de 3 u pour la chaudière et la contrainte $\omega_b(t_0^-) = 0$

CHAPITRE 4

Commande prédictive basée sur des modèles linéarisés tangents

4.1 Suivi de trajectoires d'état par commande prédictive : cas idéal

4.1.1 Motivation

On décrit dans ce chapitre un algorithme de commande prédictive remplissant la fonction "Ajustement de trajectoire temps réel", décrite dans le Chapitre 1 et montrée en Figure 1.1. La méthode décrite au chapitre précédent, ou d'autres méthodes telles que la Programmation Linéaire en Nombres Entiers (PLNE) permettent de définir des trajectoires optimales pour le système sur un horizon long. Ces trajectoires peuvent être données pour les état x_r, les sorties y_r et les commandes u_r. Ces méthodes sont toutefois limitées à des modèles simplifiés pour des raisons de puissance de calcul ou des limitations sur les classes de systèmes considérées (par exemple l'optimisation par PLNE contraint à utiliser des modèles linéaires). Ce fait s'accompagne d'une erreur de modélisation parfois importante entre le système réel et le modèle utilisé pour l'optimisation. Il est même courant que les trajectoires fournies par le modèle d'optimisation ne définissent pas exactement de la même façon les entrées, sorties ou états du système. A titre d'exemple, un modèle de PLNE fournira couramment des trajectoires en puissance et énergie mais ne donnera pas d'information sur la trajectoire des pressions et températures dans le système. De plus, des perturbations affectent le système et doivent être prises en compte afin de maintenir une cohérence entre les trajectoires prévues et celles réalisées. On considère que le système étudié est soumis à certaines perturbations w. Un sous ensemble w^m de ces perturbations est mesuré. Une correction en temps réel de la trajectoire du système s'avère donc nécessaire pour prendre en compte les perturbations et erreurs de modélisation, et assurer le respect des contraintes sur un horizon donné. Dans cette optique de commande, il est intéressant d'utiliser le modèle physique (non linéaire) du système car il s'agit du modèle le plus détaillé dont on dispose, et qui fournit les prédictions les plus précises sur l'évolution du système.
La commande prédictive, ou MPC (*Model Predictive Control*) consiste à optimiser la séquence de commandes u à appliquer à un système en s'appuyant sur les prédictions d'un modèle de

celui-ci. La commande et les prédictions sont faites dans un cadre échantillonné. Le MPC est une technique de commande intéressante car en plus de fournir des trajectoires optimisées, il permet de prendre en compte aisément les systèmes ayant plusieurs commandes, plusieurs sorties (MIMO), et surtout les contraintes sur ces variables. Dans une commande MPC, l'optimisation est faite sur un horizon de prédiction glissant, puis réitérée à chaque pas de temps. Cependant, seul le premier terme de la séquence de commande calculée est appliquée au système car les nouvelles commandes calculées sont réputées meilleures que celles calculées à l'instant précédent. En effet, au pas de temps suivant, les nouvelles mesures issues du système pour les sorties et les perturbations mesurées (y_{mes} et w^m_{mes}) sont prises en compte, ce qui introduit un retour d'état dans la boucle de commande. Le MPC est très bien étudié dans le cadre des modèles linéaires invariants dans le temps, on citera notamment les ouvrages de référence [61],[62]. La plupart du temps, le MPC a pour objectif la stabilisation autour d'un point d'équilibre (x_s, u_s) (problème de stabilisation), ou d'une succession de points d'équilibre (problème de *tracking*). Pour cette raison, un critère quadratique à horizon fini de la forme suivante est très souvent utilisé pour la fonction objectif du MPC, avec Q et R des matrices de pondération (Q définie positive, R semi-définie positive) :

$$J(x,u) = \int [(x-x_s)^T Q(x-x_s) + (u-u_s)^T R(u-u_s)]dt$$

Plusieurs travaux étendent la commande prédictive au cadre des systèmes non linéaires tels que ceux étudiés ici (NMPC). Une bonne synthèse est donnée dans [63]. Dans ces méthodes, on cherche à résoudre des problèmes de commande optimale sur le modèle non linéaire avec un horizon glissant, en utilisant certaines des stratégies présentées au chapitre précédent (tir simple, tir multiple, collocation). Afin de tenir les contraintes du temps réel, des stratégies sont mises en place pour réduire la taille du problème ou pour exploiter au mieux sa structure [64][13]. Par ailleurs, les trajectoires calculées entre deux problèmes MPC consécutifs sont souvent proches. Cette caractéristique peut être exploitée pour initialiser le nouveau problème MPC à partir du précédent (*warm start*), comme évoqué dans [65] pour la méthode de collocation, ce qui est important pour la rapidité de convergence des algorithmes d'optimisation locale employés (SQP ou points intérieurs le plus souvent). Enfin des techniques informatiques telles que la parallélisation ou la génération efficace de code compilé peuvent être mises en place.

Dans cette thèse, on propose une autre façon d'aborder la commande MPC pour les systèmes non linéaires modélisés à l'aide de Modelica. En effet, ces modèles physiques détaillés utilisés pour la simulation sont souvent de dimension trop importante pour pouvoir servir de modèle interne à un MPC non linéaire. Ici, plutôt que de résoudre un problème de commande optimale sur le modèle non linéaire, une approche hiérarchisée est proposée. Elle est un compromis permettant de bénéficier de la précision du modèle non linéaire tout en gardant une complexité acceptable pour le problème d'optimisation : la stratégie de commande prédictive pour le suivi de trajectoires s'appuie ainsi sur des modèles linéarisés tangents issus du modèle non linéaire de la centrale. On nomme cette commande TL-MPC (*Tangent Linear MPC*). Les modèles linéarisés tangents s'expriment dans les variables $(\delta x, \delta u)$ représentant l'écart à la trajectoire nominale (qui elle s'exprime en (x, u)). Ils sont une approximation du premier ordre valable à tout instant autour de la trajectoire nominale et permettent en plus de bénéficier de certaines propriétés du MPC linéaire, notamment la grande rapidité de convergence

de l'optimisation.
L'utilisation dans le MPC d'un modèle linéarisé autour d'un point d'équilibre est une pratique assez courante [66]. En revanche, la linéarisation autour d'une trajectoire non stationnaire l'est moins. Cette piste est évoquée dans [67] pour le suivi de référence pour la direction assistée d'un véhicule mais le modèle linéarisé tangent est construit manuellement d'après un modèle ODE. On propose ici une procédure automatique pour remplir cette fonction, basée sur l'utilisation de modèles Modelica, qui sont potentiellement donnés sous forme de DAE.
On s'intéresse alors à la stabilité de la commande TL-MPC. Cependant, l'analyse de stabilité du MPC qui utilise un modèle en variations $(\delta x, \delta u)$ soulève certaines difficultés théoriques supplémentaires par rapport au MPC linéaire invariants dans le temps. D'une part on doit étudier la stabilité non plus pour un modèle $[A, B]$ mais pour une famille de modèles $[A_k, B_k]$; d'autre part le domaine admissible exprimé dans les variables $(\delta x, \delta u)$ devient lui aussi variant dans le temps, ce qui nécessite une nouvelle définition pour les contraintes et l'ensemble terminal notamment. Au vu de ces difficultés, on propose une analyse de stabilité de la commande TL-MPC sous certaines hypothèses simplificatrices. On présente ensuite l'algorithme réellement utilisé, qui montre de bonnes performances en pratique mais n'inclut pas tous les éléments du cadre théorique. Cette version du MPC contient notamment des améliorations axées sur les spécificités des centrales d'énergie : il permet une normalisation des variables, la prise en compte de contraintes hybrides de type ON/OFF et peut utiliser certaines hypothèses pour améliorer la rapidité de calcul.

4.1.2 Notations

Le MPC proposé est à temps discret au pas de temps T_s, mais les trajectoires du système sont simulées en temps continu, ce qui nécessite d'éclaircir les notations utilisées. Ainsi, pour une même variable v, v_k fait référence à la valeur de v à l'instant t_k, $v(t)$ à la valeur de v à un instant t, et v à la trajectoire de la variable v (v étant alors traité comme un signal).
Pour la valeur des variables différentielles x à l'instant t, on utilisera alternativement la notation $x(t)$, ou celle plus précise $x(x_0, u, w; t)$ signifiant que la trajectoire a été obtenue par intégration de $\dot{x}$ avec comme condition initiale x_0 et comme entrées les signaux u et w.

Définition (Trajectoire d'un système) Une trajectoire d'un système est une solution des équations différentielles qui le modélisent et est notée

$$\mathtt{P} = (x, y, u, w) = \{(x(t), y(t), u(t), w(t)), \quad t \in [t_0; t_f]\}$$

Par abus de langage, on dira que cette trajectoire est une solution du modèle.

La trajectoire de référence que l'on cherche à suivre est notée

$$\mathtt{P}_r = (x_r, y_r, u_r, w_r)$$

Les notations suivantes sont liées au caractère échantillonné et à l'horizon glissant employés dans le MPC : tout d'abord, l'exposant i est utilisé lorsque cela est nécessaire pour signifier qu'une variable a été calculée à l'instant t_i ; par exemple la valeur prédite à t_i de la variable v pour l'instant t_{i+1} s'écrira $v^i(t_{i+1})$. Par ailleurs, le MPC calcule la valeur des variables sur un horizon fini (horizon de commande ou de prédiction) aux instants d'échantillonnage. Pour chacune des variables, ces prédictions peuvent alors être représentées sous forme de vecteurs. On utilisera des lettres majuscules pour se référer à ces vecteurs. V^i contiendra ainsi

les valeurs futures v^i_{i+k} de v sur un horizon donné, calculées à l'instant t_i. Lorsqu'il n'y a pas d'ambiguité, on omettra parfois l'exposant i.

La notation Δ est finalement introduite pour décrire la variation d'une variable v entre deux instants d'échantillonnage : $\Delta v_k = v_k - v_{k-1}$.

4.1.3 Modèles considérés

Plusieurs modèles vont être considérés par la suite et nécessitent d'être bien définis pour l'analyse de stabilité qui va suivre.

4.1.3.1 Système physique

On suppose que le système physique est décrit par un système d'équations algébro-différentielles, qui après traitement symbolique par les méthodes présentées au Chapitre 2 peut être mis sous forme de modèle d'état (à la seule différence qu'il existe maintenant plusieurs types d'entrées : u et w_p) :

$$\left\{ \begin{array}{rcl} \dot{x}_p(t) & = & f_p(x_p(x_{p0}, u, w_p; t), u(t), w_p(t)) \\ y_p(t) & = & h_p(x_p(t)) \end{array} \right. \tag{4.1}$$

avec $x_p \in \mathbf{R}^{n_x}$ les états, $y_p \in \mathbf{R}^{n_y}$ les sorties. Les entrées sont maintenant subdivisées en deux catégories (on ne considérait que u dans le chapitre précédent) :

- $u \in \mathbf{R}^{n_u}$ les entrées commandées du système optimisées par le MPC
- $w_p \in \mathbf{R}^{n_w}$ les entrées non commandées du système (perturbations)

Les sorties y_p peuvent être des états x_p ou des variables algébriques z_p. z_p n'apparaît donc pas explicitement dans cette formulation mais peut s'exprimer comme une fonction de x.

Pour simplifier les notations, certaines variables seront parfois omises dans les fonctions si elles n'existent pas dans le modèle en question. Pour un système non perturbé par exemple, on pourra écrire $\dot{x}_p(t) = f_p(x_p(x_{p0}, u; t), u(t))$.

4.1.3.2 Modèle non linéaire du système

Le modèle le plus détaillé du système dont on dispose est le *modèle non linéaire*, modélisé à l'aide du langage Modelica. Celui-ci peut également être mis sous forme de modèle d'état.

$$\left\{ \begin{array}{rcl} \dot{x}(t) & = & f(x(x_0, u, w; t), u(t), w(t)) \\ y(t) & = & h(x(t)) \end{array} \right. \tag{NLc}$$

Avec la condition initiale $x(t_0) = x_0$.

Le modèle non linéaire est utilisé pour générer à chaque instant t_i certaines trajectoires appelées trajectoires nominales autour desquelles seront calculés des modèles linéarisés. Ces trajectoires sont calculées sous l'hypothèse que les entrées suivent leurs valeurs de référence et s'écrivent :

$$\mathtt{P}^i_{nom} = (x_{nom}, y_{nom}, u_r, w_r)$$

Ces trajectoires sont solutions de

$$\left\{ \begin{array}{rcl} x^i_{nom}(t_i) & = & x_{nom,i} \\ \dot{x}^i_{nom}(t) & = & f(x^i_{nom}(t), u_r(t), w_r(t)) \\ y^i_{nom}(t) & = & h(x^i_{nom}(t)) \end{array} \right. \tag{NLnom}$$

Les trajectoires nominales correspondent ainsi à la réponse en boucle ouverte du système non linéaire initialisé en un point $x_{nom,i}$ avec en entrée les trajectoires de référence données par l'étage d'optimisation supérieur.
Par ailleurs, pour simplifier la notation en temps discret, on donne une expression du modèle sous sa forme échantillonnée, avec un pas d'échantillonnage T_s :

$$\left\{ \begin{array}{rcl} x_{k+1} & = & f^{(d)}(x_k, u_k, w_k; T_s) \\ y_k & = & h(x_k) \end{array} \right. \tag{NLd}$$

La fonction $f^{(d)}$ renvoie l'état au prochain pas d'échantillonnage et correspond à l'intégration de (NLc) sur cette période. On utilisera la même notation pour le système réel (fonction $f_p^{(d)}$).

4.1.3.3 Modèle linéarisé tangent

On dérive du modèle (NLc) des *modèles linéarisés tangents* autour de la trajectoire nominale $\mathtt{P}_{nom}$. On présente ici la façon de les obtenir. L'opérateur δ est introduit pour définir l'écart d'une variable à sa valeur nominale (par exemple $\delta x(t) = x(t) - x_{nom}(t)$, $\delta y(t) = y(t) - y_{nom}(t)$, $\delta u(t) = u(t) - u_r(t)$,...). Dans un voisinage de la trajectoire nominale $\mathtt{P}_{nom}$, on peut écrire

$$\begin{aligned} \dot{\delta x} =& f(x_{nom} + \delta x, u_r + \delta u, w_r + \delta w) - f(x_{nom}, u_r, w_r(t)) \\ \dot{\delta x}(t) =& \left.\frac{\partial f}{\partial x}\right|_t \delta x(t) + \left.\frac{\partial f}{\partial u}\right|_t \delta u(t) + \left.\frac{\partial f}{\partial w}\right|_t \delta w(t) + g(\delta x(t), \delta u(t), \delta w(t)) \\ =& A(t)\delta x(t) + B_u(t)\delta u(t) + B_w(t)\delta w(t) + g(\delta x(t), \delta u(t), \delta w(t)) \end{aligned}$$

Ici, la fonction $g(\delta x, \delta u, \delta w)$ représente la partie non linéaire résiduelle du modèle. On définit alors le modèle linéarisé tangent en temps continu de la façon suivante

$$\left\{ \begin{array}{ll} \dot{\delta x}(t) & = A(t)\delta x(t) + B_u(t)\delta u(t) + B_w \delta w(t) \\ \delta y(t) & = c(t)\delta x(t) \end{array} \right. \tag{TanLc}$$

Par la suite, on utilisera la fonction δf pour abréger les notations :

$$\delta f(\delta x(t), \delta u(t), \delta w(t)) = A(t)\delta x(t) + B_u(t)\delta u(t) + B_w(t)\delta w(t)$$

Dans la première implémentation du MPC présentée dans cette partie, on suppose que la trajectoire de référence $\mathtt{P}_r = (x_r, y_r, u_r, w_r)$ est une trajectoire solution du modèle non linéaire (NLc) et peut donc être utilisée comme une trajectoire nominale. C'est l'hypothèse que nous ferons dans la partie 4.2 :

$$\mathtt{P}_{nom} = \mathtt{P}_r$$

La définition de la trajectoire nominale sera en revanche modifiée dans la partie 4.3.

4.1.3.4 modèle linéarisé tangent en temps discret

Ce dernier modèle aussi nommé modèle interne est celui utilisé par le MPC pour l'optimisation de trajectoires. Le MPC est appliqué dans un cadre échantillonné. On fait l'hypothèse que les entrées commandées mais également les perturbations peuvent être considérées constantes sur un pas d'échantillonnage (*zero-order hold*). Sous cette hypothèse, le modèle (NLc) peut être approché par le modèle linéaire variant dans le temps (LTV) discret suivant :

$$\begin{cases} \delta x_{k+1} &= A_k \delta x_k + B_{u,k} \delta u_k + B_{w,k} \delta w_k \\ \delta y_k &= c_k \delta x_k \end{cases} \tag{TanLd}$$

A_k, $B_{u,k}$, $B_{w,k}$ et c_k résultent de la discrétisation temporelle des matrices de linéarisation A, B_u, etc. autour du point $\mathtt{P}_{nom}(t_k)$. La discrétisation par schéma d'Euler avant avec l'hypothèse *zero order holder* est utilisée dans notre cas. On pose

$$F_k(\delta x_k, \delta u_k, \delta w_k) = A_k \delta x_k + B_{u,k} \delta u_k + B_{w,k} \delta w_k \tag{4.2}$$

Les trajectoires prédites par le MPC s'appuient sur l'hypothèse que les perturbations suivent leur valeur nominale : $\delta w_k = 0$. La notation $\bar{.}$ est introduite pour les fonctions et variables correspondant à ce cas. On définit ainsi :

$$\bar{F}_k(\delta x_k, \delta u_k) = A_k \delta x_k + B_{uk} \delta u_k$$

Les trajectoires non perturbées par rapport aux prévisions sont alors solution de

$$\bar{x}_{k+1} = x_{nom,k+1} + \bar{F}_k(\delta x_k, \delta u_k) \tag{4.3}$$

4.1.4 Commande prédictive TL-MPC

On introduit ici une nouvelle formulation de commande prédictive nommée TL-MPC (*Tangent Linear MPC*), qui utilise comme modèle interne des modèles linéarisés tangents du système à contrôler. On rappelle que l'on a pour l'instant défini la trajectoire nominale comme $\mathtt{P}_{nom} = \mathtt{P}_r$, et que par conséquent $x_r = x_{nom}$, $y_r = y_{nom}$. L'objectif de commande est de suivre au mieux les trajectoires calculées par le programme d'optimisation long terme, tout en respectant les contraintes de l'installation. Le suivi de trajectoire concerne ici les sorties comme cela est couramment le cas, mais également les commandes du système. En effet, la trajectoire des commandes peut avoir une influence sur le critère économique utilisé dans l'optimisation long terme, ou correspondre dans certains cas à des engagements d'achat.
Dans la procédure MPC, un problème de commande optimale est résolu à chaque instant t_i, afin de calculer le meilleur ajustement δu_i à la trajectoire de commande sur l'horizon de prédiction du MPC. La commande u réellement appliquée au système est ainsi déterminée par des variations autour de la trajectoire de référence u_r, c'est-à-dire $u = u_r + \delta u$. On définit alors les vecteurs suivants, calculés à l'instant t_i, qui entreront dans la définition du problème d'optimisation du MPC : les futurs ajustements de la commande sont δU^i et correspondent au vecteur de commande U^i. La trajectoire prédite pour les états est quant à elle X^i :

$$\delta U^i = \begin{pmatrix} \delta u_i \\ \vdots \\ \delta u_{i+P-1} \end{pmatrix}, \quad U^i = \begin{pmatrix} u_i \\ \vdots \\ u_{i+P-1} \end{pmatrix}, \quad X^i = \begin{pmatrix} x_i \\ \vdots \\ x_{i+P} \end{pmatrix}, \quad \delta X^i = \begin{pmatrix} \delta x_i \\ \vdots \\ \delta x_{i+P} \end{pmatrix}$$

On retient pour le problème MPC à l'instant d'échantillonnage t_i la fonction objectif quadratique suivante, composée de la somme de coûts d'étapes $l(\delta x, \delta u)$ et d'un coût terminal $\Phi(\delta x)$, avec

$$l(\delta x, \delta u) = \|\delta x\|_Q^2 + \|\delta u\|_R^2 \quad \text{et} \quad \Phi(\delta x) = \|\delta x\|_S^2$$

$$\begin{aligned} J_i &= \Phi(\delta x_{i+P}) + \sum_{k=i}^{i+P-1} l(\delta x_k, \delta u_k) \\ &= \|\delta x_{i+P}\|_S^2 + \sum_{k=i}^{i+P-1} \|\delta x_k\|_Q^2 + \|\delta u_k\|_R^2 \\ &= \|x_{i+P} - x_{r,i+P}\|_S^2 + \sum_{k=i}^{i+P-1} \|x_k - x_{rk}\|_Q^2 + \|u_k - u_{r,k}\|_R^2 \end{aligned} \tag{4.4}$$

Avec Q une matrice définie positive, R une matrice semi-définie positive et S une matrice définie positive symétrique (se référer à l'Annexe B).

Les deux premiers termes quantifient la qualité de suivi des trajectoires de référence, pour les commandes et les états respectivement. Les matrices de pondération R et Q permettent de définir le compromis entre ces deux objectifs. Le MPC doit également prendre en compte les contraintes sur l'état et les entrées. Soit $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ l'ensemble des états admissibles et $\mathcal{U} \subset \mathbb{R}^{n_u}$ l'ensemble des commandes admissibles.

Hypothèse H5. *Les ensembles $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ et $\mathcal{U} \subset \mathbb{R}^{n_u}$ sont convexes.*

Il est également possible de définir ces ensembles de façon locale autour des trajectoires de référence. Les ensembles selon cette définition deviennent variants dans le temps. Une contrainte terminale est également considérée : on requiert que l'état à la fin de l'horizon appartienne à un ensemble $\delta\Omega_{i+P}$ contenant $\delta x_{i+P} = 0$. L'ensemble $\delta\Omega_{i+P}$ est défini pour les variables en variations (δx) mais on pourra le définir de façon équivalente pour les variables du modèle original (x) : on le notera dans ce cas Ω_{i+P}. L'indice $i+P$ utilisé fait référence au fait que le l'ensemble terminal concerne δx à l'instant $i+P$ (mais il s'agit bien de l'ensemble terminal pour le problème MPC i). L'ensemble terminal est ainsi variant dans le temps. $\Omega_{i+P} = \mathbb{R}^{n_x}$ correspond au cas où l'état terminal est libre.

On ajoute l'hypothèse suivante sur l'ensemble terminal :

Hypothèse H6. *Ω_{i+P} est un ensemble compact contenant $x_{r,i+P}$ dont la frontière est définie comme une ligne de niveau du coût terminal :*

$$\Omega_{i+P} = \left\{\delta x_{i+P} | \quad \|\delta x_{i+P}\|_S^2 \leq r_{i+P}\right\}$$

Ω peut donc aussi être décrit par la trajectoire de r.

Le problème d'optimisation $(\text{MPC})_i$ à l'instant d'échantillonnage t_i est alors le suivant :

$$\begin{array}{ll} \min\limits_{\delta U^i} & J_i(\delta U^i, \delta X^i) \\ \text{s.c.} & (TanLd) \\ & \delta x_{i+P} \in \Omega_{i+P} \\ & \delta x_{i+k} \in \delta\mathcal{X}^{i+k}, \delta u_{i+k-1} \in \delta\mathcal{U}^{i+k-1} \quad \forall k \in \overline{1..P} \end{array} \tag{4.5}$$

4.2 Vers une preuve de stabilité de la commande TL-MPC

Dans cette partie, on commence par rappeler les définitions retenues pour la stabilité, puis les fonctions de Lyapunov qui sont un moyen de conclure sur la stabilité d'un système au sens défini précédemment. Dans notre cas, le système étudié est un système Linéaire Variant dans le Temps sous commande TL-MPC. Comme il n'existe pas de façon générique de créer une fonction de Lyapunov pour un système, on présente certains résultats de la littérature pour les systèmes sous commande MPC d'une part, pour les systèmes linéaires variant dans le temps d'autre part. La combinaison de ces deux aspects ajoute des difficultés théoriques aux preuves de stabilité : en particulier les ensembles admissibles pour les contraintes deviennent variant dans le temps, et la stabilité doit être prouvée non plus pour un modèle (de la forme $[A, B]$) mais un ensemble de modèles ($[A_k, B_k]$) qu'il faudra tout d'abord réussir à caractériser. Au vu de ces difficultés, des hypothèses supplémentaires sur le système ont été formulées. On donne alors dans cette partie une preuve de stabilité pour le système sous commande TL-MPC dans un cadre simplifié.

4.2.1 Définitions de la stabilité

On rappelle ici quelques définitions classiques de la stabilité dans le cadre échantillonné, pour le cas général des systèmes variant dans le temps. La stabilité d'un point d'équilibre x_e traduit l'aptitude du système à se maintenir dans un certain voisinage de x_e, tant que la trajectoire débute suffisamment près de ce point. Une notion plus forte est celle de convergence, qui assure que toutes les trajectoires du système commençant dans une région d'attraction autour de x_e tendent asymptotiquement vers ce point. Par ailleurs, lorsque les équations du système étudié varient dans le temps, la dynamique n'est plus semblable en tout instant, et il faut alors préciser l'instant initial t_0 dans les définitions de la stabilité. On dira que le point x_e est stable à l'instant t_0. La stabilité uniforme, qui traduit la stabilité du système indépendamment de l'instant initial peut alors être une propriété souhaitable. De façon plus formelle, en notant $\mathcal{B}_d$ la boule définie par $\|x\| < d$ et en introduisant les constantes positives r (borne sur l'état initial à t_0) et R (borne sur l'état à tout instant t_i), on établit les définitions suivantes :

Définition (Stabilité) Le point d'équilibre $x_e = 0$ est stable au sens de Lyapunov au temps t_0 si pour $R > 0$, on peut trouver $r(R, t_0) > 0$ tel que si $x(t_0) \in \mathcal{B}_r$ alors $x(t_i) \in \mathcal{B}_R, \quad \forall t_i > t_0$.

Remarque En temps discret, cette définition reste valable avec $t_i = i * T_s, i \in \mathbb{N}$.

On parlera de stabilité uniforme si r peut être choisi indépendamment de t_0. La notion de convergence vers un point x_e (ou convergence du point x_e par abus de langage) est liée quant à elle à l'existence de domaines d'attraction : toute trajectoire débutant dans un domaine d'attraction autour de x_e tend asymptotiquement vers ce point. On parlera de stabilité asymptotique si le point est à la fois stable et convergent.

Définition (Convergence) Le point d'équilibre $x = 0$ est convergent à t_0 si $\forall R > 0$ on peut trouver $r(t_0) > 0$ et $T(R, t_0) \geq 0$ tel que si $x(t_0) \in \mathcal{B}_r$, $x(t_i) \in \mathcal{B}_R, \quad \forall t_i \geq t_0 + T$

La convergence est uniforme lorsque r et T peuvent être choisis indépendamment de t_0.
Ce qui diffère dans notre cas d'étude est que l'on ne recherche pas la stabilité autour d'un point

fixe, mais d'une trajectoire, en l'occurrence $\mathtt{P}_r = (x_r(t), u_r(t))$, la trajectoire de référence. Il est cependant possible de se ramener à un problème de stabilisation si l'on considère le système linéarisé $(\delta x(t), \delta u(t))$. La stabilité de $x(t)$ en $x_r(t)$ est alors équivalente à la stabilité du point $(\delta x, \delta u) = (0, 0)$, mais pour un système variant dans le temps (le système linéarisé tangent).

4.2.2 Fonctions de Lyapunov

L'analyse de stabilité peut recourir à l'utilisation de fonctions de Lyapunov, que l'on évalue le long de la trajectoire du système. Montrer que le système admet une fonction de Lyapunov pour le point $x_e = 0$ (en utilisant au besoin des changements de variables) équivaut à prouver la stabilité de ce point.

Les fonctions de Lyapunov font intervenir certaines classes de fonctions que l'on introduit ici :

Définition (Fonction $V(x)$ définie positive) : Une fonction $V : \mathbb{R}^n \to \mathbb{R}^+$ est définie positive si :
- $V(0) = 0$
- $V(x) > 0, \quad \forall x \neq 0$

Définition (Fonction $W(x, k)$ définie positive) : Une fonction $W : D \times \mathbb{N}^+ \to \mathbb{R}^+$ est définie positive sur D si :
- $W(0, k) = 0, \quad \forall k \geq 0$
- Il existe une fonction définie positive $V_1(x)$ telle que $V_1(x) \leq W(x, k), \quad \forall x \in D, \forall k$

Il existe deux définitions pour les fonctions de Lyapunov : la première concerne les systèmes invariant dans le temps (les équations sont les mêmes en chaque instant), la seconde concerne les systèmes variant dans le temps, ce qui est le cas des modèles linéarisés tangents étudiés ici.
On donne maintenant les deux théorèmes suivants qui relient la stabilité d'un point d'équilibre à l'existence d'une fonction de Lyapunov pour le système. Le Théorème 4.1 concerne le cas invariant :

Théorème 4.1. *(Stabilité pour les systèmes invariant à temps discret) Le point d'équilibre $x_e = 0$ est stable dans un voisinage $\mathcal{B}_D$ s'il existe une fonction définie positive $V(x) : \mathbb{R} \to \mathbb{R}^+$ appelée fonction de Lyapunov et une fonction V_3 définie positive telles que le taux de variation de V le long d'une trajectoire de x vérifie*

$$\Delta V(x_k) = V(x_{k+1}) - V(x_k) \leq -V_3(x_k) \quad \forall x_k \in \mathcal{B}_D \tag{4.6}$$

Le Théorème 4.2 est une généralisation au cas variant dans le temps :

Théorème 4.2. *(Stabilité uniforme pour les systèmes à temps discret variants dans le temps) Le point d'équilibre $x_e = 0$ est uniformément stable dans un voisinage $\mathcal{B}_D$ si il existe une fonction $V(x, k) : \mathbb{R} \times \mathbb{N} \to \mathbb{R}$ appelée fonction de Lyapunov et trois fonctions V_1, V_2, V_3 définies positives telles que*
-

$$V_1(x_k) \leq V(x_k, k) \leq V_2(x_k) \quad \forall x_k \in \mathcal{B}_D \tag{4.7}$$

– *Le taux de variation de V le long d'une trajectoire de x est semi-défini négatif*

$$\Delta V(x_k, k) = V(x_{k+1}, k+1) - V(x_k, k) \leq -V_3(x_k) \quad \forall x_k \in \mathcal{B}_D \tag{4.8}$$

Si la deuxième condition est renforcée par $\Delta V(x_k, k) < -V_3(x_k) \quad \forall x_k \in \mathcal{B}_D$ le point d'équilibre 0 est asymptotiquement (uniformément) stable.

Les preuves de stabilité nécessitent donc la recherche de fonctions V, V_1, V_2, V_3 (V et V_3 dans le cas invariant) et d'un domaine $\mathcal{B}_D$ vérifiant ces conditions.

4.2.3 Etat de l'art : stabilité pour le MPC et les systèmes LTV

On présente ici certains résultats de la littérature permettant de conclure à la stabilité :
- De la commande MPC dans le cas de la stabilisation des systèmes invariant dans le temps
- Des systèmes linéaires variant dans le temps
- De la commande MPC pour les systèmes linéaires variant dans le temps

La preuve de stabilité de la commande TL-MPC proposée ensuite combine certains éléments de ces approches.

4.2.3.1 Preuves de stabilité MPC pour les systèmes invariants

Dans le très cité article [68], une synthèse est faite sur un ensemble de briques élémentaires garantissant la stabilité de la commande MPC pour les systèmes invariants dans le temps, linéaires ou non linéaires. Le problème étudié est celui de la stabilisation autour du point d'équilibre $x_e = 0$ en présence de contraintes sur l'état et la commande. On considère un critère de la forme

$$J_i = \Phi(x_{i+P}) + \sum_{k=i}^{i+P-1} l(x_k, u_k)$$

Il y est noté l'importance des ingrédients suivants :
- Un ensemble terminal $\Omega \subset \mathcal{X}$ contenant 0. La contrainte $x_{i+P} \in \Omega$ est ainsi ajoutée au problème MPC_i.
- Un horizon de prédiction P suffisamment long pour garantir la faisabilité initiale : à l'instant initial, l'ensemble terminal Ω est atteignable en P pas de temps en respectant les contraintes sur x et u.
- La connaissance d'un contrôleur terminal $u = \mathcal{K}_f(x, k)$ capable de prendre le relais du MPC sur Ω au delà de l'horizon de prédiction fini. Le but de ce contrôleur est de prouver qu'il existe au moins une stratégie admissible permettant de compléter la trajectoire pour le prochain pas : celle consistant à appliquer la commande optimale calculée par le MPC jusqu'à l'instant $i+P$, puis $\mathcal{K}_f(x_{i+P+k})$. L'optimisation effectuée par le MPC ne pourra qu'améliorer cette solution existante. Le contrôleur terminal $\mathcal{K}_f$ est lié à l'ensemble terminal Ω. Il doit répondre à deux exigences : fournir un retour d'état admissible pour tous les points de l'ensemble terminal : $\mathcal{K}_f(x) \in \mathcal{U}, \forall x \in \Omega$, et rendre l'ensemble terminal positivement invariant :

$$x_{k+1} = f(x_k, \mathcal{K}_f(x_k)) \in \Omega, \forall x_k \in \Omega$$

En d'autre termes, une fois que l'ensemble terminal est atteint, l'état x y est maintenu pour tous les pas suivants, assurant le respect des contraintes en x et u au delà de l'horizon du MPC. Il faut cependant noter qu'en raison de l'horizon glissant du MPC, ce contrôleur n'est en réalité jamais appliqué au système réel.

– Un coût terminal $\Phi(x)$ dans la fonction objectif permettant de pénaliser l'état final x_{i+P}. Ce coût terminal interviendra dans les preuves de stabilité qui vont suivre en utilisant l'hypothèse suivante :

Hypothèse H7. *Le coût terminal $\Phi(x)$ et le contrôleur terminal $\mathcal{K}_f(x)$ vérifient*

$$\Phi\left(f_k(x_k, \mathcal{K}_f(x_k))\right) - \Phi(x_k) \leq -l(x_k, \mathcal{K}_f(x_k)), \quad \forall x_k \in \Omega, k \in \mathbb{N} \tag{4.9}$$

Cette dernière condition permet d'utiliser la valeur de la fonction objectif J_i^* comme fonction de Lyapunov. La preuve de stabilité du système en boucle fermée consiste à trouver des fonctions définies positives V, V_3 et un domaine $\mathcal{B}_D$ respectant les conditions du théorème 4.2. Il faut par ailleurs veiller au respect des contraintes indépendamment de la stabilité. Les grandes lignes de la démonstration sont les suivantes :

1. La fonction $l(x,u)$ permet sous certaines hypothèses de garantir l'existence d'une fonction $V_1(x) \leq l(x,u) \leq J_i^*$ telle que J_i^* soit définie positive. Dans le cadre des systèmes invariant dans le temps, il n'est pas nécessaire de prouver que J_i^* est décroissante.
2. Pour le problème (MPC$_{i+1}$), une solution est la commande consistant à appliquer U_i^* calculée au pas i de $i+1$ à $i+P-1$, puis le contrôleur terminal $\mathcal{K}_f$. Sous l'hypothèse d'un modèle parfait, cette trajectoires donnera les mêmes états X_i^* de $i+1$ à $i+P$, puis $x_{i+P+1} = \mathcal{K}_f(x_{i+P})$. Cette solution est associée à un coût sous optimal $J_{i+1} \geq J_{i+1}^*$ qui peut s'écrire sous la forme

$$J_{i+1} = J_i^* + \Phi\left(f_k\left(x_{i+P}, \mathcal{K}_f(x_{i+P})\right)\right) + l\left(f_k(x_{i+P}, \mathcal{K}_f(x_{i+P})\right) - \Phi(x_{i+P}) - l(x_i^*, u_i^*))$$

3. L'hypothèse H7 permet alors d'écrire $J_{i+1} - J_i \leq J_{i+1}^* - J_i^* \leq -l(x_i^*, u_i^*)$. Moyennant certaines hypothèses sur $l(x,u)$, cette fonction joue le rôle de V_3 dans le théorème 4.2. Les conditions sont alors réunies pour utiliser J_i^* comme une fonction de Lyapunov et conclure à la stabilité du système sous commande MPC dans le cas nominal.

Remarque En présence de perturbations, on ne peut garantir que l'état converge vers un point, mais seulement vers un ensemble borné. On utilisera alors d'autres notions relaxées de la stabilité comme la stabilité entrée-état (ISS), faisant intervenir un autre type de fonctions de Lyapunov.

4.2.3.2 Preuves de stabilité pour les systèmes LTV

Les preuves de stabilité pour les systèmes linéaires variant dans le temps sont quelque peu différentes. Une approche est de les reformuler comme des systèmes Linéaires à Paramètres Variants (LPV). Les LPV sont une façon de formuler les modèles linéaires variant dans le temps en fonction de certains paramètres (ξ) (donnant $A(\xi(t))$, $B(\xi(t))$,etc.) et non en fonction du temps comme nous l'avons fait jusqu'à présent (donnant $A(t_k)$, $B(t_k)$,etc.), même si techniquement on pourrait considérer l'instant $k \in i, .., i+P$ comme le paramètre en question. Dans [69], une preuve de stabilité est donnée pour les systèmes Linéaires à Paramètres

Variants (LPV) pour lesquels la matrice B est constante :

$$x_{k+1} = A(\xi(t_k))x_k + Bu_k$$

On suppose par ailleurs qu'il existe un ensemble de matrices sommets $\mathbf{A}^i$ définissant l'ensemble convexe des matrices $A(\xi)$. La définition de l'enveloppe convexe de matrices est donnée par la définition B.2 de l'Annexe B. De plus B étant constant, on a ici

$$[A(\xi(t_k)), B] = \sum_{i=1}^{n_A} \xi_i(t_k)[\mathbf{A}^i, B] \tag{4.10}$$

$$\sum_{i=1}^{n_A} \xi_i(t_k) = 1, \quad \xi_i(t_k) \geq 0 \tag{4.11}$$

Avec $[A, B]$ la représentation d'état du système LTV. Chacun des systèmes sommets $[\mathbf{A}^i, B], i \in \overline{1..n_A}$ est associé à une fonction de Lyapunov quadratique $V_i(x(t_k)) = x(t_k)^T P_i x(t_k)$, mais cela n'est pas suffisant pour garantir la stabilité du système pour toute valeur du paramètre ξ. Une condition nécessaire et suffisante pour que

$$V(x(t_k), \xi(t_k)) = \sum_{m=1}^{n_A} \xi_m(t_k) P_m$$

soit une fonction de Lyapunov globale valable pour toute évolution de $\xi(k)$ est donnée par le théorème suivant :

Théorème 4.3. *[69] Le système (4.10) est stabilisable par un retour d'état linéaire de la forme :*

$$u_k = \sum_{i=1}^{n_A} \xi_i(t_k) K_i x_k$$

(gain scheduled state feedback) si et seulement si il existe des matrices S^i, S^j définies positives et des matrices G^i et R^i solution de

$$\begin{bmatrix} G^i + {G^i}^T - S^i & \star \\ A^i G^i + B R^i & S^j \end{bmatrix} \geq 0, \quad \forall i \in \overline{1..n_A}, \quad \forall j \in \overline{1..n_A} \tag{4.12}$$

Où $\star$ est une notation utilisée pour remplacer les termes transposés dans les matrices symétriques. Les contrôleurs K^i sont alors donnés par

$$K^i = R^i {G^i}^{-1}$$

Et la fonction suivante est une fonction de Lyapunov du système :

$$P(\xi(t_k)) = \sum_{i=1}^{n_A} \xi_i(t_k) {S^i}^{-1}$$

On parle alors de stabilité poly-quadratique. [70] propose une version moins conservative de ce théorème dans le cas où la variation temporelle de $\xi(k)$ est bornée, ce qui est souvent le cas en pratique. Le résultat de stabilité est étendu au cas où la matrice B dépend de ξ en considérant une forme plus générale pour la fonction de Lyapunov candidate Ces résultats diffèrent de notre cas d'étude, puisque la commande recherchée est une commande MPC et non un retour d'état linéaire. Cependant, ils nous seront utiles pour la synthèse du contrôleur final $\mathcal{K}_f$ évoqué précédemment.

4.2.3.3 Preuves de stabilité MPC pour les sytèmes LTV

Dans le cadre des systèmes variant dans le temps, le problème de suivi nominal de la section 4.2.4.2 est traité par [71] dans le cas de systèmes linéarisés tangents à temps continu autour d'une trajectoire de référence $x_r(t)$. En supposant que cette trajectoire de référence est une trajectoire solution du système non linéaire, on cherche ici à stabiliser l'erreur $e(t)$, dont la dynamique est donnée par :

$$\dot{e}(t) = \dot{x}(t) - \dot{x}_r(t) = f(x(t), u(t)) - \dot{x}_r(t)$$

C'est à notre connaissance le cadre d'étude le plus proche du nôtre. La stabilité de la commande MPC est prouvée dans le cas où les trajectoires de références atteignent un état stationnaire à partir d'un instant T. Le concept introduit dans [71] est celui d'ensembles $\Omega(t)$ positivement invariants sous un contrôleur linéaire $\delta u(t) = K(t)\delta x(t)$, c'est-à-dire que pour tout point $x(t)$ dans $\Omega(t)$ à l'instant t génère une trajectoire qui restera dans Ω dans le futur si l'on applique le contrôleur $K(t)$:

$$\forall \tau, \quad x(t) \in \Omega(t) \Rightarrow x(t+\tau) \in \Omega(t+\tau)$$

Comme on peut le constater, le contrôleur K ainsi que l'ensemble Ω évoluent au cours du temps. L'invariance positive permet de garantir l'existence d'un contrôle admissible (satisfaction récursive des contraintes) sur tout l'horizon, donnant par là même une borne supérieure pour la fonction coût. Le contrôleur K est ici calculé comme solution d'une équation de Ricatti variant dans le temps, l'hypothèse sur la stationnarité de la référence à $t = T$ ($\dot{x}_r(T) = 0$) nous donnant la condition limite sur la matrice de covariance : $P(T) = 0$.
Le cas où la référence est donnée pour les sorties y_r en supposant celles-ci de classe $\mathcal{C}_1$ est aussi abordé, mais les conditions de convergence vers la référence sont données sous des hypothèses plus fortes : on suppose notamment qu'il existe une trajectoire d'entrée telle que $y_r(t) = h(x(t))$ (*output trackability*), et l'on impose la contrainte terminale $y_r(t_i + T) = h(x_i(t+T))$ pour chaque problème MPC_i.

Par ailleurs les ensembles $\Omega(t)$ sont choisis comme des ellipsoïdes de rayon $\Pi(t)$. L'auteur donne alors une façon approchée de calculer ces ensembles $\Omega(t)$ afin qu'ils soient les plus grands possibles. Par la suite, l'erreur de modélisation est prise en compte en considérant une marge de sécurité sur ces ensembles, c'est-à-dire en définissant un ensemble $\Omega^r \subset \Omega$ de telle sorte que Ω^r reste invariant pour toutes les réalisations possibles de l'incertitude. Ω^r est ici construit comme une contraction homothétique de Ω (soit pour le rayon $\Pi^r(t) < \Pi(t)$).
Cette approche, où l'on fixe la forme de l'ensemble terminal mais on en modifie la taille par homothétie, se retrouve chez [72][73][74] (ensembles polytopiques). Dans notre cas, on retiendra une approche quelque peu similaire. Cependant, la condition de stationnarité des trajectoires de référence n'est en général pas vérifiée dans notre cas. Plutôt que de garantir la stabilité sur un horizon infini, on se restreindra alors à une condition d'invariance à M pas au delà de l'horizon de prédiction P, nous garantissant que le problème restera réalisable pour les M problèmes MPC suivants. L'horizon $P + M$ peut être choisi suffisamment long pour qu'une procédure de sécurité puisse être déclenchée si une infaisabilité est détectée. La démonstration qui suit considère $M = 1$.

4.2.4 Analyse de stabilité de la commande prédictive TL-MPC pour le modèle idéal non perturbé

Dans cette partie, on propose une analyse de stabilité pour une version simplifiée de la commande TL-MPC. On commence par décrire le système étudié et les limites du cadre d'étude de la stabilité dans ce que l'on nomme le cas nominal. La démonstration fait intervenir certaines hypothèses sur l'ensemble terminal, le coût terminal et sur l'existence d'un contrôleur terminal rendant l'ensemble terminal invariant. On montre alors comment il est possible de construire ces différents éléments afin que les hypothèses formulées soient vérifiées.

4.2.4.1 Évolution du système en boucle fermée

La commande u_i appliquée au système à chaque instant d'échantillonnage t_i résulte de l'optimisation des trajectoires du modèle (TanLd) qui minimisent le critère J_i. Cette commande dépend ainsi implicitement de l'état courant mais aussi de l'instant d'optimisation puisque les modèles linéarisés tangents F_k qui interviennent dans J_i varient dans le temps. La commande calculée par le MPC peut alors être exprimée comme un retour d'état implicite variant dans le temps noté κ :

$$u_k^* = \kappa(x_k, k) \tag{4.13}$$

L'évolution du système en boucle fermée est alors donnée par

$$x_{p,k+1} = f_p^{(d)}(x_{p,k}, \kappa(x_k, k), w_{pk}) \tag{4.14}$$

On fait maintenant l'hypothèse que l'incertitude liée aux perturbations et à l'erreur du modèle interne peut être représentée comme une perturbation additive d_k. La relation entre les trajectoires prédites par le MPC (4.3) et l'évolution réelle de la centrale peut alors être définie comme :

$$x_{k+1} = \bar{x}_{k+1} + d_k = x_{r,k+1} + A_k \delta x_k + B_{uk} \delta u_k + d_k \tag{4.15}$$

4.2.4.2 Stabilité dans le cas nominal

On propose ici une analyse de stabilité du système sous commande TL-MPC(4.14), dans le cas nominal suivant :

Hypothèse H8. *Le cas nominal correspondant à $d_k = 0$ est lié aux hypothèses suivantes :*
- *L'optimisation est faite sous l'hypothèse que les perturbations w suivent leurs trajectoires de référence w_r. Les perturbations sont donc nulles pour le modèle interne en variations $\delta w = 0$*
- *Le modèle de prédiction (TanLd) est supposé parfait. Il prédit exactement l'évolution du système lorsque $\delta w = 0$. Ainsi, la trajectoire de référence $\mathtt{P}_r = (x_r, u_r, w_r)$ est une trajectoire solution de (TanLd).*

On énonce ci-dessous les propriétés souhaitées pour le système en boucle fermée dans le cas nominal :

Problème (Problème de suivi nominal) Étant donnée une trajectoire de référence en l'état x_r solution de (4.1), le MPC doit satisfaire :

– *Convergence* L'état de la centrale converge vers la référence :

$$\lim_{k\to\infty} \|x_{p,k} - x_{r,k}\| = 0$$

– *Respect des contraintes* Les contraintes sur les entrées et les états sont respectées à tout instant :

$$\forall t \geq 0, x_p(t) \in \mathcal{X}, u(t) \in \mathcal{U}$$

Dans le MPC, la commande appliquée au système est le résultat d'un problème d'optimisation, et n'est donnée que de façon implicite. L'étude de la stabilité pour le système sous commande MPC est donc par nature plus complexe que celle de la stabilité d'un système en boucle fermée pour une loi de commande connue. On cherche donc à vérifier si appliquer le premier terme d'une commande optimale recalculée à chaque pas de temps sur un nouvel horizon permet de répondre au problème de suivi nominal.

Le problème MPC_i résolu à chaque pas de temps est celui décrit en (4.5), ayant pour fonction objectif

$$J_i = \|\delta x_{i+P}\|_S^2 + \sum_{k=i}^{i+P-1} \|\delta x_k\|_Q^2 + \|\delta u_k\|_R^2 \tag{4.16}$$

On cherche à prouver la stabilité du système autour de la trajectoire de référence (x_r, u_r), ce qui, en faisant le changement de variables en variables locales $(\delta x, \delta u)$ revient à prouver la stabilité du modèle en variation (TanLd) autour de $(\delta x, \delta u) = (0, 0)$. L'étude du système (TanLd) permet de nous rapprocher du problème de stabilisation autour de l'origine introduit dans le paragraphe précédent. Cependant, il existe une différence majeure avec ce qui a été présenté jusqu'alors : le modèle considéré est maintenant variant dans le temps, et il en va de même pour le domaine admissible des contraintes lorsqu'on les exprime non plus en fonction de (x, u) mais en fonction de $(\delta x, \delta u)$. Pour cette raison, les éléments introduits précédemment doivent être adaptés.
En particulier, on introduit un contrôleur terminal $\mathcal{K}_f$ et un ensemble terminal $\delta\Omega$ pour le modèle en variations, qui à la différence de ce qui était suggéré précédemment, est maintenant autorisé à varier dans le temps.

Hypothèse H9. *Le contrôleur terminal est un contrôleur linéaire variant dans le temps*

$$\mathcal{K}_f(\delta x, k) = K_k \delta x \tag{4.17}$$

Il vérifie

$$u_{r,k} + \mathcal{K}_f(\delta x, k) \in \mathcal{U}, \quad \forall \delta x_k \in \delta\Omega_k \tag{4.18}$$

L'ensemble terminal de δx_{i+P} pour le problème MPC_i est choisi comme un ensemble compact variant dans le temps noté $\delta\Omega_{i+P}$. Pour les variables (x, u) du modèle original, cet ensemble correspond donc également à un ensemble Ω_{i+P} compact contenant $x_{r,i+P}$. Sa construction sera détaillée par la suite.

Hypothèse H10. *(Invariance positive de l'ensemble terminal sous $\mathcal{K}_f$) L'ensemble terminal et le contrôleur $\mathcal{K}_f$ satisfont les propriétés suivantes :*

– *Invariance positive :*

$$(A_{k+P} + B_{k+P}K_{k+P})\delta x_{k+P} \in \delta\Omega_{k+P+1}, \quad \forall \delta x_{k+P} \in \delta\Omega_{k+P}, \quad \forall k$$

– *Respect des contraintes :*

$$\delta x_{k+P} \in \delta\mathcal{X}_{k+P}, \quad K_{k+P}\delta x_{k+P} \in \delta\mathcal{U}_{k+P}, \quad \forall k$$

Ces deux dernières hypothèses garantissent qu'une fois que l'état a atteint l'ensemble terminal, le contrôleur terminal permet de l'y maintenir tout en garantissant le respect des contraintes. L'ensemble terminal est dit invariant positivement sous le contrôleur $\mathcal{K}_f$, un terme qui peut ici porter à confusion car Ω_k varie dans le temps. Une méthode pour vérifier ces deux dernières hypothèses sera donnée par la suite dans la partie 4.2.4.4.

On introduit maintenant une hypothèse similaire à l'hypothèse H7. Comme on le montrera par la suite, cette hypothèse va permettre d'utiliser J_i^*, la valeur de la fonction objectif du MPC comme une fonction de Lyapunov V du système. Dans notre cas, l'hypothèse H7 s'exprime de la façon suivante :

Hypothèse H11. *Le contrôleur $\mathcal{K}_f(\delta x, k) = K_k\delta x$ et les matrices S, Q et R vérifient :*

$$\|F_k(\delta x_k, K_k\delta x_k)\|_S^2 - \|\delta x_k\|_S^2 \leq -(\|\delta x_k\|_Q^2 + \|K_k\delta x_k\|_R^2), \quad \forall \delta x_k \in \delta\mathcal{X}_k, k \in \mathbb{N} \tag{4.19}$$

On commence par introduire un résultat intermédiaire utile dans la preuve de stabilité qui donne une borne supérieure pour J_i^*.

Lemme 4.4. *Si $\|\delta x_i\|_S^2 \leq r_i$ (où r_i définit la frontière de l'ensemble terminal), et l'hypothèse H11 est vérifiée, alors $J_i^* \leq \|\delta x_i\|_S^2$*

Démonstration. La borne est obtenue en comparant la trajectoire de commande optimale calculée par le MPC sur l'horizon $[t_i, t_{i+P}]$ avec la commande (sous-optimale) calculée par le contrôleur final $\mathcal{K}_f(x, k)$. Par l'hypothèse H10, l'application du contrôleur $\mathcal{K}_f(x, k)$ fournit une trajectoire admissible pour tout $t > t_i$. On note $J_i^{\mathcal{K}_f}$ le coût associé à cette commande. D'après l'hypothèse H11, on peut écrire :

$$\|F_i(\delta x_i, K_i\delta x_i)\|_S^2 - \|\delta x_i\|_S^2 + (\|\delta x_i\|_Q^2 + \|K_i\delta x_i\|_R^2) \leq 0$$

Si le contrôleur terminal $\mathcal{K}_f$ est appliqué sur l'horizon $[t_i, t_{i+P}]$, alors la trajectoire réelle du système vérifiera $\delta x_{i+k+1} = F_{i+k}(\delta x_{i+k}, K_k\delta x_{i+k})$. On peut alors écrire la même égalité pour tous les $k \in \overline{0..P-1}$:

$$\begin{aligned} &\|F_{i+k}(\delta x_{i+k}, K_{i+k}\delta x_{i+k}\|_S^2 - \|\delta x_{i+k}\|_S^2 - \left(\|\delta x_{i+k}\|_Q^2 + \|K_{i+k}\delta x_{i+k}\|_R^2\right) \leq 0 \\ \Leftrightarrow\ &\|\delta x_{i+k+1}\|_S^2 - \|\delta x_{i+k}\|_S^2 - \left(\|\delta x_{i+k}\|_Q^2 + \|K_{i+k}(\delta x_{i+k})\|_R^2\right) \leq 0 \end{aligned}$$

La somme de ces P inégalités donne :

$$\|\delta x_{i+P}\|_S^2 - \|\delta x_i\|_S^2 + \sum_{k=0}^{P-1}(\|\delta x_{i+k}\|_Q^2 + \|K_k\delta x_{i+k}\|_R^2) \leq 0 \quad \Leftrightarrow \quad J_i^{\mathcal{K}_f} \leq \|\delta x_i\|_S^2 \tag{4.20}$$

Par optimalité, on conclut que

$$J_i^* \leq J_i^{\mathcal{K}_f} \leq r_i, \quad \forall \delta x_i \in \Omega_i$$

∎

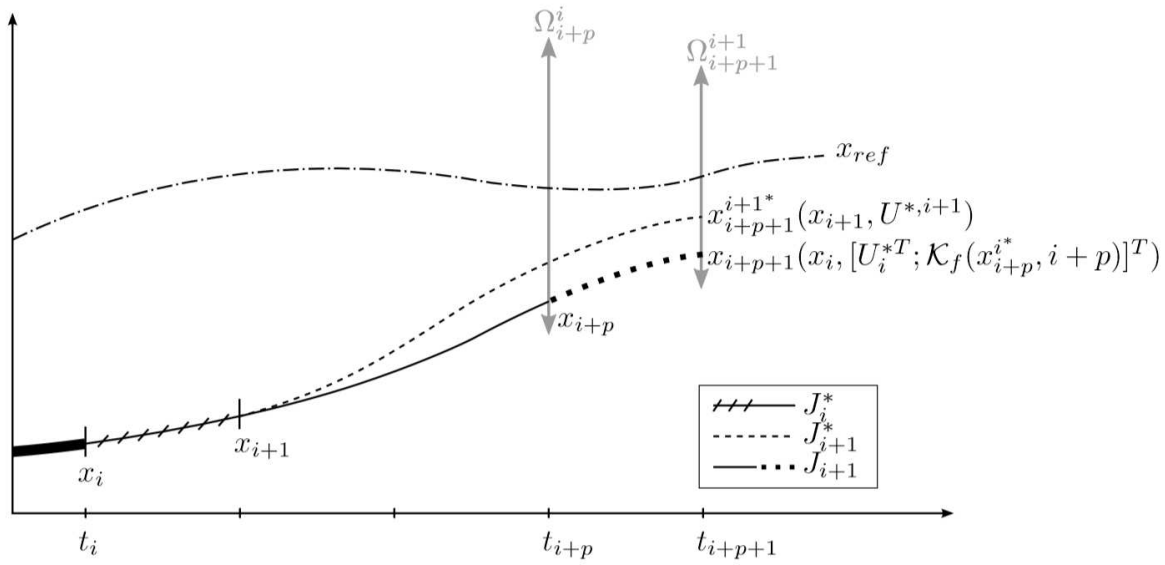


FIGURE 4.1 – Trajectoires optimales et sous-optimales utilisées dans la preuve de stabilité

Le résultat suivant démontre l'existence d'une fonction de Lyapunov pour le système commandé par TL-MPC dans le cas nominal, et par conséquent sa stabilité. La preuve fait intervenir certaines trajectoires montrées dans la Figure 4.1 que l'on détaille ici. La première trajectoire est définie de la façon suivante : à l'instant courant t_i, l'état est en x_i et une trajectoire optimale est calculée, montrée en trait plein. L'application de son premier terme u_i^* amène le système en x_{i+1} à l'instant t_{i+1} (courbe hachurée). En raison de l'hypothèse de modèle parfait et de l'absence de perturbations, $x_{i+1} = x^*_{i+1}$ et de même pour les pas suivants si l'on applique la commande optimale $U^{i,*}$ (courbe en trait plein). A la fin de l'horizon du problème MP^i, l'état est en x^*_{i+P} et on applique alors le contrôleur terminal $\mathcal{K}_f(\delta x^*_{i+P}, i+P)$ pour compléter la trajectoire jusqu'en t_{i+P+1} (courbe en pointillés). Cette trajectoire est donc optimale sur $[t_i, t_{i+P}]$ mais pas sur $[t_{i+1}, t_{i+P+1}]$ La seconde trajectoire (courbe en tirets) correspond quant à elle à la commande optimale du problème MPC_{i+1} calculée en t_{i+1} et partant de l'état x_{i+1}. Cette trajectoire est donc optimale sur $[t_{i+1}, t_{i+P+1}]$

Théorème 4.5. *Sous les hypothèses H8 et H11, le système (TanLd) est stabilisé en* $(\delta x, \delta u) = (0,0)$ *pour tout* $\delta x_0 = \delta x(t_0)$ *tel que* $\|\delta x_0\|^2_S \leq r_0$.

Démonstration. 1. Les matrices S, Q et R sont définies positives. Par conséquent,

$$J_i^* = \|\delta x^*_{i+P}\|^2_S + \sum_{k=i}^{i+P-1} \|\delta x^*_k\|^2_Q + \|\delta u^*_k\|^2_R \geq \|\delta x^*_i\|^2_Q$$

Soit $V_1(\delta x) = \|\delta x\|^2_Q$. V_1 est définie positive car Q est une matrice définie positive, d'où $J_i^* \geq V_1(\delta x)$. Par conséquent, J_i^* est une fonction définie positive.

2. On montre ensuite que le taux de variation de J_i^* est semi-défini négatif. Pour cela, on commence par construire une séquence de commande admissible (mais pas forcément optimale) pour le problème (MPC_{i+1}) et l'on étudie la variation du coût dans ce scénario.

J_i^* correspond à la commande optimale du problème (MPC$_i$) :

$$U^{i,*} = \begin{pmatrix} \delta u_i^* \\ \vdots \\ \delta u_{i+P-1}^* \end{pmatrix}$$

A l'instant t_i, le MPC applique δu_i^* au système. Comme on fait l'hypothèse que le modèle est parfait, le problème suivant (MPC$_{i+1}$) a pour condition initiale la valeur prédite $\delta x_{i+1}^* = A_i \delta x_i^* + B_{ui} \delta u_i^*$, et il en est de même pour les termes δx_{i+k}^* suivants si l'on applique le reste de la séquence $U^{i,*}$. $U^{i,*}$ est donc une commande admissible pour le problème (MPC$_{i+1}$) de t_{i+1} à t_{i+P-1}. Pour le dernier pas $i+P+1$ qui n'était pas dans l'horizon du problème précédent, on peut choisir d'appliquer le contrôleur final. On note $\bar{U}^{i+1}$ cette stratégie de commande sous-optimale pour le problème (MPC$_{i+1}$) :

$$\bar{U}^{i+1} = \begin{pmatrix} \delta u_{i+1}^* \\ \vdots \\ \delta u_{i+P-1}^* \\ K_{i+P} \delta x_{i+P} \end{pmatrix}$$

On calcule maintenant la variation de J pour cette stratégie. En raison de l'hypothèse modèle parfait, les termes de $i+1$ à $i+P$ se simplifient.

$$\begin{aligned}
\Delta J_{i+1} =& J_{i+1} - J_i^* \\
=& \|\delta x_{i+P+1}^K\|_S^2 + \sum_{k=i+1}^{i+P} \|\delta x_k^*\|_Q^2 + \|\delta u_k^*\|_R^2 \\
& - \left(\|\delta x_{i+P}\|_S^2 + \sum_{k=i}^{i+P-1} \|\delta x_k^*\|_Q^2 + \|\delta u_k^*\|_R^2 \right) \\
=& \|\delta x_{i+P+1}\|_S^2 - \|\delta x_{i+P}\|_S^2 + \|\delta x_{i+P}\|_Q^2 + \|\delta u_{i+P-1}^*\|_R^2 - \left(\|\delta x_i^*\|_Q^2 + \|\delta u_i^*\|_R^2 \right)
\end{aligned}$$

En utilisant l'hypothèse H11, on a

$$\Delta J_{i+1} \leq -\left(\|\delta x_i^*\|_Q^2 + \|\delta u_i^*\|_R^2 \right) \leq -\|\delta x_i^*\|_Q^2$$

Soit

$$V_3(x_i) = V_1(x_i) = \|\delta x_i^*\|_Q^2$$

V_3 est une fonction définie positive. De plus, par optimalité on a : $\Delta J_{i+1}^* \leq \Delta J_{i+1}$. On obtient pour la variation du coût optimal

$$\Delta J_{i+1}^* \leq -V_3(x_i) \tag{4.21}$$

Le taux de variation de J_i^* est donc semi-défini négatif.

3. Le lemme 4.4 donne $J_i^* \leq \|\delta x_i\|_S^2, \quad \forall \|\delta x_i\|_S^2 \leq r_i$. $V_2(x) = \|x\|_S^2$ étant une fonction définie positive, on en conclut que J_i^* est majorée par une fonction définie positive.

J_i^* est ainsi une fonction de Lyapunov pour le système variant dans le temps (TanLd). On en conclut la stabilité uniforme du système en $\delta x = 0$ pour tout δx_i tel que $\|\delta x_i\|_S^2 \leq r_i$.

∎

4.2.4.3 Satisfaction de l'hypothèse H11

On va maintenant chercher des matrices S et K_k satisfaisant l'hypothèse H11. Les propriétés utiles à la démonstration, traitant des inégalités matricielles, sont rappelées en Annexe B (complément de Schur, congruence de matrices, résolution simultanée de plusieurs LMI) . Dans la partie précédente, on a montré qu'il était possible de conclure à la stabilité du système si l'on fait l'hypothèse H11. On montre maintenant comment il est possible de construire K_{i+P} et S pour que cette hypothèse soit toujours vérifiée. Sans perte de généralité et pour simplifier la notation, on remplace l'indice $i+P$ par un indice $k \in \overline{P..H}$

$$\begin{aligned}
&\|\delta x_{k+1}^K\|_S^2 - \|\delta x_k\|_S^2 + \|\delta x_k^K\|_Q^2 + \|K_k \delta x_k\|_R^2 \leq 0 \\
\Leftrightarrow &\|(A_k + B_k K_k)\delta x_k\|_S^2 - \|\delta x_k\|_S^2 + \|(A_k + B_k K_k)\delta x_k\|_Q^2 + \|K_k \delta x_k\|_R^2 \leq 0 \\
\Leftrightarrow &\, \delta x_k^T \left((A_k + B_k K_k)^T S (A_k + B_k K_k) - S + Q + K_k^T R K_k \right) \delta x_k \leq 0
\end{aligned}$$

La dernière inéquation est toujours vérifiée si $\forall k \in \overline{P..H}$ les inégalités matricielles suivantes sont vérifiées.

$$\begin{aligned}
&(A_k + B_k K_k)^T S (A_k + B_k K) - S + Q + K_k^T R K_k \leq 0 \\
\Leftrightarrow &-S - \begin{pmatrix} (A_k + B_k K_k)^T & K_k^T & I_{n_x} \end{pmatrix} \begin{pmatrix} -S & 0 & 0 \\ 0 & -R & 0 \\ 0 & 0 & -Q \end{pmatrix} \begin{pmatrix} A_k + B_k K_k \\ K_k \\ I_{n_x} \end{pmatrix} \preceq 0
\end{aligned}$$

En appliquant le complément de Schur (B.1), on obtient :

$$M_k = \begin{pmatrix} -S & (A_k + B_k K_k)^T & K_k^T & I_{n_x} \\ (A_k + B_k K_k) & -S^{-1} & 0 & 0 \\ K_k & 0 & -R^{-1} & 0 \\ I_{n_x} & 0 & 0 & -Q^{-1} \end{pmatrix} \preceq 0, \quad \forall k \in \mathbb{N} \tag{4.22}$$

Si S est une variable d'optimisation, l'inégalité (4.22) n'est pas une LMI car elle fait intervenir S et son inverse. Cependant, il est possible de se ramener à une LMI en utilisant la propriété de congruence. On commence par rappeler que $S^T = S$ car S est par hypothèse symétrique. La matrice M_k peut être factorisée de la façon suivante :

$$M_k = L^T \underbrace{\begin{pmatrix} -S^{-1} & S^{-1}A_k^T + S^{-1}K_k^T B_k^T & S^{-1}K_k^T & S^{-1} \\ A_k S^{-1} + B_k K_k S^{-1} & -S^{-1} & 0 & 0 \\ K_k S^{-1} & 0 & -R^{-1} & 0 \\ S^{-1} & 0 & 0 & -Q^{-1} \end{pmatrix}}_{\tilde{M}_k} L \preceq 0, \quad \forall k \in \mathbb{N}$$

Avec

$$L = \begin{pmatrix} S & 0 & 0 & 0 \\ 0 & I_{n_x} & 0 & 0 \\ 0 & 0 & I_{n_u} & 0 \\ 0 & 0 & 0 & I_{n_x} \end{pmatrix}$$

On pose $Y_k = K_k S^{-1}$ et $Z = S^{-1}$. Ainsi

$$\tilde{M}_k(A_k, B_k, Y_k, Z) = \begin{pmatrix} -Z & Z^T A_k^T + Y_k^T B_k^T & Y_k^T & S^{-1} \\ A_k Z + B_k Y_k & -Z & 0 & 0 \\ Y & 0 & -R^{-1} & 0 \\ Z & 0 & 0 & -Q^{-1} \end{pmatrix}$$

L est une matrice diagonale par bloc composée de matrices semi-définies positives. Par conséquent L est semi-définie positive, et l'on peut utiliser la propriété de congruence introduite dans l'Annexe B pour M_k et $\tilde{M}_k$.

$$M_k \prec 0 \Leftrightarrow \tilde{M}_k = \tilde{M}(A_k, B_k, Y_k, Z) \prec 0 \tag{4.23}$$

Les inégalités $\tilde{M}_k \prec 0$ sont des inégalités matricielles en Y_k et Z et peuvent être résolues. Cependant, la difficulté de cette formulation réside dans le fait qu'il faut garantir les inégalités pour tous les instants k, et donc pour tous les couples (A_k, B_k) rencontrés, ce qui d'après le Lemme B.3 revient à :

$$D_M = diag(\tilde{M}_0, \tilde{M}_1, \ldots, \tilde{M}_{H-N}) \prec 0$$

Dans la pratique, cela nécessiterait d'évaluer à l'avance toutes les matrices (A_k, B_k) le long de la trajectoire $\mathtt{P}_r$ au pas d'échantillonnage du MPC et de résoudre $D_M \prec 0$. Cette solution ne semble pas envisageable en raison de la quantité d'information à stocker et du nombre d'inconnues résultantes dans la LMI. On introduit donc les hypothèses H12 et H13 qui sont similaires à celles proposées dans [69] pour rendre le problème de taille plus raisonnable.

Hypothèse H12. *On connaît un ensemble de matrices sommets*

$$\mathcal{C}(A) = \{\mathbf{A}^n, \quad n \in \overline{1..n_A}\}$$

définissant l'enveloppe convexe des matrices $A_k, \quad k \in \overline{N..H}$ *:*

$$A_k = \sum_{n=1}^{n_A} \bar{\xi}_{k,n} \mathbf{A}^n, \quad \sum_{n=1}^{n_A} \bar{\xi}_{k,n} = 1 \tag{4.24}$$

Où $\bar{\xi}_k$*, de composantes* $\bar{\xi}_{k,n}$ *est la réalisation particulière de* ξ *correspondant à l'instant* k*.*

Hypothèse H13. *La matrice* B_k *est constante. On pose alors* $B_k = B$

On montre que sous ces deux hypothèses, il est suffisant de se limiter à la résolution pour les matrices sommets :

$$M(\mathbf{A}^n, \mathbf{B}^n, \mathbf{Y}^n, Z) \prec 0, \quad \forall n \in \overline{1..n_A} \tag{4.25}$$

On obtient ainsi n_A contrôleurs $\mathbf{K}^n = \mathbf{Y}^n S$. Un contrôleur final garantissant le respect de (4.23) pour tout $A_k \in co(A)$ peut alors être construit comme une combinaison linéaire des $\mathbf{K}^n$.

Lemme 4.6. *Sous les hypothèses H13,H12, si l'inéquation (4.25) admet une solution* $\{Z, Y_k, \quad k \in \overline{1..n_A}\}$ *, avec* $\mathbf{K}^n = \mathbf{Y}^n Z$ *et* $S = Z^{-1}$*, alors pour tout instant* k*, le retour d'état donné par*

$$K_k = \sum_{n=1}^{n_A} \xi_n \mathbf{K}^n, \text{ où } \xi \text{ vérifie (4.24)}$$

satisfait l'hypothèse H11.

Démonstration. Par hypothèse, et en utilisant l'équivalence (4.23), S et les $\mathbf{K}^n$ ont été déterminés et satisfont les inéquations

$$M^n = \begin{pmatrix} -S & (\mathbf{A}^n + B\mathbf{K}^n)^T & \mathbf{K}^{nT} & I_{n_x} \\ \mathbf{A}^n + B\mathbf{K}^n & -S^{-1} & 0 & 0 \\ \mathbf{K}^n & 0 & -R^{-1} & 0 \\ I_{n_x} & 0 & 0 & -Q^{-1} \end{pmatrix} \preceq 0, \quad \forall n \in \overline{1..n_A}$$

On a $K_k = \sum_{n=1}^{n_A} \bar{\xi}_{k,n}\mathbf{K}^n$. Par hypothèse, $\bar{\xi}_k \succ 0$. On peut donc écrire l'inégalité suivante :

$$\begin{aligned}
&\sum_{n=1}^{n_A} \bar{\xi}_{k,n} M^n \prec 0 \\
\Rightarrow &\begin{pmatrix}
-\sum_{n=1}^{n_A} \bar{\xi}_{k,n} S^{-1} & * & * & * \\
\sum_{n=1}^{n_A} \bar{\xi}_{k,n}\mathbf{A}^n + B(\sum_{n=1}^{n_A} \bar{\xi}_{k,n} K^n) & -S^{-1} & * & * \\
\sum_{n=1}^{n_A} \bar{\xi}_{k,n}\mathbf{K}^n & 0 & -R^{-1} & * \\
I_{n_x} & 0 & 0 & -\sum_{m=1}^{n_A} \xi_m Q^{-1}
\end{pmatrix} \prec 0 \\
\Leftrightarrow &\begin{pmatrix}
-S^{-1} & * & * & * \\
A_k + BK_k) & -S^{-1} & * & * \\
K_k & 0 & -R^{-1} & * \\
I_{n_x} & 0 & 0 & -\sum_{m=1}^{n_A} \xi_m Q^{-1}
\end{pmatrix} \preceq 0
\end{aligned} \tag{4.26}$$

L'équation (4.22) est donc satisfaite pour toutes les réalisations de A_k. Par conséquence, le retour d'état

$$\mathcal{K}_f(\delta x, k) = K_k \delta x = \sum_{n=1}^{n_A} \bar{\xi}_{k,n}\mathbf{K}^n, \quad \text{avec } \bar{\xi}_k \text{ tel que } \quad \sum_{n=1}^{n_A} \bar{\xi}_{k,n} = 1, \quad \sum_{n=1}^{n_A} \bar{\xi}_{k,n}\mathbf{A}^n$$

assure le respect de l'hypothèse H7 pour toutes les valeurs de A_k. ∎

4.2.4.4 Satisfaction de l'hypothèse H10 : ensemble terminal

On cherche ici à construire les ensembles terminaux Ω_{i+P} pour assurer le respect récursif des contraintes. On commence par poser ici certaines hypothèses supplémentaires sur le domaine des contraintes :

Hypothèse H14. *Les ensembles $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ et $\mathcal{U} \subset \mathbb{R}^{n_u}$ sont polytopiques. On a*

$$\mathcal{X} = \{x | M_x x \leq b_x\} \quad \mathcal{U} = \{u | M_u u \leq b_u\} \tag{4.27}$$

Il est également possible de définir ces ensembles de façon locale autour des trajectoires de référence. Les ensembles selon cette définition deviennent variant dans le temps.

$$\mathcal{X} \equiv \delta\mathcal{X}^i = \{\delta x | M_x(x_{r,i} + \delta x) \leq b_x\} = \{\delta x | M_x \delta x \leq b_{x,i}\} \tag{4.28}$$

$$\mathcal{U} \equiv \delta\mathcal{U}^i = \{\delta u | M_u(u_{r,i} + \delta u) \leq b_u\} = \{\delta u | M_u \delta u \leq b_{u,i}\} \tag{4.29}$$

Avec $b_{x,i} = b_x - M_x x_{r,i}$ et $b_{u,i} = b_x - M_u u_{r,i}$.

Les ensembles polytopiques permettent notamment de modéliser des contraintes de type bornes sur les variables, ce qui est typiquement le genre de contraintes que l'on rencontre. La méthode [71] évoquée précédemment pour des modèles linéarisés à temps continu permet de

définir des ensembles terminaux $\Omega(t)$ qui garantissent la satisfaction récursive des contraintes mais nécessite pour cela que la trajectoire de référence se maintienne à un état stationnaire (x_s, u_s) à partir d'un instant T fini ($x(t \geq T) = x_s$, et $f(x_s, u_s) = 0$) . Comme ce n'est pas le cas des trajectoires rencontrées dans les applications considérées ici, on se limitera à l'étude de l'invariance à un ou à V pas, c'est-à-dire en construisant un ensemble terminal qui garantit la faisabilité pour les V problèmes suivants.
On suppose que les n_A sommets de l'enveloppe convexe des matrices A_k sont connus et ont permis de déterminer les n_A retours d'état $\mathbf{K}^n$ par résolution de (4.22) ainsi que la matrice S utilisée pour définir la fonction de coût terminal $\Phi(\delta x) = \|\delta x\|_S$. Pour chaque point $x_{r,i+P}$ on peut donc calculer le vecteur ξ_{i+P} tels que

$$A_{i+P} = \sum_{n=1}^{n_A} \xi_{i+P,n} \mathbf{A}^n$$

puis le contrôleur linéaire décrit dans le Lemme 4.6 :

$$K_{i+P} = \sum_{n=1}^{n_A} \xi_{i+P,n} \mathbf{K}^n$$

On rappelle que l'ensemble terminal est défini par

$$\delta\Omega_{i+P} = \left\{ \delta x_{i+P} | \quad \|\delta x_{i+P}\|_S^2 \leq r_{i+P} \right\}$$

$\delta\Omega$ peut donc aussi être décrit par la trajectoire de r. On souhaite ainsi déterminer r_{i+P} tel que $\delta\Omega_{i+P}$ satisfasse les trois conditions suivantes :

1. $\delta\Omega_{i+P}$ est admissible :
$$\delta\Omega_{i+P} \subset \delta\mathcal{X} \tag{4.30}$$
2. Le contrôleur final $\mathcal{K}_f(\delta x, i+P) = K_{i+P}\delta x$ est admissible sur $\delta\Omega_{i+P}$:
$$\forall \delta x \in \delta\Omega_{i+P}, \begin{cases} K_{i+P}\delta x & \in \delta\mathcal{U}^{i+P} \\ (A_{i+P} + BK_{i+P})\delta x & \in \delta\mathcal{X}^{i+P} \end{cases} \tag{4.31}$$
3. $\delta\Omega^i_{i+P}$ est positivement invariant à un pas sous le contrôleur $\mathcal{K}_f$:
$$(A_{i+P} + BK_{i+P})\delta x \in \delta\Omega_{i+P+1} \quad \forall \delta x \in \delta\Omega_{i+P} \tag{4.32}$$

Calcul du plus grand ensemble terminal admissible en x :

L'ensemble terminal $\delta\Omega_{i+P} = \{\delta x_{i+P} | \quad \|\delta x_{i+P}\|_S^2 \leq r_{i+P}\}$ correspond à un ellipsoïde de rayon r_{i+P} en norme S. Pour contraindre le problème le moins possible, on cherche ainsi la valeur maximale de r_{i+P} permettant de satisfaire (4.30-4.32). Soit R_{i+P} la distance entre le point $x_{r,i+P}$ et l'ensemble admissible $\mathcal{X}$ en norme S. Ainsi, $\Omega_{i+P} \subset \mathcal{X}$ si $r_{i+P} \leq R_{i+P}$. Graphiquement, calculer R_{i+P} revient à chercher le plus petit ellipsoïde (en norme S) qui touche la frontière de $\mathcal{X}$ (première activation d'une contrainte). On traduit cette contrainte en introduisant des variables d'écart aux contraintes $s^{(j)}$, j étant l'index de la contrainte dans les matrices M_x et M_u définies en (4.27). On note e_j le vecteur unitaire dont la j^{me} composante vaut 1. R_{i+P} est ainsi la valeur de la fonction objectif du problème d'optimisation suivant :

$$\begin{array}{ll} \max\limits_{s,\delta x_{i+P}} & R_{i+P} = \|\delta x_{i+P}\|_S^2 \\ \text{s.c.} & e_j^T M_x \delta x_{i+P} + s^{(j)} - b^{(j)}_{x,i+P} = 0 \\ & s^{(j)} \geq 0, \quad \forall j \in \overline{1..n_c} \end{array}$$

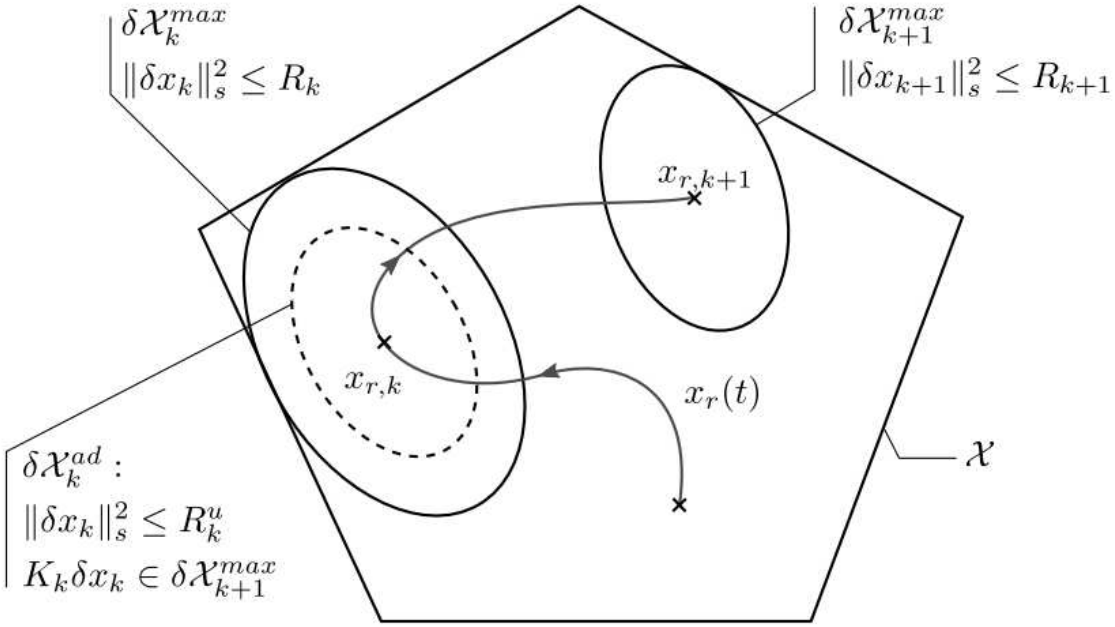


FIGURE 4.2 – Ensemble terminal invariant à un pas

On nomme cet ellipsoïde maximale admissible en x

$$X_{i+P}^{max} = \{\delta x_{i+P} | \quad \|\delta x_{i+P}\|_S^2 \leq R_{i+P}\} \tag{4.33}$$

Cependant, comme il existe des contraintes sur u, l'ensemble admissible pour le contrôleur terminal peut être plus petit que $\mathcal{X}_{i+P}^{max}$. On recherche la plus grande approximation en norme S de cet ensemble admissible, que l'on note $\mathcal{X}_{i+P}^{ad}$:

$$\mathcal{X}_{i+P}^{ad} = \{\delta x_{i+P} \in \mathcal{X}_{i+P}^{max} \quad | \quad \|\delta x_{i+P}\|_S^2 \leq R_{i+P}^u\} \tag{4.34}$$

Ces différents ensembles sont représentés sur la Figure 4.2. Les conditions (4.31) reviennent à résoudre le problème d'optimisation suivant.

$$\begin{aligned}
\max_{\delta x_{i+P}, \mathbf{s}} \quad & R_{i+P}^u = \|\delta x_{i+P}\|_S^2 \\
\text{s.c.} \quad & e_j^T M_x \delta x_{i+P} + s^{(j)} - b_{x,i+P}^{(j)} = 0 \\
& \delta x_{i+P}^T (A_{i+P} + BK_{i+P})^T S (A_{i+P} + BK_{i+P}) \delta x_{i+P} \leq R_{i+P+1} \\
& M_u K_{i+P} \delta x_{i+P} \leq b_{\delta u, i+P} \\
& s^{(j)} \geq 0, \quad \forall j \in \overline{1..n_c}
\end{aligned}$$

Il s'agit d'un problème SOCP (*Second Order Cone Programming*) qui peut être résolu par des solveurs d'optimisation convexe tels CVX [75][76] .

4.2.5 Conclusions et perspectives pour l'analyse de stabilité

La partie 4.2.4 a fourni une analyse de stabilité pour le système sous commande TL-MPC dans le cas nominal H8, qui suppose que le système réel se comporte exactement comme le

modèle (TanLd). Cependant, il existe plusieurs sources d'erreur responsables d'une différence entre les prédictions du modèle interne du MPC et leur réalisation. La première concerne les perturbations w (entrées non contrôlées) pour lesquelles on ne dispose que de prévisions. Une seconde source d'erreur provient de l'erreur de modélisation du modèle interne. Il existe ainsi dans la littérature certaines formulations modifiées du MPC permettant de prendre en compte ces incertitudes dans l'algorithme afin d'en accroitre la robustesse, l'idée étant de garantir le respect des contraintes pour toutes les réalisation de l'incertitude. Or trouver une commande admissible dans ce cas peut être très conservatif et dégrader l'optimalité ou le domaine de faisabilité du problème [77]. Pour cette raison, au lieu de calculer une séquence de commande U^i_{rob} certaines formulations MPC dites en boucle fermée calculent une séquence de lois de commandes de la forme $u_k = \mathcal{U}(x_k, w_k)$ (paramétrées la plupart du temps pour limiter la complexité de la formulation).
Le MPC min-max [77][78] envisage quant à lui le problème de commande comme un jeu entre deux adversaires : le contrôleur u qui cherche à minimiser la fonction objectif et les perturbations w qui cherchent à la maximiser.
En raison de l'incertitude, la convergence asymptotique vers un point ne peut plus être assurée. Les notions de stabilité présentées dans le paragraphe précédent sont alors adaptées en conséquence : on se concentre plutôt sur la convergence vers un domaine borné dépendant de la taille des perturbations. Ces exigences sont formalisées dans la notion de *stabilité entrée-état* ISS (*Input to State Stability*), ou de stabilité ISpS (*Input to State Practical Stability*), qui autorisent de plus la présence d'erreurs persistantes). Ces nouvelles définitions de la stabilité font appel à des fonctions de Lyapunov modifiées nommées fonctions de Lyapunov ISS (respectivement ISpS).

Stabilité Entrée Etat

On a fait l'hypothèse en (4.15) que l'erreur entre les prédictions du modèle (TanLd) et le système réel pouvait s'exprimer comme une perturbation additive **d** : $x_{k+1} = \bar{x}_{k+1} + d_k$.

Définition (*Stabilité IsPS* ([79])) Le système sous commande MPC (4.14) est localement ISS si il existe des constantes non négatives r, d_M, D, une fonction β de classe $\mathcal{K}$ et une fonction γ de classe $\mathcal{K}$ telles que

$$\|x_k\| \leq \beta(\|x_0\|, k) + \gamma(\max_{n \in \overline{0..k}} \|d_n\|) + D$$

pour toute valeur initiale $\|\delta x_0\| \leq r$ et toute perturbation additive $\|d_k\| \leq d_M$

Les définitions des fonctions de classe $\mathcal{K}$, $\mathcal{K}^\infty$ et $\mathcal{KL}$ sont données en Annexe B.

L'erreur d contient en fait deux facteurs distincts :

- l'erreur de modélisation : même en l'absence de perturbations, le modèle interne (TanLd) diffère du système réel (4.1).
- la présence de perturbations δw

Comme la notion de stabilité ISpS fait intervenir la taille de l'erreur, il est nécessaire de pouvoir borner celle-ci. On présente donc en Annexe D un premier résultat permettant de quantifier l'erreur de modélisation sous certaines hypothèses. La quantification de l'incertitude due aux perturbations δw, et les autres pistes évoquées feront quant à elles l'objet de travaux futurs visant à établir une preuve de stabilité dans le cas perturbé.

4.3 Suivi de trajectoires optimales pour les centrales de production d'énergie

4.3.1 Difficultés soulevées par l'architecture dans le cas idéal

On présente dans cette partie une architecture MPC légèrement différente de celle introduite précédemment pour des raisons que l'on justifie ci-après.
Tout d'abord, il peut être difficile en pratique d'obtenir des trajectoires de référence $\mathtt{P}_r = (x_r, u_r, w_r)$ qui soient solutions de (NLc). D'une part, le modèle d'optimisation simplifié et le modèle de commande ne partagent pas nécessairement le même vecteur d'état ni d'ailleurs le même vecteur de commande. Les trajectoires optimales $\mathtt{P}^*$ du modèle d'optimisation simplifié sont ainsi données le plus souvent pour certaines commandes u^* et sorties y^* qu'il va falloir associer à des grandeurs du modèle de commande (NLc). Dans l'exemple de la centrale de cogénération, si l'on utilise un modèle de PLNE comme modèle d'optimisation, les commandes u^* étaient les puissances de combustible, les sorties y_r^* les puissances thermiques et électriques produites, et l'état x_r^* la charge du stock. Dans le modèle non linéaire en revanche, les commandes u sont non seulement les puissances de combustible mais aussi les débits dans les différentes branches. De même, l'état x a une dimension plus importante dans le modèle (NLc) : par exemple l'état du stockage qui était de dimension 1 dans le modèle de PLNE est désormais décrit par l'état thermodynamique (P_i, h_i) du fluide dans chacune des strates i modélisées.
Une première approche consiste ainsi à reconstruire une trajectoire de référence x_r en l'état pour le modèle non linéaire d'après $\mathtt{P}^*$. Cet aspect a déjà été évoqué en partie 3.5.1 dans un autre contexte (on cherchait alors à ne reconstruire que y_r et u_r, mais x_r s'obtient de la même façon). En d'autres termes, il s'agit de résoudre un problème d'inversion sur le modèle non linéaire. Dans l'exemple précédent, la taille du modèle permettait encore d'appliquer cette méthode. Cependant dans la plupart des cas, et en particulier pour les modèles faisant intervenir des tables de propriétés, la complexité du modèle non linéaire rend la résolution de ce problème inverse trop complexe.
Pour cette raison, dans cette seconde implémentation du MPC, on considère désormais des trajectoires de référence (y_r, u_r) données pour la sortie et la commande plutôt que de chercher à obtenir des trajectoires de références (x_r, u_r) données pour l'état et la commande. Ces trajectoires de référence associées au trajectoires (u^*, y^*) du modèle d'optimisation sont fournies par le bloc "*Génération de références admissibles*" montré en Figure 1.1. On lève donc ici la restriction que ces trajectoires de référence soient des trajectoires solution de (NLc).

Le fait que x_r ne soit plus défini soulève deux problèmes : d'une part cette donnée est nécessaire dans la preuve de stabilité énoncée précédemment pour définir un ensemble terminal, d'autre part la linéarisation ne peut plus se faire autour de x_r car celui-ci est désormais inconnu. On propose donc une autre définition pour la trajectoire nominale qui est utilisée pour dériver les modèles linéarisés tangents. Désormais, la trajectoire nominale $\mathtt{P}_{nom}^i$ à l'instant t_i correspond à la simulation du modèle non linéaire sur l'horizon de prédiction du MPC avec pour entrée (u_r, v_r, w_r) et pour point initial l'état courant (estimé ou mesuré) $\hat{x}_{i+1}$. Elle correspond à l'évolution prédite du système en boucle ouverte.
En accord avec cette définition, une différence majeure avec la trajectoire nominale $\mathtt{P}_{nom} = \mathtt{P}_r$ utilisée précédemment est que celle-ci change à chaque itération, même dans le cas d'un mo-

dèle parfait sans perturbation. En effet, au pas de temps i, une commande différente de la commande nominale, $u_i \neq u_{r,i}$ est appliquée au système. De fait, le prochain état diffère lui aussi de sa valeur nominale $x_{i+1} = x(x_i, u_i; t_{i+1}) \neq x^i_{nom,i+1}$. En supposant que pour un modèle parfait le point de départ de la prochaine trajectoire nominale est $\hat{x}_{i+1} = x_{i+1}$ on constate que $\mathtt{P}^{i+1}_{nom}$ et $\mathtt{P}^{i}_{nom}$ ne sont pas concordantes (ne partagent pas les mêmes points de passage). Cette particularité de non concordance des trajectoires nominales entre deux problèmes MPC consécutifs ne pose pas de problème dans l'implémentation pratique mais complique les analyses de stabilité, car les linéarisations se font autour de points différents. En première approche, on veillera ainsi à choisir un pas d'échantillonnage suffisamment fin et une adaptation δu de la commande suffisamment limitée pour que les trajectoires nominales $\mathtt{P}^{i}_{nom}$ et $\mathtt{P}^{i+1}_{nom}$ ne s'éloignent pas trop. De même, on supposera que les trajectoires $\mathtt{P}_{nom}$ et $\mathtt{P}_r$ sont suffisamment proches pour que l'approximation par un modèle linéarisé tangent dans un voisinage de $\mathtt{P}_r$ reste acceptable.

4.3.2 Architecture TL-MPC adaptée aux centrales d'énergie

4.3.2.1 Architecture du MPC

Dans cette architecture modifiée, on dispose seulement d'une trajectoire de référence pour les entrées u_r et les sorties y_r. L'obtention de modèles linéarisés tangent autour de x_r n'est alors plus possible, et le MPC s'appuie désormais sur des modèles linéarisés tangents de la centrale non plus autour de x_r, mais autour d'une trajectoire nominale. La trajectoire nominale utilisée est désormais recalculée à chaque instant d'échantillonnage t_i : elle correspond à celle prédite par le modèle non linéaire auquel on appliquerait u_r en boucle ouverte, en partant de l'état courant estimé ($x^i_{nom}(t_i) = \hat{x}_i$).

On fait ici l'hypothèse que la trajectoire de référence est suffisamment proche de la trajectoire nominale. La trajectoire nominale des états est nommée x_{nom}, celle des sorties y_{nom}. On remarque alors que pour les entrées uniquement, on a l'équivalence u_r. Les grandes lignes de l'algorithme proposé, illustrées par la Figure 4.3 sont les suivantes :

- *Initialisation* : les trajectoires de référence sont obtenues pour les commandes (u_r) et les sorties (y_r) sur un grand horizon H. Cette étape se fait hors ligne.
- *Étape 1* : Un estimateur d'état évalue l'état actuel du modèle Modelica d'après les mesures faites sur la centrale réelle. L'estimateur utilise le modèle non linéaire en phase de prédiction et les matrices de linéarisation pour la mise à jour du filtre et du gain d'observation.
- *Étape 2* : Le modèle Modelica est simulé avec u_r en entrée pour prédire les trajectoires nominales des sorties et obtenir les matrices de linéarisations du modèle autour de cette trajectoire. Cette étape se fait sur un horizon de prédiction $P \ll H$.
- *Étape 3* : Le MPC adapte les commandes à appliquer au système et trouve un compromis entre le suivi des entrées et des sorties sur l'horizon P.
- *Étape 4* : Seule la première commande calculée à l'étape 3 est appliquée au modèle, et les étapes 1 à 4 sont répétées.

La Figure 4.4 montre la trajectoire de référence (trait plein) et la trajectoire corrigée par le MPC (croix) des entrées (u_{nom} et u). La trajectoire nominale de la sortie (tiret-pointillé) calculée à l'instant t_i correspond au résultat de simulation du modèle non linéaire avec les commandes u_{nom}. Cette trajectoire a pour condition initiale $x^i_{nom,i} = x_{NL,i}$, et l'on peut voir

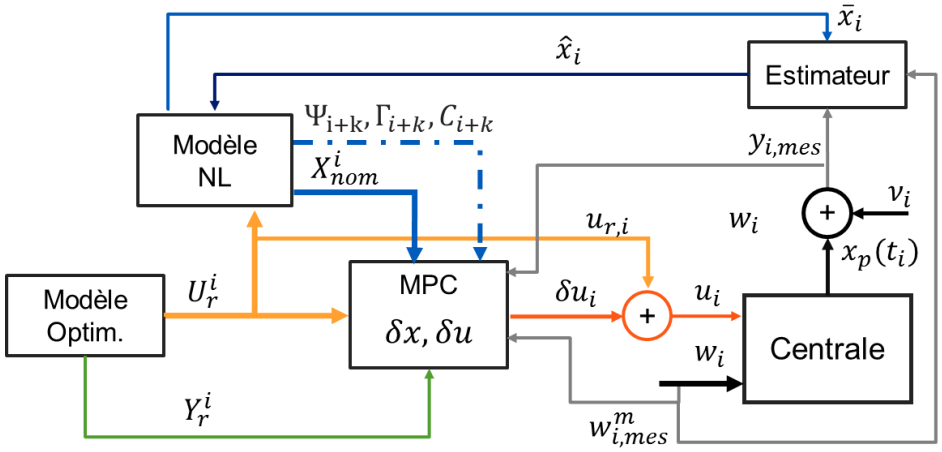


FIGURE 4.3 – Architecture du MPC linéarisé tangent

qu'elle diffère de la trajectoire de référence pour la sortie (trait plein). Enfin, la prédiction échantillonnée du MPC pour les P prochains pas est matérialisée par les croix bleues.

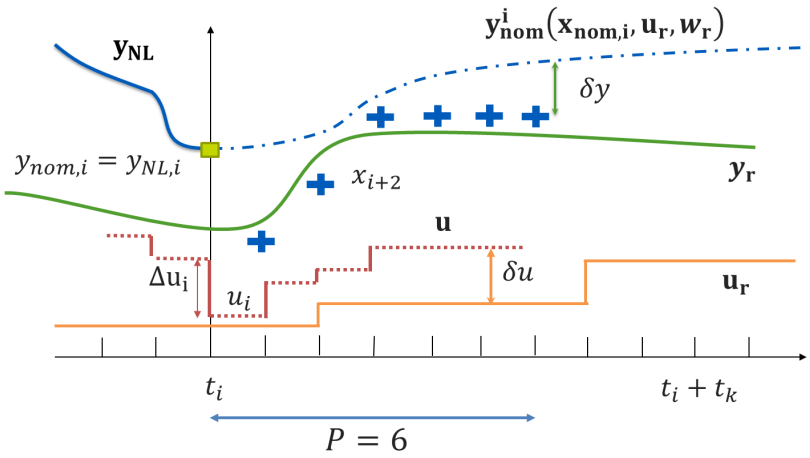


FIGURE 4.4 – Définition des trajectoires d'entrée et de sortie (cas 1 entrée, 1 sortie)

4.3.2.2 Problème d'optimisation MPC

En tenant compte des remarques précédentes, la fonction objectif (4.4) est modifiée pour pénaliser les écarts à la référence des variables de sorties. On introduit alors le vecteur des sorties prédites (original et en variations) qui remplacera maintenant le vecteur des états prédits dans la nouvelle définition du critère et de la fonction objectif :

$$Y^i = \begin{pmatrix} y_{i+1} \\ \vdots \\ y_{i+P} \end{pmatrix}, \quad \delta Y^i = \begin{pmatrix} \delta y_{i+1} \\ \vdots \\ \delta y_{i+P} \end{pmatrix}$$

L'indice commence ici à $i+1$ car la sortie y_i à t_i n'est pas influencée par la commande et peut être retirée du problème d'optimisation MPC_i. De même, on considère les sorties jusque $i+P$ en raison de l'absence du coût terminal (en d'autres termes $S = Q_y$ ici). On considère désormais la fonction objectif suivante :

$$J_i = \|U^i - U_r^i\|_{Q_u}^2 + \|Y^i - Y_r^i\|_{Q_y}^2 + \|\Delta U^i\|_R^2 \tag{4.35}$$

$$= \|\delta_U^i\|_{Q_u}^2 + \|\delta Y^i + Y_{nom}^i - Y_r^i\|_{Q_y}^2 + \|\Delta U^i\|_R^2 \tag{4.36}$$

Les contraintes suivantes sont considérées. Il s'agit de contraintes de bornes

- Des bornes sur les commandes $u_{min} \leq u \leq u_{max}$
- Des bornes sur les sorties $y_{min} \leq y \leq y_{max}$
- Des bornes sur l'écart à la trajectoire nominale $\delta u_{min} \leq \delta u \leq \delta u_{max}$ et $\delta y_{min} \leq \delta y \leq \delta y_{max}$

Composants extinguibles :
Dans les cas d'application qui nous intéressent, certains composants peuvent être allumés ou éteints. Les commandes correspondantes doivent alors souvent respecter des contraintes du type "*éteint (0) ou compris dans une plage de fonctionnement* $[lmin, lmax]$". On définit n_u Booléens $extinguible_s$ indiquant si la composante u_s du vecteur u doit respecter ce type de contrainte. On suppose que le MPC ne remet pas en cause les décisions d'engagement d'un composant, ce qui reviendrait à changer le mode du système. Or dans ce cas, l'utilisation d'un modèle linéarisé tangent, valable localement autour d'une trajectoire nominale, ne peut plus se justifier. Ainsi, si une variable extinguible a une valeur de référence nulle pour un pas de temps donné, le MPC n'a pas la liberté de moduler cette valeur. En revanche, si le composant est allumé, il devra respecter la plage de fonctionnement $[lmin, lmax]$. Cette fonctionnalité est réalisée en modifiant de façon dynamique les bornes des variables dans le problème d'optimisation. On introduit les Booléens $OFF_{s,k} = \{u_{s,r,k} = 0\}$. Les bornes des variables u_s s'expriment comme :

$$u_{s,min,k} = \begin{cases} 0 & \text{si} \quad \text{extinguible}_s \text{ et } OFF_{s,k} \\ \text{lmin} & \text{sinon} \end{cases} \tag{4.37}$$

$$u_{s,max,k} = \begin{cases} 0 & \text{si} \quad \text{extinguible}_s \text{ et } OFF_{s,k} \\ \text{lmax} & \text{sinon} \end{cases} \tag{4.38}$$

Ces contraintes sont reformulées de la sorte :

$$u_{s,min,k} = \text{lmin} * [(1 - OFF_{s,k}) * \text{ extinguible}_s + (1 - \text{ extinguible}_s)]$$

$$u_{s,max,k} = \text{lmax} * [(1 - OFF) * \text{ extinguible}_s + (1 - \text{ extinguible}_s)]$$

4.3.3 Formulation matricielle du MPC

4.3.3.1 Obtention de deux modèles internes augmentés

La présence d'un terme intégrateur dans un modèle linéaire permet d'éviter une erreur statique sur la sortie. Dans cette optique, [61][62] suggèrent d'utiliser dans la formulation MPC la variation de la commande plutôt que la commande elle même. Dans notre cas, cela

revient à utiliser $\Delta\delta u$ comme variable de commande plutôt que δu. Ces deux variables sont reliées par l'intégrateur suivant :

$$\delta u_k = \delta u_{k-1} + \Delta\delta u_k$$

On présente ici deux modèles augmentés utilisant $\Delta\delta u$ comme vecteur de commande. Le premier utilise l'état augmenté $x_{a,k} = \begin{pmatrix} \Delta\delta x_k \\ \delta y_k \end{pmatrix}$, le second utilise $x_{a,k} = \begin{pmatrix} \Delta\delta x_k \\ \delta u_k \end{pmatrix}$. On montre alors comment il est possible de reformuler de façon explicite les contraintes et la fonction objectif du MPC en fonction du nouveau vecteur de commande :

$$\Delta\delta U^i = \begin{pmatrix} \Delta\delta u_i \\ \vdots \\ \Delta\delta u_{i+P-1} \end{pmatrix}$$

Expression des entrées

On commence dans un premier temps par exprimer les termes relatifs aux commandes en fonction du vecteur d'optimisation $\Delta\delta U^i$, car ils seront similaires dans les deux modèles augmentés. Les expressions pour les sorties diffèrent quant à elles et seront présentées séparément pour chacun de ces modèles. Les expressions pour les termes relatifs aux commandes peuvent être données sous forme vectorielle (pour un pas de temps), ou matricielle (pour tous les pas de temps de l'horizon), et sont synthétisés dans la Table 4.1. On montre ci-dessous le détail des calculs :
Pour le pas de temps k, on a :

$$\begin{aligned} \Delta\delta u_k &= \delta u_k - \delta u_{k-1} = u_k - u_{r,k} - u_{k-1} + u_{r,k-1} \\ &= \Delta u_k - \Delta u_{r,k} \end{aligned}$$

$$\sum_{n=0}^{k} \Delta u_{r,n} = u_{r,0} - u_{r,-1} + u_{r,1} - u_{r,0} + \cdots + u_{r,k} - u_{r,k-1} = u_{r,k} - u_{r,-1}$$

$$u_{i+k} = u_{i-1} + \sum_{n=0}^{k} \Delta u_{i+n} = u_{i-1} + \sum_{n=0}^{k} (\Delta\delta u_{i+n} + \Delta u_{r,i+n})$$

L'entrée u_{i+k} peut ainsi être ramenée à la variable d'optimisation $\Delta\delta u_{i+k}$ de la façon suivante :

$$\begin{aligned} u_{i+k} &= u_{i-1} + \textstyle\sum_{n=0}^{k} \Delta u_{i+n} \\ &= u_{i-1} + \textstyle\sum_{n=0}^{k} (\Delta\delta u_{i+n} + \Delta u_{r,i+n}) \\ &= u_{i-1} + \textstyle\sum_{n=0}^{k} \Delta\delta u_{i+n} + u_{r,i+k} - u_{r,i-1} \\ &= \delta u_{i-1} + u_{r,i+k} + \textstyle\sum_{n=0}^{k} \Delta\delta u_{i+n} \end{aligned}$$

Pour présenter ce résultat sous forme matricielle, on introduit la matrice carrée L de côté n_u :

$$L = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 1 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 1 & 1 & \dots & 1 \end{pmatrix}$$

Définition	Expression scalaire	Expression matricielle
Commande	$u_{i+k} = u_{i-1} + \sum_{n=0}^{k} (\Delta\delta u_r + \Delta u_{r,n})$	$U^i = \begin{pmatrix} u_{i-1} \\ \vdots \\ u_{i-1} \end{pmatrix} + L\left(\Delta\delta U^i + \Delta U_{n,i}\right)$
	$u_{i+k} = \delta u_{i-1} + u_{r,i+k} + \sum_{n=0}^{k} \Delta\delta u_{r,i+k}$	$U^i = \begin{pmatrix} \delta u_{i-1} \\ \vdots \\ \delta u_{i-1} \end{pmatrix} + U_r^i + L\Delta\delta U^i$
Variation de la commande	$\Delta u_{i+k} = \Delta\delta u_{i+k} + \Delta u_{r,i+k}$	$\Delta U^i = \Delta\delta U^i + \Delta U_r^i$
Écart de la commande au nominal	$\delta u_{i+k} = \delta u_{i-1} + \sum_{n=0}^{k} \Delta\delta u_{i+n}$	$\delta U^i = \begin{pmatrix} \delta u_{i-1} \\ \vdots \\ \delta u_{i-1} \end{pmatrix} + L\Delta\delta U^i$

TABLE 4.1 – Expressions explicites des termes du MPC relatifs aux entrées en fonction de la variable d'optimisation

Modélisation des perturbations

A l'inverse des entrées commandées qui sont des variables d'optimisation, des hypothèses doivent être faites sur l'évolution des entrées non commandées w. Pour les perturbations non mesurées $w \backslash w^m$ et en l'absence d'information supplémentaire, on fait l'hypothèse que $\delta w = 0$. Cette hypothèse est cohérente avec la modélisation utilisée dans l'estimateur d'état, et qui les considère comme un bruit blanc. Pour w^m en revanche, on dispose de mesures qui permettent de raffiner cette hypothèse.
A chaque pas de temps t_i, une nouvelle mesure $w^m_{mes,i}$ des perturbations mesurées w^m est disponible. La trajectoire du modèle non linéaire satisfait ainsi la condition initiale $w^m_i = w^m_{mes,i}$. On peut ainsi calculer à t_i une nouvelle trajectoire nominale pour w^m prenant en compte cette information. Pour le modèle linéarisé tangent, cela se traduit par $\delta w^m_i = w^m_i - w^m_{r,i} = 0$. Pour les pas suivants, on peut par exemple considérer pour $\Delta\delta w^m_{i+k}$ les trois situations suivantes représentées dans la Figure 4.5.

- La perturbation suit sa valeur nominale (prédictions) après i,
- L'écart au nominal à l'instant i se maintient de t_i à t_{i+P}
- La perturbation $w^m_{i,mes}$ mesurée à l'instant i se maintient constante de t_i à t_{i+P}.

Il y a alors deux façons de prendre en compte les perturbations mesurées dans l'architecture de commande :

- On peut générer le vecteur δW^i correspondant à l'hypothèse retenue. Dans ce cas, les contraintes et la fonction objectif devront être ré-exprimées en fonction de ce terme.
- On peut modifier la trajectoire de référence w^m_r à chaque pas de temps : à chaque pas de temps t_i, une nouvelle mesure $w^m_{mes,i}$ des perturbations mesurées w^m_i est disponible. La trajectoire du modèle non linéaire peut alors être redéfinie pour vérifier $w^m_{r,i} = w^m_i = w^m_{mes,i}$, et respecter l'hypothèse retenue pour les pas $i + k$. Les perturbations

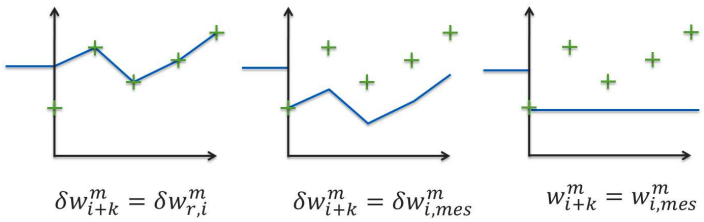


FIGURE 4.5 – Hypothèses sur la trajectoire des perturbations mesurées

mesurées w^m seront alors prises en compte au moment de la génération de la trajectoire nominale, et non plus dans le modèle interne du MPC. On pourra alors, tout comme pour les perturbations non mesurées, faire l'hypothèse $\delta w^m = w^m - w^m_r = 0$

La deuxième méthode a été retenue car elle permet de retirer le terme δw^m des modèles internes augmentés présentés par la suite et simplifie l'expression des contraintes et de la fonction objectif.

Expression du premier modèle interne augmenté

Le premier modèle augmenté est dérivé de celui proposé par[62]. L'état δx_k est remplacé par sa variation entre deux pas d'échantillonnage $\Delta\delta x_k = \delta x_k - \delta x_{k-1}$, et la sortie est ajoutée à l'état. Comme mentionné précédemment, on suppose que les perturbations w suivent leur trajectoire nominale w_r. On a donc $\delta w = 0$, et l'on retirera ces variables du modèle interne.

On sépare par la suite la matrice B en deux matrices B_u et B_w correspondant aux entrées u et w respectivement, de telle sorte que $B_k = [B_{u,k}, B_{w,k}]$. L'évolution de l'état $\Delta\delta x_k$ peut s'exprimer comme :

$$\begin{aligned}
\Delta\delta x_{k+1} &= \delta x_{k+1} - \delta x_k \\
&= A_k\delta x_k + B_k\begin{pmatrix}\delta u_k\\ \delta w_k\end{pmatrix} - \delta x_k + A_k\delta x_{k-1} - A_k\delta x_{k-1} + B_k\begin{pmatrix}\delta u_{k-1}\\ \delta w_{k-1}\end{pmatrix} - B_k\begin{pmatrix}\delta u_{k-1}\\ \delta w_{k-1}\end{pmatrix} \\
&= A_k\Delta\delta x_k + B_k\begin{pmatrix}\delta u_k\\ \delta w_k\end{pmatrix} + \left(A_k\delta x_{k-1} + B_k\begin{pmatrix}\delta u_{k-1}\\ \delta w_{k-1}\end{pmatrix} - \delta x_k\right) \\
&= A_k\Delta\delta x_k + B_k\begin{pmatrix}\delta u_k\\ \delta w_k\end{pmatrix} + (\delta x_k - \delta x_k) \\
&= A_k\Delta\delta x_k + B_{u,k}\Delta\delta u_k + B_{w,k}\Delta\delta w_k
\end{aligned}$$

Par ailleurs, on a pour les sorties

$$\begin{aligned}
\delta y_{k+1} &= c_k\delta x_{k+1} = c_k\left(\delta x_k + \Delta\delta x_{k+1}\right) \\
&= c_k\delta x_k + c_k\left(A_k\Delta\delta x_k + B_{u,k}\Delta\delta u_k + B_{w,k}\Delta\delta w_k\right) \\
&= \delta y_k + c_kA_k\Delta\delta x_k + c_kB_{u,k}\Delta\delta u_k + c_kB_{w,k}\Delta\delta w_k
\end{aligned} \tag{4.39}$$

L'évolution de l'état augmenté $x_{a,k} = \begin{pmatrix}\Delta\delta x_k\\ \delta y_k\end{pmatrix}$ est alors donnée par le modèle d'état suivant :

$$\begin{aligned}\overbrace{\begin{pmatrix}\Delta\delta x_{k+1}\\ \delta y_{k+1}\end{pmatrix}}^{x_{a,k+1}} &= \overbrace{\begin{bmatrix}A_k & 0\\ c_k A_k & I\end{bmatrix}}^{\Psi_k}\begin{pmatrix}\Delta\delta x_k\\ \delta y_k\end{pmatrix} + \overbrace{\begin{bmatrix}B_{u,k}\\ c_k B_{u,k}\end{bmatrix}}^{\Gamma_k}\Delta\delta u_k + \overbrace{\begin{bmatrix}B_{w,k}\\ 0\end{bmatrix}}^{G_k}\Delta\delta w_k\\ \delta y_k &= \underbrace{\begin{bmatrix}0 & I\end{bmatrix}}_{C}\begin{pmatrix}\Delta\delta x_k\\ \delta y_k\end{pmatrix}\end{aligned} \tag{4.40}$$

Expression du second modèle interne augmenté

On retient dans ce cas l'état augmenté suivant suggéré par [61] :

$$x_{a,k} = \begin{pmatrix}\delta x_k\\ \delta u_{k-1}\end{pmatrix}$$

En utilisant les résultats précédents, l'évolution du système augmenté est donnée par

$$\begin{aligned}\overbrace{\begin{pmatrix}\delta x_{k+1}\\ \delta u_k\end{pmatrix}}^{x_{a,k+1}} &= \overbrace{\begin{bmatrix}A_k & B_k\\ 0 & I\end{bmatrix}}^{\Psi_k}\begin{pmatrix}\delta x_k\\ \delta u_{k-1}\end{pmatrix} + \overbrace{\begin{bmatrix}B_{u,k}\\ I\end{bmatrix}}^{\Gamma_k}\Delta\delta u_k + \overbrace{\begin{bmatrix}B_{w,k}\\ 0\end{bmatrix}}^{G_k}\delta w_k\\ \delta y_k &= \underbrace{\begin{bmatrix}c_k & 0\end{bmatrix}}_{C_k}\begin{pmatrix}\delta x_k\\ \delta u_{k-1}\end{pmatrix}\end{aligned} \tag{4.41}$$

Le vecteur U^i des commandes peut alors s'exprimer en fonction de l'état augmenté comme :

$$U^i = \left(\begin{bmatrix}I & 0\end{bmatrix} x_{a,i}\right) + U_r^i + L\Delta\delta U^i$$

4.3.3.2 Expression des sorties

On cherche à ré-exprimer la sortie du modèle augmenté à l'instant $i+k$, car les termes y_{i+k} interviennent dans les contraintes et la fonction objectif. Afin d'alléger la notation, on supposera sans perte de généralité $i = 0$.

On montre ici que pour les deux modèles augmentés, le vecteur des sorties peut s'exprimer à l'aide des matrices S_x^i, S_u^i et S_w^i. Le premier modèle augmenté respecte l'équation :

$$\delta Y_i = S_x^i x_{a,i} + S_u^i \Delta\delta U^i + S_w^i \Delta\delta W^i \tag{4.42}$$

Le second modèle augmenté selon :

$$\delta Y_i = S_x^i x_{a,i} + S_u^i \Delta\delta U^i + S_w^i \delta W^i \tag{4.43}$$

Seule l'expression des matrices $S_\bullet^i$ dépend du modèle interne augmenté que l'on considère. On rappelle que les prédictions du modèle interne sont faites sous l'hypothèse $\delta W^i = \mathbf{0}$. Dans ce cas, le vecteur de sortie s'exprime alors comme :

$$Y_i = Y_{nom}^i + S_x^i x_{a,i} + S_u^i \Delta\delta U^i \tag{4.44}$$

Sorties du premier modèle augmenté

En utilisant les matrices introduites dans le modèle (4.40), on montre comment il est possible d'exprimer la sortie en fonction de $\Delta\delta U^i$ et $\Delta\delta W^i$. On commence ici par donner l'exemple pour l'instant $i+3$:

$$\begin{aligned}
\delta y_3 &= Cx_{a,3} \\
&= C\left[\Psi_2 x_{a,2} + \Gamma_2\Delta\delta u_2 + G_2\Delta\delta w_2\right] \\
&= C\left[\Psi_2\left(\Psi_1\left(\Psi_0 x_{a,0} + \Gamma_0\Delta\delta u_0 + G_0\Delta\delta w_0\right) + \Gamma_1\Delta\delta u_1 + G_1\Delta\delta w_1\right) + \Gamma_2\Delta\delta u_2 + G_2\Delta\delta w_2\right] \\
&= C\left[\Psi_2\Psi_1\Psi_0 x_{a,0} + \Psi_2\Psi_1\left(\Gamma_0\Delta\delta u_0 + G_0\Delta\delta w_0\right) + \Psi_2\left(\Gamma_1\Delta\delta u_1 + G_1\Delta\delta w_1\right)\right. \\
&\quad \left. + \Gamma_2\Delta\delta u_2 + G_2\Delta\delta w_2\right]
\end{aligned}$$

La formule qui généralise cette expression est :

$$\delta y_k = C\left[\left(\prod_{n=1}^{k}\Psi_{k-n}\right)x_{a,0} + \sum_{n=1}^{k}\left[\left(\prod_{m=1}^{k-n}\Psi_{k-m}\right)\left(\Gamma_{n-1}\Delta\delta u_{n-1} + G_{n-1}\Delta\delta w_{n-1}\right)\right]\right]$$

Les matrices S_x^i et S_u^i de l'équation (4.42) sont alors définies de la façon suivante :

$$S_x^i = \begin{pmatrix} C\Psi_i \\ C\Psi_{i+1}\Psi_i \\ \vdots \\ C\Psi_{i+P}\ldots\Psi_{i+1}\Psi_i \end{pmatrix} \tag{4.45}$$

$$S_u^i = \begin{pmatrix} C\Gamma_i & 0 & \ldots & 0 \\ C\Psi_{i+1}\Gamma_i & C\Gamma_{i+1} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C\prod_{n=1}^{P-2}\Psi_{i+P-1-n}\Gamma_i & C\prod_{n=1}^{P-3}\Psi_{i+P-1-n}\Gamma_{i+1} & \ldots & C\Gamma_{i+P-2} \\ C\prod_{n=1}^{P-1}\Psi_{i+P-n}\Gamma_i & C\prod_{n=1}^{P-2}\Psi_{i+P-n}\Gamma_{i+1} & \ldots & C\Gamma_{i+P-1} \end{pmatrix} \tag{4.46}$$

La matrice S_w^i s'obtient en remplaçant Γ par G dans l'équation de S_u^i mais ne sera pas utilisée par la suite en raison de l'hypothèse que les perturbations w suivent leur référence ($\Delta\delta w = 0$).

Sorties du second modèle augmenté

En utilisant le deuxième modèle augmenté, les matrices S_x^i et S_u^i de l'équation (4.43) sont définies de la façon suivante :

$$S_u^i = \begin{pmatrix} C_{i+1}\Gamma_i & 0 & \dots & 0 \\ C_{i+2}\Psi_{i+1}\Gamma_i & C_{i+2}\Gamma_{i+1} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C_{i+P-1}\prod_{n=1}^{P-2}\Psi_{i+P-1-n}\Gamma_i & C_{i+P-1}\prod_{n=1}^{P-3}\Psi_{i+P-1-n}\Gamma_{i+1} & \dots & C_{i+P-1}\Gamma_{i+P-2} \\ C_{i+P}\prod_{n=1}^{P-1}\Psi_{i+P-n}\Gamma_i & C_{i+P}\prod_{n=1}^{P-2}\Psi_{i+P-n}\Gamma_{i+1} & \dots & C_{i+P}\Gamma_{i+P-1} \end{pmatrix} \tag{4.47}$$

$$S_x^i = \begin{pmatrix} C_{i+1}\Psi_i \\ C_{i+2}\Psi_{i+1}\Psi_i \\ \vdots \\ C_{i+P}\Psi_{i+P-1}\dots\Psi_{i+1}\Psi_i \end{pmatrix} \tag{4.48}$$

Ici, les matrices C_k, Ψ_k et Γ_k sont celles de (4.41). Les calculs sont donnés en annexe E.

4.3.3.3 Expression des contraintes

On rappelle ici les contraintes du problème MPC que l'on reformule à l'aide des variables $\Delta\delta u_{i+k}$. Une synthèse est donnée en (4.54a) et (4.55).

– Contraintes sur u

$$\begin{aligned} u_{min} \leq & \quad u_{i+k} & \leq u_{max} \\ u_{min} \leq & \quad \delta u_{i-1} + u_{r,i+k} + \sum_{n=0}^{k}\Delta\delta u_{i+P} & \leq u_{max} \end{aligned}$$

On a ainsi les deux ensembles de contraintes suivants :

$$\begin{aligned} -\sum_{n=0}^{k}\Delta\delta u_{i+P} &\leq -u_{min} + \delta u_{i-1} + u_{r,i+k} \\ \sum_{n=0}^{k}\Delta\delta u_{i+P} &\leq u_{max} - \delta u_{i-1} - u_{r,i+k} \end{aligned}$$

Ou sous forme matricielle

$$\begin{aligned} -L\Delta\delta U^i &\leq (\delta u_{i-1} - u_{min})\mathbf{1} + U_r^i \\ L\Delta\delta U^i &\leq (u_{max} - \delta u_{i-1})\mathbf{1} - U_r^i \end{aligned}$$

– Contraintes (symétriques) sur δu

$$\begin{aligned} -\delta u_{max} \leq & \quad \delta u_{i+k} & \leq \delta u_{max} \\ -\delta u_{max} \leq & \quad \delta u_{i-1} + \sum_{n=0}^{k}\Delta\delta u_{i+P} & \leq \delta u_{max} \end{aligned}$$

Sous forme matricielle

$$\begin{aligned} -L\Delta\delta U^i &\leq (\delta u_{i-1} - \delta u_{max})\mathbf{1} \\ L\Delta\delta U^i &\leq (\delta u_{max} - \delta u_{i-1})\mathbf{1} \end{aligned}$$

– Contraintes (symétriques sur Δu)

$$\begin{array}{rcl} -\Delta u_{max} \leq & \Delta u_{i+k} & \leq \Delta u_{max} \\ -\Delta u_{max} \leq & \Delta\delta u_{i+k} + \Delta\delta u_{r,i+k} & \leq \Delta u_{max} \end{array}$$

Sous forme matricielle

$$\begin{array}{rl} -\Delta\delta U^i & \leq \left(\Delta U_{max} - \Delta U_r^i\right) \\ \Delta\delta U^i & \leq \left(\Delta U_{max} - \Delta U_r^i\right) \end{array}$$

– Contraintes sur y : d'après (4.44)

$$\begin{array}{rcl} Y_{min} \leq & Y_i & \leq Y_{max} \\ Y_{min} \leq & Y_{nom}^i + S_x^i x_{a,i} + S_u^i \Delta\delta U^i & \leq Y_{max} \end{array}$$

Sous forme matricielle

$$\begin{array}{r} S_u^i \Delta\delta U^i \leq Y_{max} - Y_{nom}^i - S_x^i x_{a,i} \\ -S_u^i \Delta\delta U^i \leq Y_{nom}^i - Y_{min} + S_x^i x_{a,i} \end{array}$$

– Contraintes (symétriques) sur δy

$$\delta Y_{max} \leq \delta Y_i \leq \delta Y_{max}$$

Donne

$$\begin{array}{r} S_u^i \Delta\delta U^i \leq \delta Y_{max} - S_x^i x_{a,i} \\ -S_u^i \Delta\delta U^i \leq \delta Y_{max} + S_x^i x_{a,i} \end{array}$$

4.3.3.4 Expression de la fonction objectif

On reformule maintenant la fonction objectif (4.35) en fonction de $\Delta\delta U^i$

$$J_i = \|U^i - U_r^i\|_{Q_u}^2 + \|Y^i - Y_r^i\|_{Q_y}^2 + \|\Delta U^i\|_R^2$$

D'après la Table 4.1 et l'équation (4.44),

$$\delta U^i = \underbrace{\begin{pmatrix} \delta u_{i-1} \\ \vdots \\ \delta u_{i-1} \end{pmatrix}}_{\delta U_l^i} + L\Delta\delta U^i \tag{4.49}$$

$$\Delta U^i = \Delta\delta U^i + \Delta U_r^i \tag{4.50}$$

$$Y^i - Y_r^i = \underbrace{S_x^i x_{a,i} + Y_{nom}^i - Y_r^i}_{\delta Y_l} + S_u^i \Delta\delta U^i \tag{4.51}$$

La fonction objectif s'exprime alors en $\Delta\delta U^i$ comme :

$$J^i = \|\delta U_l^i + L\Delta\delta U^i\|_{Q_u}^2 + \|\delta Y_l^i + S_u^i \Delta\delta U^i\|_{Q_y}^2 + \|\Delta\delta U^i + \Delta U_r^i\|_R^2$$

Cette forme se développe pour donner la forme quadratique suivante :

$$J^i(\Delta\delta U^i) = \Delta\delta U^{iT} S \Delta\delta U^i + h\Delta\delta U^i + \text{cst}$$

Les matrices S et h sont données dans l'équation suivante. Comme elles impliquent des quantités variant dans le temps, ces matrices doivent être recalculées à chaque pas de temps :

$$J^i = \Delta\delta U^{i\,T} \underbrace{\left(L^T Q_u L + {S_u^i}^T Q_y S_u^i + R^T \Delta U_r^i\right)}_{S} \Delta\delta U^i + 2\underbrace{\left(L^T Q_u \delta U_l^i + {S_u^i}^T Q_y \delta Y_l^i + R^T \Delta U_r^i\right)}_{h} \Delta\delta U^i + \underbrace{\delta {U_l^i}^T Q_u \delta U_l^i + \delta {Y_l^i}^T Q \delta Y_l^i + \Delta {U_r^i}^T R \Delta U_r^i}_{\text{constante}} \quad (4.52)$$

4.3.3.5 Problème d'optimisation MPC

Pour synthétiser, le problème d'optimisation quadratique initial ayant pour variables d'optimisation (U^i, Y^i) ($P(n_u + n_y)$ variables) a été reformulé en un problème d'optimisation de $\Delta\delta U^i$ (Pn_u variables). Celui-ci s'écrit :

$$\min_{\Delta\delta U^i} \|\delta U_l^i + L\Delta\delta U^i\|^2_{Q_u} + \|\delta Y_l^i + S_u^i \Delta\delta U^i\|^2_{Q_y} + \|\Delta\delta U^i + \Delta U_r^i\|^2_R \quad (4.53)$$

s.c.

$$(u_{min} - \delta u_{i-1})\mathbf{1} \leq L\Delta\delta U^i + U_r^i \leq (u_{max} - \delta u_{i-1})\mathbf{1} \quad (4.54a)$$

$$(\delta u_{i-1} - \delta u_{max})\mathbf{1} \leq L\Delta\delta U^i \leq (\delta u_{max} - \delta u_{i-1})\mathbf{1} \quad (4.54b)$$

$$Y_{min} - Y_n^i \leq S_u^i \Delta\delta U^i \leq Y_{max} - Y_n^i \quad (4.54c)$$

$$-\delta Y_{max} \leq S_u^i \Delta\delta U^i \leq \delta Y_{max} \quad (4.54d)$$

Avec $U_l^i = \begin{pmatrix} \delta u_{i-1} \\ \vdots \\ \delta u_{i-1} \end{pmatrix} = \delta u_{i-1}\mathbf{1}$, $\delta Y_l^i = Y_{nom}^i - Y_r^i$, et $\mathbf{1}$ un vecteur de taille appropriée ne contenant que des 1. Pour simplifier la notation, on regroupera par la suite les contraintes (4.54a) sous l'inégalité matricielle

$$A\Delta\delta U^i \leq b \quad (4.55)$$

4.3.3.6 Initialisation des modèles

Initialisation du modèle non linéaire (NLc) :

Le modèle Modelica non linéaire est censé représenter l'état réel du système. Ainsi, à chaque instant t_i du MPC, celui-ci est réinitialisé avec le dernier état estimé noté $\hat{x}_i$. On a alors

$$x^i(t_i) = \hat{x}_i = x_n^i(t_i)$$

La trajectoire nominale est ensuite calculée en partant de ce point, en appliquant au modèle Modelica la commande U_r^i donnée par l'optimisation long terme. Cela permet notamment d'obtenir la valeur nominale du point suivant, notée $x_n^i(t_{i+1})$

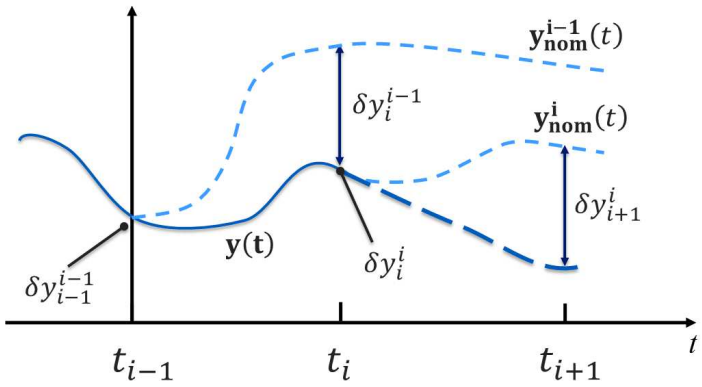


FIGURE 4.6 – Illustration des trajectoires réelles et nominales de y entre deux pas du MPC

Initialisation du premier modèle interne augmenté (4.40)

L'initialisation du premier modèle augmenté n'est pas évidente : celui-ci fait intervenir des états qui dépendent d'une trajectoire nominale. Or les trajectoires nominales entre deux problèmes MPC consécutifs ne coïncident pas, puisque celles-ci sont recalculées à chaque pas de temps. Pour rappel la trajectoire nominale dépend de la valeur initiale de l'état et des entrées nominales appliquées au système. La définition de la valeur initiale pour le problème MPC_i : $\Delta\delta x_i = \delta x_i - \delta x_{i-1}$ est donc problématique en soi. En effet, pour le problème suivant, on peut soit considérer

- $\Delta\delta x(t_{i+1}) = \delta x^i(t_{i+1}) - \delta x^i(t_i)$
- $\Delta\delta x(t_{i+1}) = \delta x^{i+1}(t_{i+1}) - \delta x^i(t_i)$. Or par construction des trajectoires nominales, $\delta x^{i+1}(t_{i+1}) = \delta x^i(t_i) = 0$. Cela revient à initialiser le modèle à chaque pas de temps avec $\Delta\delta x_i = 0$.

Ces deux situations sont illustrées dans la Figure 4.6 et implémentées dans l'algorithme.

Initialisation du second modèle interne augmenté (4.41) :

Le second modèle augmenté a pour état initial $x_{a,i} = \begin{pmatrix} \delta x_i \\ \delta u_{i-1} \end{pmatrix}$. On rappelle $\delta u_{i-1} = u_{i-1} - u_{r,i-1}$. Comme la trajectoire de référence des commandes u_r reste la même pour deux problèmes MPC consécutifs, il n'y a pas la même ambiguïté que celle rencontrée précédemment pour les états. Le modèle interne augmenté est ainsi réinitialisé pour chaque problème MPC_i avec

$$x_{a,i} = \begin{pmatrix} 0 \\ \delta u_{i-1} \end{pmatrix}$$

On retiendra par la suite ce second modèle augmenté car sa ré-initialisation n'est pas ambiguë contrairement au premier modèle augmenté.

4.3.3.7 Normalisation

Les modèles considérés font intervenir simultanément des grandeurs physiques qui peuvent avoir des ordres de grandeur très différents. Il est donc important de normaliser celles-ci, à la fois pour des raisons de précision numérique, mais aussi de convergence lors de la résolution du

problème quadratique. On propose la normalisation suivante, basée sur les valeurs nominales des variables x, u et y (on a regroupé w dans u pour alléger la notation). Les variables normalisées $\tilde{x}$, $\tilde{u}$, $\tilde{y}$ sont respectivement l'état, la commande et la sortie normalisés.

$$x = N_x\tilde{x} \quad u = N_u\tilde{u} \quad y = N_y\tilde{y} \tag{4.56}$$

Les matrices N_v sont les matrices diagonales de normalisation avec pour terme à la ligne i : $N_{v,i} = v_{max,i} - v_{min,i}$. L'approximation linéaire tangent normalisée est alors

$$\left\{ \begin{array}{rcl} \tilde{x}_{k+1} & = & {N_x}^{-1}A_kN_x\delta\tilde{x}_k + {N_x}^{-1}B_kN_u\delta\tilde{u}_k \\ \delta\tilde{y}_k & = & {N_y}^{-1}c_kN_x\delta\tilde{x}_k + {N_y}^{-1}d_kN_u\delta\tilde{u}_k \end{array} \right. \tag{4.57}$$

Concernant le critère, il est plus évident pour l'utilisateur de définir les pondérations sur les grandeurs normalisées, sans se soucier des unités. Le critère est ainsi modifié de la façon suivante dans l'implémentation de l'algorithme :

$$\begin{aligned} J_i &= \|\tilde{U}^i - \tilde{U}_r^i\|_{Q_u}^2 + \|\tilde{Y}^i - \tilde{Y}_r^i\|_{Q_y}^2 + \|\Delta\tilde{U}^i\|_R^2 \\ &= \|{N_u}^{-1}(U^i - U_r^i)\|_{Q_u}^2 + \|{N_y}^{-1}(\delta Y^i - Y_r^i)\|_{Q_y}^2 + \|{N_u}^{-1}\Delta U^i\|_R^2 \\ &= \|{N_u}^{-1}(U^i - U_r^i)\|_{Q_u}^2 + \|{N_y}^{-1}(\delta Y^i - Y_r^i)\|_{Q_y}^2 + \|{N_u}^{-1}\Delta U^i\|_R^2 \\ &= \|U^i - U_r^i\|_{{N_u}^{-1^T}Q_uN_u^{-1}}^2 + \|\delta Y^i - Y_r^i\|_{{N_y}^{-1^T}Q_yN_y^{-1}}^2 + \|\Delta U^i\|_{{N_u}^{-1^T}RN_u^{-1}}^2 \end{aligned}$$

On constate ainsi que la normalisation n'a pas affecté la forme du critère (4.35), seules les matrices de pondération sont modifiées pour prendre en compte la normalisation. Le vecteur b donnant les bornes des contraintes doit également être modifié.

4.3.3.8 Elimination de la matrice D

Il est possible que les modèles Modelica présentent un couplage direct entre certaines entrées et certaines sorties. Notons u_c les commandes couplées à au moins une sortie, et $u \setminus u_c$ les autres. Dans ce cas, l'équation pour les sorties s'exprime

$$y = h(x, u_c)$$

et non

$$y = h(x)$$

Le modèle linéarisé tangent (TanLd) possède ainsi une matrice de couplage entrée / sortie D_k non nulle. Les termes non nuls sont présents dans les colonnes de D correspondant à u_c. On a :

$$\left\{ \begin{array}{rl} \delta x_{k+1} & = A_k\delta x_k + B_k \begin{pmatrix} \delta u_k & \delta w_k \end{pmatrix}^T \\ \delta y_k & = c_k\delta x_k + D_k \begin{pmatrix} \delta u_k^T & \delta w_k^T \end{pmatrix}^T \end{array} \right. \tag{4.58}$$

Comme évoqué dans le paragraphe 4.3.3.1, on considère par la suite $\delta w = 0$ dans le modèle interne.

Par filtrage des entrées

Une première façon d'éliminer ces termes est de modifier le modèle de simulation en introduisant des filtres rapides du premier ordre sur les commandes u_c. On note i_c les indices de ces colonnes, et $i \setminus i_c$ les autres indices. Les lignes d'indices I d'une matrice M sont notées $M_{(I)}$. Un filtre du premier ordre a pour équation :

$$\dot{\tilde{u}_c}(t) = \frac{1}{T}\left(u_c(t) - \tilde{u}_c(t)\right)$$

En choisissant T petit devant les constantes de temps caractéristiques du système, la commande $\tilde{u}_c$ converge rapidement vers u_c. Le système prend alors la forme

$$\begin{cases} \begin{pmatrix} \dot{\delta x} \\ \dot{\delta \tilde{u}_c} \end{pmatrix} &= \begin{bmatrix} A(t) & B_{(i_c)}(t) \\ 0 & -\frac{1}{T} \end{bmatrix} \begin{pmatrix} \delta x(t) \\ \delta \tilde{u}_c(t) \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{T} \end{pmatrix} \delta u_c(t) + \begin{pmatrix} B_{(\overline{i_c})}(t) \\ 0 \end{pmatrix} \delta u_k \\ \delta y(t) &= \begin{bmatrix} c(t) & d_{(i_c)}(t) \end{bmatrix} \begin{pmatrix} \delta x(t) \\ \delta \tilde{u}_c(t) \end{pmatrix} \end{cases} \tag{4.59}$$

La discrétisation temporelle appliquée à ce système ne fera plus apparaître de matrice D, et l'on pourra alors utiliser la formulation MPC décrite précédemment sans plus de modifications. Cependant, cette solution n'est pas acceptable car elle demande à l'utilisateur d'une part de modifier le modèle de simulation, d'autre part de connaître les entrées responsables d'un couplage direct entrée/sortie. On propose donc dans le paragraphe suivant les modifications nécessaires à la formulation MPC pour prendre en compte ce couplage sur le modèle initial.

Par changement de variable pour les sorties

La prise en compte de la matrice D dans l'algorithme MPC se fait en recourant au changement de variable suivant : $\delta\tilde{y}_k = \delta y_k - d_k \delta u_k$.

Le modèle interne devient

$$\begin{cases} \delta x_{k+1} &= A_k \delta x_k + B_k \begin{pmatrix} \delta u_k^T & \delta w_k^T \end{pmatrix}^T \\ \delta\tilde{y}_k &= c_k \delta x_k \end{cases} \tag{4.60}$$

Cette reformulation a des impacts sur l'expression de la fonction objectif et des contraintes sur les sorties. Le vecteur des sorties s'exprime alors comme :

$$\begin{aligned} Y^i &= Y^i_{nom} + \delta Y^i \\ &= Y^i_{nom} + \delta\tilde{Y}^i + S^i_d \delta U^i \\ &= Y^i_{nom} + \delta\tilde{Y}^i + S^i_d \left(\delta u_{i-1}\mathbf{1} + L\Delta\delta U^i\right) \end{aligned}$$

Avec S^i_d une matrice diagonale par bloc de dimension $(n_y * P) \times (n_u * P)$:

$$S^i_d = \begin{pmatrix} d_{i+1} & 0 & \dots & 0 \\ 0 & d_{i+2} & \dots & 0 \\ \vdots & \dots & \ddots & \vdots \\ 0 & 0 & 0 & d_{i+P-1} \end{pmatrix}$$

On a :

$$Y^i = Y^i_{nom} + \left(S^i_d L + S^i_u\right) \Delta\delta U^i + S^i_d \left(\delta u_{i-1}\mathbf{1}\right) + S^i_x x_{a,i}$$

En posant :

$$\delta\tilde{Y}_l = Y^i_{nom} - Y^i_r + S^i_x x_{a,i} + S^i_d \left(\delta u_{i-1}\mathbf{1}\right) \tag{4.61}$$

la fonction objectif (4.35) peut alors être reformulée de la façon suivante

$$J_i = \|U^i - U^i_r\|^2_{Q_u} + \|Y^i - Y^i_r\|^2_{Q_y} + \|\Delta U^i\|^2_R \tag{4.62}$$
$$= \|\delta U^i_l + L\Delta\delta U^i\|^2_{Q_u} + \|\delta\tilde{Y}^i_l + S^i_u \Delta\delta U^i\|^2_{Q_y} + \|\Delta\delta U^i + \Delta U^i_r\|^2_R \tag{4.63}$$

Les contraintes en y se réécrivent :

$$(S^i_u + S^i_d L)\Delta\delta U^i \leq Y_{max} - Y^i_{nom} - (\delta u_{i-1}\mathbf{1}) - S^i_x x_{a,i} \tag{4.64}$$
$$-(S^i_u + S^i_d L)\Delta\delta U^i \leq Y^i_{nom} - Y_{min} + S^i_d \left(\delta u_{i-1}\mathbf{1}\right) + S^i_x x_{a,i} \tag{4.65}$$
$$(S^i_u + S^i_d L)\Delta\delta U^i \leq \delta Y_{max} - S^i_x x_{a,i} - S^i_d(\delta u_{i-1}\mathbf{1}) \tag{4.66}$$
$$-(S^i_u + S^i_d L)\Delta\delta U^i \leq \delta Y_{max} + S^i_x x_{a,i} + S^i_d(\delta u_{i-1}\mathbf{1}) \tag{4.67}$$

4.3.3.9 Accélération du MPC

Dans la procédure MPC décrite en 4.1.4, on réalise à chaque pas i du MPC P linéarisations du modèle correspondant aux P prochains instants d'échantillonnage. Or en raison de l'horizon glissant, on constate que les matrices Ψ et Γ sont calculées plusieurs fois pour les mêmes instants. La seule raison qui justifie cela est que la trajectoire réelle évolue différemment de la trajectoire nominale, donc que le point de linéarisation pour un même instant change entre deux problèmes MPC consécutifs. Cependant, l'horizon MPC reste relativement court, et l'hypothèse a été faite que l'on reste dans un voisinage du point de référence. Pour diminuer le temps de calcul, on peut ainsi supposer que les matrices de linéarisation calculées pour un instant t_k ne varient pas beaucoup entre deux problèmes MPC consécutifs. Il est alors possible de faire un "démarrage à chaud", ou *warm start* pour le problème MPC suivant. Avec cette nouvelle hypothèse, on constate que les matrices S_u et S_x pour le problème MPC_{i+1} sont très semblables à celles calculées pour le problème MPC_i. On donne ici le détail des calculs pour le modèle augmenté (4.40)

Évolution de S_u entre deux itérations MPC

Entre deux problèmes MPC consécutifs, si l'on considère que les matrices de linéarisation sont identiques, on a :

$$S_u^i = \begin{pmatrix} C\Gamma_i & 0 & \dots & 0 \\ C\Psi_{i+1}\Gamma_i & C\Gamma_{i+1} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C\prod_{n=1}^{P-2}\Psi_{i+P-1-n}\Gamma_i & C\prod_{n=1}^{P-3}\Psi_{i+P-1-n}\Gamma_{i+1} & \dots & C\Gamma_{i+P-2} \\ C\prod_{n=1}^{P-1}\Psi_{i+P-n}\Gamma_i & C\prod_{n=1}^{P-2}\Psi_{i+P-n}\Gamma_{i+1} & \dots & C\Gamma_{i+P-1} \end{pmatrix}$$

$$S_u^{i+1} = \begin{pmatrix} C\Gamma_{i+1} & 0 & \dots & 0 \\ C\Psi_{i+2}\Gamma_{i+1} & C\Gamma_{i+2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C\prod_{n=1}^{P-2}\Psi_{i+P-n}\Gamma_{i+1} & C\prod_{n=1}^{P-1}\Psi_{i+P-n}\Gamma_{i+1} & \dots & C\Gamma_{i+P-1} \\ C\prod_{n=1}^{P-1}\Psi_{i+P+1-n}\Gamma_{i+1} & C\prod_{n=1}^{P-2}\Psi_{i+P+1-n}\Gamma_{i+2} & \dots & C\Gamma_{i+P} \end{pmatrix}$$

On constate que seule la dernière ligne fait apparaître des éléments qui n'avaient pas encore été calculés : Ψ_{i+P} et Γ_{i+P}. On définit la matrice carrée $\tilde{C}$ de dimension $n_y \times P$

$$\tilde{C} = \begin{pmatrix} C & 0 & \dots 0 \\ 0 & C & \dots 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots C \end{pmatrix}$$

ainsi que la matrice $\tilde{S}_u^i$ réellement stockée dans le programme :

$$\tilde{S}_u^i = \begin{pmatrix} \Gamma_i & 0 & \dots & 0 \\ \Psi_{i+1}\Gamma_i & \Gamma_{i+1} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \prod_{n=1}^{P-2}\Psi_{i+P-1-n}\Gamma_i & \prod_{n=1}^{P-3}\Psi_{i+P-1-n}\Gamma_{i+1} & \dots & \Gamma_{i+P-2} \\ \prod_{n=1}^{P-1}\Psi_{i+P-n}\Gamma_i & \prod_{n=1}^{P-2}\Psi_{i+P-n}\Gamma_{i+1} & \dots & \Gamma_{i+P-1} \end{pmatrix} \tag{4.68}$$

La matrice S_u^{i+1} peut ainsi se construire d'après S_u^i de la façon suivante :

- Premières lignes : Recopie décalée de la partie inférieure droite de $\tilde{S}_u^i$: $\tilde{S}_u^{i+1}[i,j] := \tilde{S}_u^i[i+1, j+1], \quad (i,j) \in \overline{1..P-1}^2$
- Dernière ligne : égale à la multiplication par Ψ_{i+P} de l'avant dernière ligne par la gauche
- Terme inférieur droit : Γ_{i+P}
- Retour à S_u : $S_u^{i+1} = \tilde{C}\tilde{S}_u^{i+1}$

Ainsi, à chaque problème MPC, une seule linéarisation à l'instant t_{i+P+1} est maintenant nécessaire au lieu de P.

Évolution de S_x entre deux itérations MPC

En ce qui concerne l'évolution de la matrice S_x, on a :

$$S_x^i = \begin{pmatrix} C\Psi_i \\ C\Psi_{i+1}\Psi_i \\ \vdots \\ C\Psi_{i+P-1}\dots\Psi_{i+1}\Psi_i \end{pmatrix}$$

$$S_x^{i+1} = \begin{pmatrix} C\Psi_{i+1} \\ C\Psi_{i+2}\Psi_{i+1} \\ \vdots \\ C\Psi_{i+P-1}\dots\Psi_{i+1} \\ C\Psi_{i+P}\Psi_{i+P-1}\dots\Psi_{i+1} \end{pmatrix}$$

De nombreux termes se retrouvent également, sauf pour les termes à gauche et à droite. Pour construire la matrice S_x^i, on introduit une matrice auxiliaire que l'on nomme $\tilde{S}_x$ pour stocker les valeurs de Ψ_k. Entre deux itérations successives, si l'on suppose que les matrices de linéarisation sont conservées, on a ainsi :

$$\tilde{S}_x^i = \begin{pmatrix} \Psi_i \\ \Psi_{i+1} \\ \vdots \\ \Psi_{i+P-1} \end{pmatrix}, \quad \tilde{S}_x^{i+1} = \begin{pmatrix} \Psi_{i+1} \\ \vdots \\ \Psi_{i+P-1} \\ \Psi_{i+P} \end{pmatrix} \tag{4.69}$$

On effectue seulement ensuite les multiplications nécessaires pour aboutir à S_x^i : la matrice S_x se construit en multipliant tout d'abord de façon récursive les blocs-lignes de $\tilde{S}_x$ par la gauche, puis en multipliant la matrice obtenue par la matrice $\tilde{C}$ (par la gauche également) :

$$S_x^i = \begin{pmatrix} C & 0 & \dots 0 & \\ 0 & C & \dots 0 & \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots C & \end{pmatrix} \begin{pmatrix} \Psi_i \\ \Psi_{i+1}\Psi_i \\ \vdots \\ \Psi_{i+P}\dots\Psi_i \end{pmatrix} = \tilde{C} \begin{pmatrix} \Psi_i \\ \Psi_{i+1}\Psi_i \\ \vdots \\ \Psi_{i+P}\dots\Psi_i \end{pmatrix} \tag{4.70}$$

Dans le modèle augmenté (4.41), la seule différence réside au niveau de la matrice $\tilde{C}$ qui varie dans le temps et est définie comme $\tilde{C}^i = diag(C_{i+k}, \quad k \in \overline{1..P})$.

4.3.4 Estimateur d'état

On présente dans cette partie l'estimateur utilisé pour estimer l'état réel de la centrale. Celui-ci doit être conçu pour prendre en compte les diverses erreurs pouvant exister entre les prédictions du modèle non linéaire et la réalité. Ces erreurs sont retranscrites dans une matrice P_k variant dans le temps, appelée matrice de covariance. On considère les sources d'erreur suivantes :

- la présence de perturbations non mesurées $w = w_r + \delta w$, où δw correspond à une erreur de prévision, pouvant être modélisée par un bruit blanc de covariance W
- la présence de bruit de mesure υ que l'on suppose être un bruit blanc de covariance V, venant s'ajouter à la sortie réelle

– des erreurs de modélisation w_m venant s'ajouter à l'état dans l'équation (TanLd). Ces erreurs sont considérées elles aussi comme un bruit de covariance W_m.

Ainsi, en appelant ϵ_k l'erreur de prédiction à t_k, la réponse réelle de la centrale et les sorties mesurées peuvent s'écrire sous la forme suivante :

$$\begin{cases} x_{p,k} &= x(t_k) + \epsilon_k \\ x_{p,k+1} &= f_p^{(d)}(x_{p,k}, u_k, w_k) + w_m \\ y_{mes,k} &= h(x_{p,k}) + \upsilon_k \end{cases} \tag{4.71}$$

On utilise un filtre de Kalman étendu (EKF), variant dans le temps, pour l'estimation d'état, en se basant sur le modèle non linéaire pour la phase de prédiction, et sur le modèle linéaire tangent (TanLd) pour la mise à jour de la matrice de covariance et la phase de correction. Une description du filtre de Kalman étendu peut être trouvée dans [80]. Le filtre de Kalman est tout d'abord initialisé avec une matrice de covariance $P_0 = B_{w,0} W B_{w,0}^T + W_m$.

Phase de prédiction : La prédiction de la valeur de l'état et de la sortie au pas i se fait au pas $i-1$ par simulation du modèle non linéaire sur un pas de temps. On suppose que les commandes sont maintenues constantes sur la période d'échantillonnage (*zero order hold approximation*), et que les perturbations sont à leur valeur nominale (mesurée si l'information existe).

$$\begin{cases} \tilde{x_i} &= f^{(d)}\left(x_{i-1}, u_{i-1}, w_{r,i-1}; T_s\right) \\ \tilde{y_i} &= h(\tilde{x_i}) \end{cases} \tag{4.72}$$

Phase de correction : A l'instant t_i, la mesure réelle des sorties devient disponible. On corrige donc l'état $\tilde{x_i}$ obtenu en phase de prédiction en tenant compte de cette nouvelle information. L'état estimé $\hat{x_i}$ par le filtre au pas de temps i se calcule alors comme :

$$K_i = P_i c_i^T \left(c_i^T P_i^T c_i + V \right)^{-1} \tag{4.73}$$

$$\hat{x_i} = \tilde{x_i} + K_i(y_{mes,i} - \tilde{y_i}) \tag{4.74}$$

Mise à jour du filtre : Enfin, la matrice de covariance P est mise à jour à la fin de chaque pas :

$$P_{i+1} = A_i(I - K_i c_i)A_i^T + B_{w,i} W B_{w,i}^T + W_m \tag{4.75}$$

4.3.5 Algorithme MPC détaillé

L'algorithme MPC est détaillé sous forme de diagramme dans la Figure 4.7, dans une version simple où toutes les matrices de linéarisation sont recalculées à chaque pas de temps. La fonction *linéariser* prend en entrée l'état initial, le temps courant et l'instant de linéarisation, ainsi que les vecteurs U_r, V_r, W_r. La fonction *c2d* permet la conversion des matrices de linéarisation en temps discret, avec un pas d'échantillonnage T_s.

4.4 Application à la cogénération

Présentation du cas d'étude :

L'algorithme TL-MPC est testé sur le modèle physique d'une centrale de cogénération, illustré sur la Figure 4.8. Ce modèle est un modèle plus complexe que le modèle d'optimisation

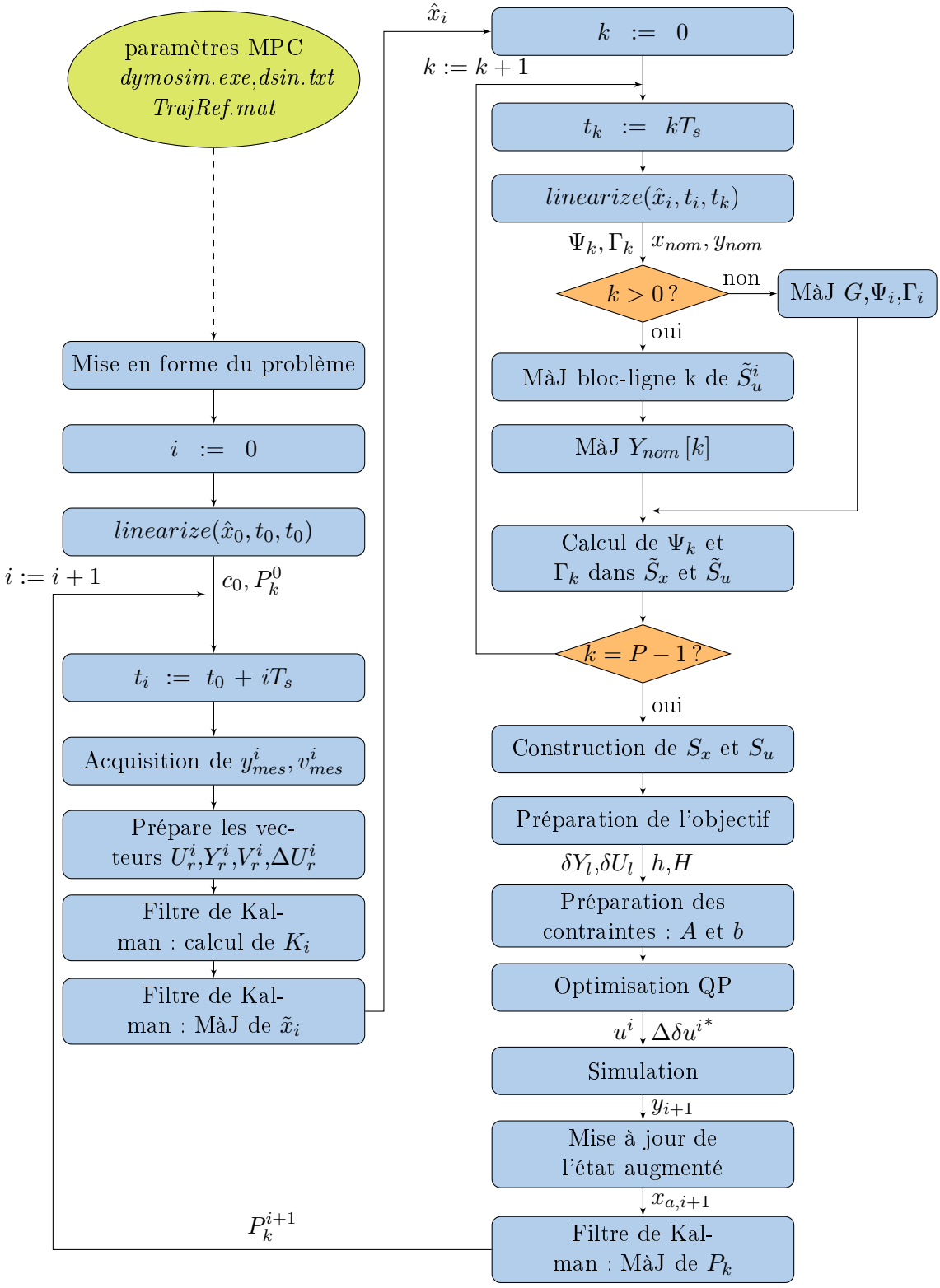


FIGURE 4.7 – Algorithme MPC simple

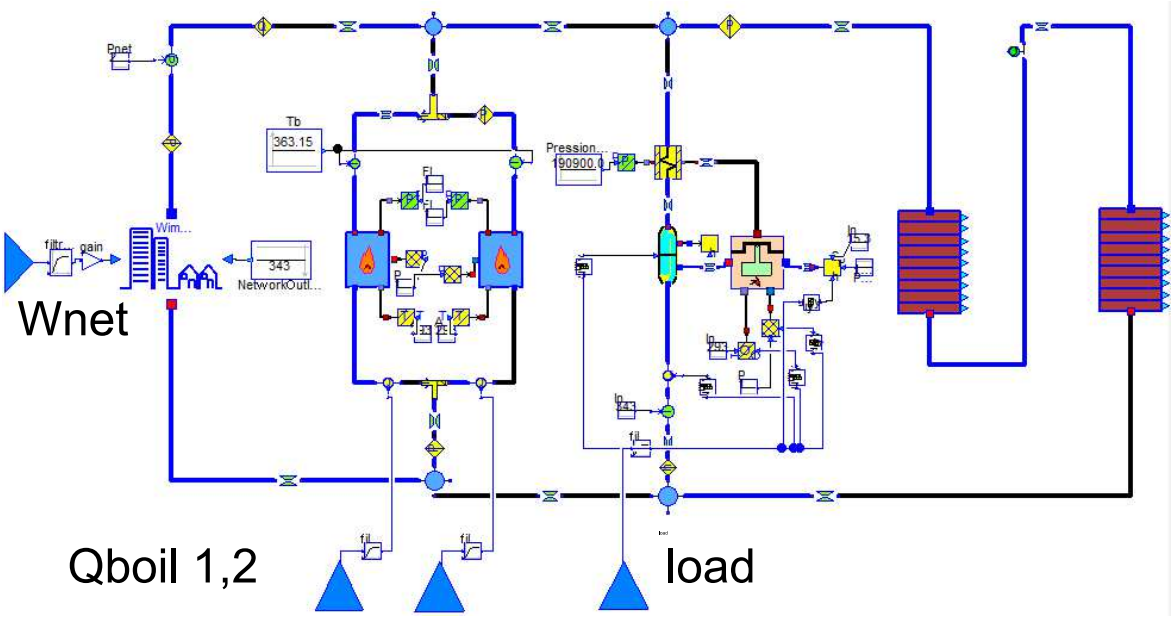


FIGURE 4.8 – Modèle Modelica de la cogénération de Barkantine

présenté en partie 3 : il contient deux stockages d'eau chaude en série discrétisés en 9 strates, deux chaudières d'appoint, la combustion est modélisée dans le moteur et les propriétés de l'eau sont calculées suivant le standard IF97 [81]. Ce modèle possède 31 états et 726 variables algébriques. La régulation automatique des deux chaudières ainsi que le pilotage de la charge du moteur ont été retirés et sont désormais placés sous le contrôle du MPC. Certaines régulations internes sont présentes dans le modèle : ainsi, les débits d'eau, de combustible et d'air dans la branche du moteur et celles des chaudières sont pilotés en fonction de la charge de ceux-ci. Le vecteur de commande est ainsi $u = \begin{pmatrix} \text{load Engine} \\ \text{load B1} \\ \text{load B2} \end{pmatrix}$ (charge électrique du moteur et débit d'eau dans les deux chaudières). La demande W_{demand} du réseau de chaleur et le prix de l'électricité sont modélisés comme des entrées mesurées mais non commandées du modèle, d'où $w^m = \begin{pmatrix} \text{W demand} \\ \text{C elec} \end{pmatrix}$.

Le logiciel PILOT est utilisé pour définir un programme de production optimal (d'un point de vue économique) sur un horizon d'une journée, afin de donner les trajectoires de référence y_r et u_r pour le MPC. Les trajectoires optimales ont été calculées au pas de temps de la demi-heure en utilisant les prévisions de demande. Les contraintes suivantes sont prises en compte :

- Puissances minimales et maximales de fonctionnement pour le moteur et les deux chaudières
- Stock minimal et maximal
- Contrainte finale sur le niveau de stock
- Deux démarrages par jour au maximum pour le moteur

Le système sous commande TL-MPC est simulé sur les Figures 4.9 et 4.10 pour un jour de printemps, en utilisant la courbe de demande en chaleur réellement observée ce jour. Les

trajectoires de référence sont en croix bleues et les résultats de simulation en trait plein noir. La trajectoire de référence $w^m_{1,r(t)}$ de la demande est construite d'après les prévisions obtenues par un modèle statistique *Generalized Additive Model* (GAM) décrit en [82]. Ce modèle de prévision de courbe de charge prend en compte de nombreux facteurs incluant les prévisions météorologiques (température, vent) et la saisonnalité. La courbe de charge est représentée sur le dernier graphique de la Figurre 4.10 : les prévisions sont en croix bleues, la demande réelle est en trait plein noir. Dans une optique de comparaison, on simule également le système sans correction MPC, c'est à dire en appliquant en boucle ouverte les trajectoires de commande u_r et les perturbations réelles w^m. Les résultats de cette simulation sont en trait tiret-pointillé magenta sur la Figure 4.9.
Le MPC est paramétré avec un horizon $P = 10$ et une période d'échantillonnage de 10 minutes (soit un horizon de 1 h 40 min). En raison de la taille importante du modèle, la version accélérée pour le calcul des matrices décrite en 4.3.3.9 a été utilisée (il y a donc une linéarisation et non P linéarisations à chaque pas d'échantillonnage). Deux sorties ont été considérées pour le suivi de trajectoire et sont représentées en Figure 4.9 : l'énergie totale E_{sto} contenue dans les stocks et la température T_{net} de l'eau fournie au réseau. Les entrées sont quant à elles représentées en Figure 4.10 : la charge électrique au moteur puis les charges thermiques des chaudières 1 et 2. Dans ces deux figures, les croix bleues représentent la trajectoire de référence (u_r ou y_r) et les lignes rouges pointillé les contraintes sur la variable. Enfin, à la fin de l'horizon, les tirets verts représentent la séquence de commande calculée par le MPC pour les P prochains pas.

Définitions de l'objectif et des contraintes :

On s'intéresse tout d'abord aux commandes du système. Le moteur et les chaudières sont des composants *extinguibles* comme décrit en 4.3.2.2. Les contraintes sur les entrées load Engine, load B1 et load B2 évoluent donc dans le temps : lorsque l'état de ces composants est OFF dans la solution du MILP, les bornes u_{min} et u_{max} sont mises à 0. En revanche, les bornes suivantes sont utilisées lorsque le composant est ON : $[0.5; 1]$ pour le moteur et $[60; 100\%]$ pour la charge des chaudières. Comme il s'agit de variables commandées, on utilise ici des contraintes dures. On s'intéresse maintenant aux pondérations sur les commandes dans la fonction objectif, et donc au choix des matrices Q_u et R utilisées. Les chaudières étant très flexibles, on ne pondère pas leur déviation par rapport à u_r, en revanche, les variations brusques sur le moteur doivent être évitées, et l'écart à la référence ne doit pas être trop important car il correspond à un gain en terme de vente d'électricité. Au vu de ces remarques, les matrices de pondérations suivantes sont choisies : $Q_u = diag(1, 0, 0)$ and $R = diag(0.01, 0, 0)$. En ce qui concerne les sorties, on considère que le plus important est de suivre le profil de charge du stockage afin de garantir la faisabilité sur toute la journée. Pour la température de l'eau envoyée au réseau en revanche, l'exigence de suivi n'est pas aussi stricte : on requiert seulement que celle-ci soit comprise dans une bande de $90 + 2^oC$. Pour ces raisons, on choisit la matrice de pondération suivante pour y : $Q_y = diag(10, 0)$, et l'on ajoute au problème des contraintes sur la capacité maximale du stockage et la température au réseau. Contrairement aux entrées, des contraintes douces sont employées pour les sorties : l'excursion en dehors des bornes est autorisée mais fortement pénalisée, ce qui permet d'éviter les problèmes de faisabilité du problème MPC.

Analyse des résultats :

Un premier point que l'on note est que la température envoyée au réseau durant les premières heures passe bien en dessous de la limite autorisée (mais qui ne met pas le MPC en échec en raison des contraintes douces sur les sorties). La raison en est la suivante : au début de l'horizon, le moteur et les deux chaudières sont à l'état OFF d'après l'optimisation MILP. Comme le MPC ne remet pas en cause ce type de décisions logiques, les trois entrées sont forcées à 0 par le MPC, ce qui signifie que seul le stock est utilisé pour répondre à la demande de chaleur durant les premières heures. La température de celui-ci baisse donc inexorablement. Ce type de comportement n'est pas surprenant car l'on a utilisé un modèle MILP en énergie qui ne prend pas en compte le profil de température dans le stock. Pour contrer ce problème, et afin de ne pas passer sous un seuil de température critique, un lancement d'urgence de la chaudière 1 est présent dans le modèle. On constate que cette procédure (qui bypasse la commande MPC) s'active de 5 h à 6 h environ (trait tiret-pointillé cyan).
Après cette première phase, les chaudières puis le moteur sont allumés. On constate sur la Figure 4.9 que le système sous commande TL-MPC parvient à mieux suivre le profil du stockage et à mieux respecter les bornes sur la température envoyée au réseau par rapport à la solution sans MPC (magenta), malgré la présence de perturbations sur la demande.
Un autre point intéressant à noter est que dans la solution sans MPC, l'état du stock s'écarte beaucoup des prévisions. Cela peut avoir des conséquences fâcheuses lorsque l'on se rapproche des contraintes, qui peuvent cesser d'être respectées dans le système réel alors qu'elles l'étaient dans les prévisions MILP.

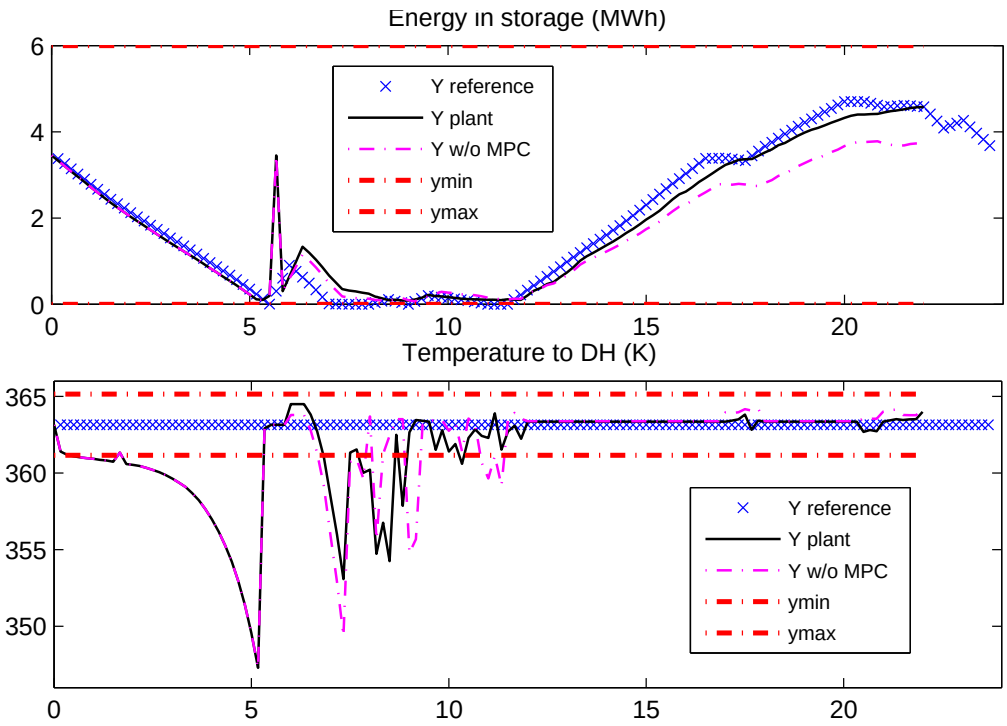


FIGURE 4.9 – Simulation du MPC sur une cogénération avec la demande réeelle : sorties

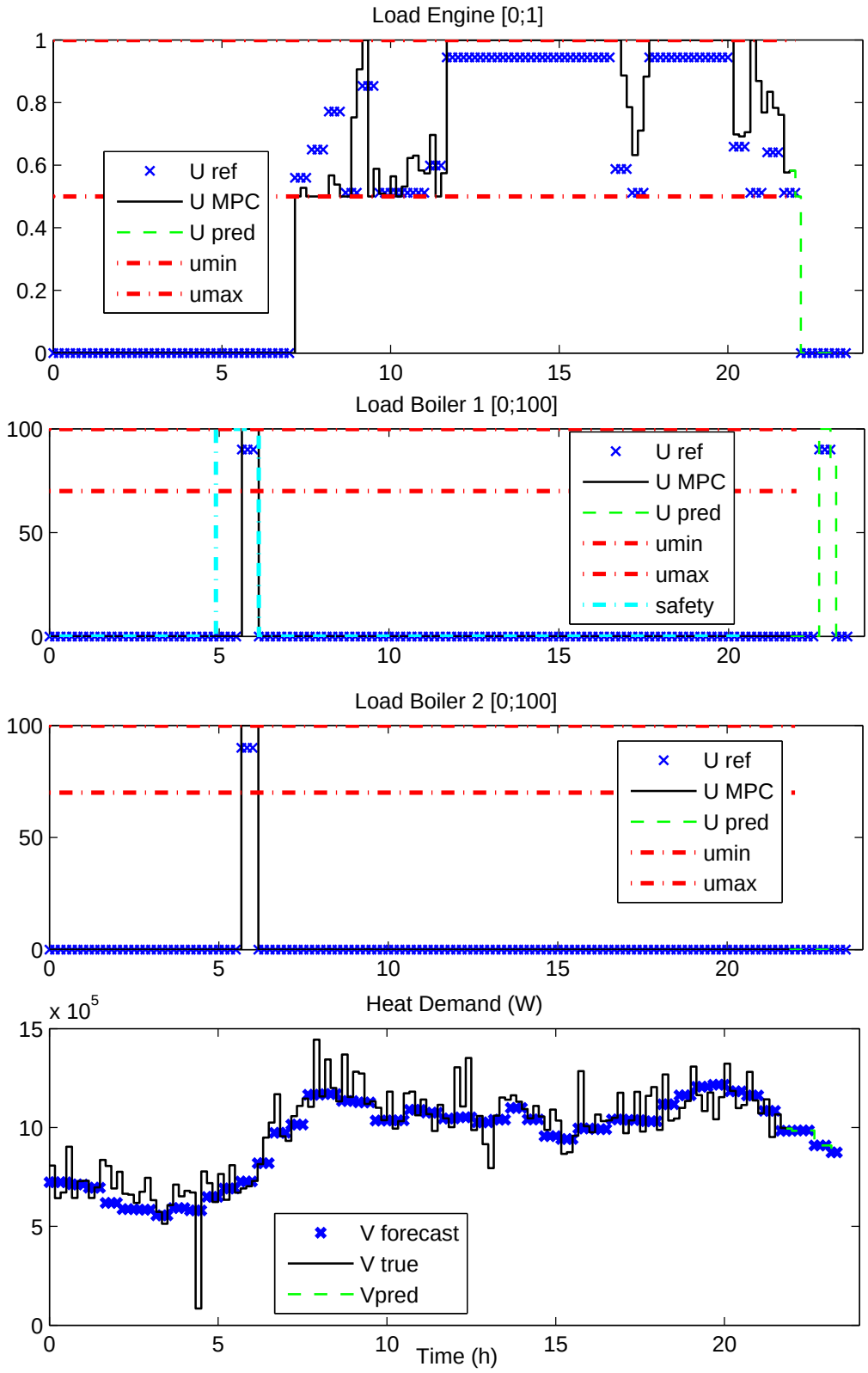


FIGURE 4.10 – Simulation du MPC sur une cogénération avec la demande réeelle : entrées (commandes et perturbations)

CHAPITRE 5

Conclusion et perspectives

5.1 Synthèse du travail effectué

Cette thèse propose un ensemble de méthodes pour le pilotage optimisé de centrales de production d'énergie en utilisant des modèles physiques des installations développés dans le langage Modelica. Modelica est un langage de modélisation de haut niveau permettant de créer des modèles complexes par assemblage de composants élémentaires avec une approche orientée objet. Dans cette thèse, on met à profit les capacités de traitement symbolique des modèles offertes par ce langage dans le cadre de deux algorithmes : le premier concerne l'optimisation du planning de production des installations, le second la commande prédictive pour le suivi de ce planning en temps réel.

5.1.1 Optimisation dynamique hybride

Dans le cas des centrales d'énergie modélisées de façon physique, l'optimisation du planning de production est un problème d'optimisation dynamique non-linéaire hybride, les aspects hybrides étant dus notamment à la présence de composants pouvant être inactivés ou opérer dans différents modes de fonctionnement. Ce type de problème est donc à la croisée des problèmes d'optimisation dynamique, qui se formulent uniquement sur des modèles continus, et des problèmes d'optimisation impliquant des variables discrètes, la Programmation Linéaire en Nombres Entiers en étant l'exemple le plus courant.
La méthode développée, qui mêle ces deux aspects concerne deux points :

- on énonce un ensemble de règles pour mettre le modèle sous une forme adaptée à l'optimisation : cette reformulation s'effectue en introduisant de nouvelles commandes $\omega^{ji}(t)$, qui indiquent si un composant j du système se trouve dans le mode i à l'instant t ($\omega^{ji}(t) = 1$) ou non ($\omega^{ji}(t) = 0$). Ces commandes sont par la suite relaxées sur l'intervalle $[0, 1]$ et utilisées pour convexifier les équations et contraintes du système hybride initial. En particulier, on détaille un moyen permettant de reformuler les contraintes hybrides définissant une plage de fonctionnement lorsqu'un composant est activé. Le modèle obtenu est un modèle continu qui admet les trajectoires du modèle hybride comme un

cas particulier (lorsque $\omega^{ji}(t) \in \{0;1\}$).
- on applique alors sur le modèle créé une méthode itérative d'optimisation dynamique hybride, qui alterne entre la résolution d'un problème de commande optimale sur le modèle relaxé et la génération de trajectoires de commande admissibles pour le modèle hybride (non relaxé). A chaque itération, un algorithme de raffinement de la grille temporelle utilisée pour l'optimisation est appliqué. Son objectif est la localisation des instants de commutation optimaux pour le système hybride, et le contrôle de l'erreur de la méthode d'optimisation utilisée.

Dans cette approche, les points suivants ont ainsi été développés :

Méthode de collocation avec contrôle de l'erreur

La méthode d'optimisation dynamique utilisée dans cette thèse pour résoudre le problème relaxé est la méthode de collocation, implémentée dans le logiciel JModelica. Cette méthode s'appuie sur une discrétisation de l'horizon temporel, des commandes et des états du système et transforme le problème de commande optimale en un problème d'optimisation non linéaire (NLP). Cette approximation des trajectoires de commande et d'état s'accompagne d'une erreur qui dépend de la finesse de la grille temporelle utilisée et de la dynamique du système. On propose donc de raffiner de façon itérative la grille temporelle support afin d'atteindre une précision donnée. Deux critères pour évaluer l'erreur relative de la méthode de collocation sur chaque pas de temps sont ainsi proposés dans cette thèse et servent de base au raffinement de la grille courante. Ce raffinement prend également en compte la trajectoire des commandes binaires relaxées : la grille est ainsi affinée sur les pas de temps où cette trajectoire n'a pas un caractère binaire ($\omega^{ji}(t) \in \{0;1\}$).

Méthode SURminT

La méthode permettant de générer les trajectoires des variables binaires est une variante de la méthode *Sum Up Rounding* (SUR) [54]. Dans la méthode SUR initiale, aucune restriction n'est faite sur les commutations, et il est possible d'obtenir des commutations à chaque instant de la grille temporelle support, ce qui n'est pas souhaitable dans le cas des centrales d'énergie, où des commutations fréquentes peuvent endommager les composants, ou n'être tout simplement pas autorisées (respect des temps minimum de marche et d'arrêt). L'amélioration principale de la méthode SUR que l'on propose dans cette thèse, dénommée SURminT concerne la prise en compte de temps minimums T_{min} pour chacun des modes sans pour autant restreindre les commutations à des instants $t = k * T_{min}$. La méthode prend également en compte le mode à l'instant initial dans cette contrainte (ainsi que la durée depuis laquelle ce mode est actif).

Par ailleurs, les règles de modélisation formulées dans cette thèse permettent de définir le mode courant d'un système hybride comme la réunion des modes de ses différents composants (forme normale conjonctive CNF), alors que dans sa forme initiale l'algorithme SUR nécessitait une énumération par l'utilisateur de toutes les combinaisons possibles à l'échelle du système (forme normale disjonctive DNF). Des temps minimum de marche et d'arrêt peuvent alors être exprimés pour chaque composant. Cette nouvelle formulation s'intègre mieux dans l'approche orientée objet de Modelica.

La méthode itérative de résolution du problème d'optimisation dynamique hybride est synthétisée en Figure 3.9. La méthode SURminT qui est une brique de cette méthode est décrite

dans l'Algorithme 6.

Coûts de transition dans la méthode de collocation

Des modifications ont été apportées à l'algorithme de collocation pour prendre en compte des coûts de transition ou plus généralement des coûts de modulation à la hausse ou à la baisse des entrées constantes par morceaux. Dans le cas des centrales d'énergie, les coûts de transition peuvent typiquement être utilisés pour modéliser les coûts de démarrage (ou d'arrêt) d'un composant. Ils font partie des aspects hybrides et sont liés au passage d'une commande binaire ω^{ji} de 0 à 1 (ou vice-versa). On montre que ces transitions peuvent se reformuler comme un problème MPCC (*Mathematical Problem with Complementarity Constraints*). Les coûts de transition peuvent alors être intégrés dans le NLP résultant de la méthode de collocation. Cependant, les hypothèses de convergence de la méthode SUR ne sont plus vérifiées lorsque la fonction objectif inclut des coûts de transition : en lieu de celle-ci, l'utilisation d'une méthode d'arrondi simple avec respect des temps minimaux de commutation a fourni des résultats satisfaisants pour le cas d'étude.

Résultats obtenus

La méthodologie développée est une alternative à la Programmation Linéaire en Nombres Entiers pour le pilotage optimisé de centrales de production d'énergie, lorsque des contraintes sur les variables physiques doivent être prises en comptes ou qu'une modélisation linéaire de l'installation s'avère inappropriée. La méthode est démontrée sur deux cas d'étude et permet de générer des trajectoires optimisées qui respectent les contraintes physiques des installations, les durées minimales de marche et d'arrêt des différents composants, ainsi que leurs coûts de démarrage.

5.1.2 Commande prédictive TL-MPC

La méthode précédente, tout comme la Programmation Linéaire en Nombres Entiers utilisée actuellement à EDF dans le logiciel PILOT, permettent de générer une trajectoire optimisée pour l'installation. Cette optimisation se fait hors ligne sur un horizon long en se basant sur des prévisions pour les entrées incertaines du système (typiquement des courbes de charge ou de prix). Cependant, deux sources d'erreur doivent être prises en compte pour le suivi de cette trajectoire optimisée en temps réel : d'une part les entrées incertaines ont une réalisation différente des prévisions, d'autre part les simplifications nécessaires pour obtenir un modèle (linéaire ou non-linéaire) adapté à l'optimisation sur un horizon long font que le système réel finira par dévier des trajectoires prévues. On propose dans cette thèse un algorithme de commande prédictive pour assurer le suivi des trajectoires de commandes optimales et le respect des contraintes de l'installation en temps réel. L'algorithme de commande prédictive, nommé *Tangent Linear Model Predictive Control* (TL-MPC) s'appuie sur un modèle physique détaillé de l'installation. De ce modèle sont dérivés en temps réel des modèles linéarisés tangents de l'installation sur l'horizon du MPC le long d'une trajectoire nominale, en mettant à profit les capacités de traitement symbolique (mise sous forme de modèle d'état) et de différentiation automatique offertes par Modelica.

Stabilité de la commande TL-MPC dans le cas nominal

Un premier résultat de stabilité de la commande TL-MPC est fourni dans le cas nominal (en l'absence de perturbations et d'erreur de modélisation), lorsque les trajectoires de référence sont données en l'état et la commande. La preuve utilise un contrôleur terminal capable de stabiliser le système dans un ensemble terminal à la fin de l'horizon de prédiction du MPC. L'utilisation de modèles linéarisés tangents complexifie sensiblement l'analyse de stabilité par rapport à l'utilisation de modèles linéarisés autour d'un point d'équilibre. Une des raisons principales est que le domaine admissible des contraintes, le contrôleur terminal et l'ensemble terminal utilisés dans la preuve de stabilité deviennent variants dans le temps eux aussi. On donne de façon théorique une méthode pour calculer les contrôleurs et ensembles terminaux par la résolution d'une inégalité matricielle (LMI).
La variation dans le temps du domaine des contraintes rend aussi difficile la preuve de la faisabilité récursive. Plutôt que de faire l'hypothèse que les trajectoires de référence deviennent stationnaires au delà d'un certain temps (ce qui n'est pas notre cas), on propose de garantir à chaque itération de commande la faisabilité pour le problème MPC suivant. Cette stratégie a du sens si l'on suppose que l'horizon MPC est suffisamment long pour qu'une procédure de sécurité puisse être déclenchée si l'on détecte qu'une violation des contraintes va se produire.

Commande TL-MPC adaptée aux centrales d'énergie

Outre les difficultés théoriques évoquées précédemment, le modèle d'optimisation et le modèle physique détaillé ont souvent des vecteurs d'état différents. Pour cette raison, l'architecture MPC implémentée dans cette thèse utilise des trajectoires de référence données pour les sorties communes aux deux modèles plutôt que sur les états. Une conséquence de ce changement de paradigme est que l'ensemble terminal évoqué précédemment ne peut plus être défini. Cependant, on observe en pratique un bon comportement des systèmes sous commande TL-MPC pour des horizons de prédictions suffisamment longs et une période d'échantillonnage adaptée. On décrit donc une implémentation modifiée de la commande prédictive. Cette implémentation prend en compte certaines spécificités propres aux systèmes énergétiques, notamment la normalisation des variables (amélioration du conditionnement numérique), et la non remise en cause des décisions d'engagement (il n'est pas possible d'ajuster les commandes d'un composant inactivé).
La méthode a été démontrée en simulation en utilisant le modèle physique d'une installation développé avec la librairie EDF ThermoSysPro.

5.2 Perspectives de développement

5.2.1 Optimisation dynamique hybride

Plusieurs pistes d'amélioration sont envisagées pour la méthode de collocation hybride :

Durées minimales des modes

Jusqu'alors, les durées minimales des modes sont définies pour chaque disjonction. Cela signifie par exemple que la même valeur est utilisée pour les durée minimales de marche et

d'arrêt d'un composant. La méthode SURminT doit être adaptée pour prendre en compte une durée minimale différente pour chacun des termes d'une disjonction.

Modes actifs à l'instant initial

La méthode SURminT prend en compte la durée depuis laquelle les modes sont actifs à l'instant initial lors de la génération de trajectoires binaires. Ainsi, si le mode actif d'une disjonction j n'a pas encore atteint sa durée minimale, les entrées ω^{ji} correspondantes sont maintenues à leur valeur le temps nécessaire, jusqu'à un instant t^j_{min}. Elles ne peuvent donc pas être optimisées à proprement parler sur l'intervalle $[t_0, t^j_{min}]$. L'optimisation sur les variables ω^{ji} relaxées devrait ainsi commencer à partir de t^j_{min}. Cela nécessiterait de modifier le code de collocation en éliminant les variables ω^{ji}_k des variables de décision, sur les éléments k correspondants.

Raffinement de la grille temporelle

Dans certains cas, des problèmes de convergence ont été observés lorsqu'une grille trop fine est utilisée ou que le problème NLP devient trop grand. A l'heure actuelle, la méthode de raffinement ne peut qu'accroître le nombre de pas de temps utilisés dans la méthode de collocation, bien que dans une version initiale des fusions entre deux éléments consécutifs étaient considérées (mais cela restreignait les instants de commutation possibles). Une méthode d'adaptation de la grille permettant d'augmenter localement l'ordre de la méthode de collocation (plutôt que le nombre d'éléments) pourrait également être envisagée afin d'augmenter la précision numérique à moindre coût. Il pourrait enfin être intéressant de voir si l'utilisation d'une grille temporelle flexible (où les durées des éléments sont une variable d'optimisation) permettrait une meilleure localisation des instants optimaux de commutation (au prix d'une nonlinéarité accrue dans le problème NLP).

5.2.2 Commande prédictive TL-MPC

Extension des preuves de stabilité

La preuve de stabilité présentée dans cette thèse, valable dans un cadre nominal, a soulevé les difficultés théoriques posées par la formulation de la commande TL-MPC. Cette preuve fait appel à des contrôleurs linéaires à temps variants qui stabilisent le système dans un ensemble terminal. La méthode proposée pour les construire suppose cependant que la matrice de linéarisation B soit constante et que l'on connaisse l'enveloppe convexe des matrices de linéarisation A_k. Dans la pratique, une méthode numérique pour déterminer cette enveloppe convexe doit encore être trouvée, et il faudra vérifier par la suite que le nombre des matrices sommets obtenues est compatible avec une résolution par LMI. Par ailleurs, l'utilisation d'une enveloppe convexe amène du conservatisme dans le calcul des trajectoires et des ensembles terminaux. Des solutions devront être apportées si ce conservatisme est trop important. On pourrait par exemple envisager de regrouper les points en plusieurs zones.
Par ailleurs, les incertitudes dues à la présence de perturbations et les erreurs de modélisation doivent être considérées pour compléter la preuve de stabilité. L'étude de la stabilité entrée état (ISS ou ISpS) brièvement introduite en 4.2.5 semble être le cadre adapté à cette étude. Ces méthodes nécessitent cependant de connaître des bornes sur l'incertitude. Un premier

pas dans ce sens est la quantification de l'erreur de modélisation montrée en Annexe D. La prochaine étape consiste maintenant à borner l'erreur due aux perturbations.

Compromis complexité précision

Le MPC utilise des modèles linéarisés à temps variant, ce qui permet d'avoir un modèle valable tout au long de la trajectoire (notamment lors de transitoires), mais ajoute une certaine complexité à la formulation. Il pourra être intéressant de quantifier le gain apporté par cette complexité en fonction des applications étudiées.

Plateforme de démonstration

Les solutions proposées ont été testées en simulation et semblent pouvoir fonctionner en temps réel. Aussi le développement d'une plateforme de démonstration en temps réel (ou accéléré) pourrait permettre de valider les aspects temps réel de la méthode. Une architecture logicielle utilisant le standard de communication OPC de Microsoft (*OLE for Process Control*) ou le standard libre FMI (*Functionnal Mock-up Interface*) peut être envisagée. Les effets des erreurs de modèles pourront être testés en utilisant comme centrale virtuelle le même modèle que le modèle du MPC mais avec des variations sur les paramètres.

Bibliographie

[1] Hubertus Tummescheit. *Design and Implementation of Object-Oriented Model Libraries using Modelica*. Lund, 2002.

[2] A. Tica, H. Gueguen, D. Dumur, D. Faille, and F. A. Davelaar. Hierarchical Model Predictive Control Approach for Start-up Optimization of a Combined Cycle Power Plant. In *8th Power Plant and Power Systems Control Symposium*, September 2012.

[3] B. El Hefni, D. Bouskela, and G. Lebreton. Dynamic modelling of a combined cycle power plant with ThermoSysPro. In *Proceedings 8th Modelica Conference*, March 2011.

[4] O. Deneux, B. El Hafni, B. Péchiné, E. Di Penta, G. Antonucci, and P. Nuccio. Establishment of a Model for a Combined Heat and Power Plant with ThermosysPro Library. *Procedia Computer Science*, 19 :746–753, January 2013.

[5] Sylvain Girard. *Diagnostic du colmatage des générateurs de vapeur à l'aide de modèles physiques et statistiques*. PhD thesis, Ecole Nationale des Mines de Paris, 2012.

[6] Olivier Deneux, Bruno Péchiné, and Manon Fouquet. Optimization of Design and Operation of a Combined Heat and Power Plant, by Use of a 1d-Physical Model. page V002T08A006. ASME, July 2013.

[7] F. Casella, F. Donida, and J. Akesson. Object-Oriented Modeling and Optimal Control : A Case Study in Power Plant Start-Up. In *18th IFAC World Congress*, August 2011.

[8] Adrian Tica, Hervé Guéguen, Didier Dumur, Damien Faille, and Frans Davelaar. Design of a combined cycle power plant model for optimization. *Applied Energy*, 98 :256–265, October 2012.

[9] Per-Ola Larsson, Håkan Runvik, Stéphane Velut, Sara Modarres Razavi, Andreas Nilsson, Bohlin, Markus, and Funkuist, Jonas. Decision Support for Short-Term Production Planning of District Heating using Non-linear Programming. Technical report, Värmeforsk Serviceaktiebolag, November 2014.

[10] Alexander W. Dowling, John P. Eason, Jinliang Ma, David C. Miller, and Lorenz. T. Biegler. Coal Oxycombustion Power Plant Optimization Using First Principles and Surrogate Boiler Models. *Energy Procedia*, 63 :352–361, 2014.

[11] Léo,Jessica. *Modélisation et conduite optimale d'un cycle combiné hybride avec source solaire et stockage*. PhD thesis, GIPSA-Grenoble, 2015.

[12] Rolf Findeisen and Frank Allgöwer. An introduction to nonlinear model predictive control. *21st Benelux Meeting on Systems and Control*, 11, 2002.

[13] M. Diehl, H. J. Ferreau, and N. Haverbeke. Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation. pages 391–417. Springer, 2009.

[14] Modelica Association. *Modelica® - A Unified Object-Oriented Language for Systems Modeling.* July 2014.

[15] W. Wagner, J. R. Cooper, A. Dittmann, J. Kijima, H.-J. Kretzschmar, A. Kruse, R. Mareš, K. Oguchi, H. Sato, I. Stöcker, O. Šifner, Y. Takaishi, I. Tanishita, J. Trübenbach, and Th. Willkommen. The IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam. *Journal of Engineering for Gas Turbines and Power*, 122(1) :150–184, January 2000.

[16] Sanford Gordon and Bonnie J. Mcbride. Computer program for calculation of complex chemical equilibrium compositions and applications. Part 1 : Analysis. Technical Report NASA-RP-1311, E-8017, NAS 1.61 :1311, NASA, October 1994.

[17] L.R. Petzold. Description of DASSL : a differential/algebraic system solver. In *10. international mathematics and computers simulation congress on systems simulation and scientific computation, Montreal, Canada, 9 Aug 1982.* September 1982.

[18] François E. Cellier and Ernesto Kofman. *Continuous system simulation.* Springer, New York, 2006.

[19] Robert Tarjan. Depth-First Search and Linear Graph Algorithms. *SIAM Journal on Computing*, 1(2) :146–160, June 1972.

[20] Francesco Casella. Properties of Differential-Algebraic Equation Systems (DAE), 2011.

[21] H. Elmqvist and M. Otter. Methods for tearing systems of equations in object oriented modelling. *Proceedings of the Conference on Modelling and Simulation*, pages 326–332, 1994.

[22] Constantinos C. Pantelides. The Consistent Initialization of Differential-Algebraic Systems. *SIAM Journal on Scientific and Statistical Computing*, 9(2) :213–231, March 1988.

[23] S. E. Mattsson, H. Olsson, and H. Elmqvist. Dynamic Selection of States in Dymola. In *Modelica Workshop 2000 Proceedings*, pages 61–67, October 2000.

[24] Joel Andersson, Johan Åkesson, and Moritz Diehl. CasADi : A Symbolic Package for Automatic Differentiation and Optimal Control. In Shaun Forth, Paul Hovland, Eric Phipps, Jean Utke, and Andrea Walther, editors, *Recent Advances in Algorithmic Differentiation*, volume 87 of *Lecture Notes in Computational Science and Engineering*, pages 297–307. Springer Berlin Heidelberg, 2012.

[25] Ernst Hairer, Gerhard Wanner, and Syvert P. Norsett. Solving ordinary differential equations i : Nonstiff problems. *Springer series in computational mathematics*, 14, 1996.

[26] P. Borne and J.P. Richard. *Modélisation et identification des processus.* Number vol. 2 in Automatique : identification et modélisation des processus. Technip, 1992.

[27] Hans Olsson, Hubertus Tummescheit, Hilding Elmqvist, AB Dynasim, and AB Modelon. Using automatic differentiation for partial derivatives of functions in Modelica. 2005.

[28] Frank H. Clarke. *Optimization and nonsmooth analysis.* Number 5 in Classics in applied mathematics. SIAM, Philadelphia, 1990.

[29] Emmanuel Trélat. *Contrôle optimal : théorie et applications*. 2013.

[30] Metin Türkay and Ignacio E. Grossmann. Logic-based {MINLP} algorithms for the optimal synthesis of process networks. *Computers & Chemical Engineering*, 20(8) :959 – 978, 1996.

[31] Jan Oldenburg and Wolfgang Marquardt. Disjunctive modeling for optimal control of hybrid systems. *Computers & Chemical Engineering*, 32(10) :2346 – 2364, 2008.

[32] Lorenz T. Biegler. Chapter 11. Process Optimization with Complementarity Constraints. In *Nonlinear programming : concepts, algorithms, and applications to chemical processes*, MOS-SIAM series on optimization. Society for Industrial and Applied Mathematics : Mathematical Programming Society, Philadelphia, 2010.

[33] William F. Feehery and Paul I. Barton. Dynamic optimization with state variable path constraints. *Computers & Chemical Engineering*, 22(9) :1241–1256, August 1998.

[34] Martin Schlegel and Wolfgang Marquardt. Detection and exploitation of the control switching structure in the solution of dynamic optimization problems. *Journal of Process Control*, 16(3) :275–290, March 2006.

[35] H. G. Bock and K. J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *9th IFAC World Congress Budapest*, July 1984.

[36] L. Wirsching. *An SQP Algorithm with Inexact Derivatives for a Direct Multiple Shooting Method for Optimal Control Problems*. PhD thesis, Universität Heidelberg, 2006.

[37] Fady Assassa and Wolfgang Marquardt. Dynamic optimization using adaptive direct multiple shooting. *Computers & Chemical Engineering*, 60 :242–259, January 2014.

[38] Shivakumar Kameswaran and Lorenz T. Biegler. Simultaneous dynamic optimization strategies : Recent advances and challenges. *Computers & Chemical Engineering*, 30(10-12) :1560–1575, September 2006.

[39] Lorenz T. Biegler. Chapter 10. Simultaneous Methods for Dynamic Optimization. In *Nonlinear programming : concepts, algorithms, and applications to chemical processes*, MOS-SIAM series on optimization. Society for Industrial and Applied Mathematics : Mathematical Programming Society, Philadelphia, 2010.

[40] Fredrik Magnusson and Johan Åkesson. Collocation Methods for Optimization in a Modelica Environment. *9th International Modelica Conference*, September 2012.

[41] Ignacio E. Grossmann and Lorenz T. Biegler. Part II. Future perspective on optimization. *Computers & Chemical Engineering*, 28(8) :1193 – 1218, 2004.

[42] MarcoA. Duran and IgnacioE. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, 36(3) :307–339, 1986.

[43] María Lorena Bergamini, Ignacio Grossmann, Nicolás Scenna, and Pío Aguirre. An improved piecewise outer-approximation algorithm for the global optimization of {MINLP} models involving concave and bilinear terms. *Computers & Chemical Engineering*, 32(3) :477 – 493, 2008.

[44] Michael N. Jung, Christian Kirches, and Sebastian Sager. On Perspective Functions and Vanishing Constraints in Mixed-Integer Nonlinear Optimal Control. In Michael Jünger and Gerhard Reinelt, editors, *Facets of Combinatorial Optimization*, pages 387–417. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[45] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106 :25–57, 2006.

[46] Christian Kanzow and Alexandra Schwartz. A New Regularization Method for Mathematical Programs with Complementarity Constraints with Strong Convergence Properties. *SIAM Journal on Optimization*, 23(2) :770–798, April 2013.

[47] Arvind U. Raghunathan, M. Soledad Diaz, and Lorenz T. Biegler. An {MPEC} formulation for dynamic optimization of distillation operations. *Computers & Chemical Engineering*, 28(10) :2037 – 2052, 2004. <ce :title>Special Issue for Professor Arthur W. Westerberg</ce :title>.

[48] Oliver Stein, Jan Oldenburg, and Wolfgang Marquardt. Continuous reformulations of discrete–continuous optimization problems. *Computers & Chemical Engineering*, 28(10) :1951 – 1966, 2004.

[49] Hande Y. Benson, Arun Sen, David F. Shanno, and Robert J. Vanderbei. Interior-Point Algorithms, Penalty Methods and Equilibrium Problems. *Comput. Optim. Appl.*, 34(2) :155–182, June 2006.

[50] Sven Leyffer, Gabriel López-calva, and Jorge Nocedal. Interior methods for mathematical programs with complementarity constraints. *SIAM J. Optim*, 17 :52–77, 2004.

[51] Sangbum Lee and Ignacio E. Grossmann. Global optimization of nonlinear generalized disjunctive programming with bilinear equality constraints : applications to process networks. *Computers & Chemical Engineering*, 27(11) :1557 – 1575, 2003.

[52] Jan Oldenburg and Wolfgang Marquardt. Optimization of Discrete-Continuous Dynamic Systems Based on Disjunctive Programming. *PAMM*, 5(1) :51–54, 2005.

[53] Sorin C. Bengea and Raymond A. DeCarlo. Optimal control of switching systems. *Automatica*, 41(1) :11 – 27, 2005.

[54] Sebastian Sager. *Numerical methods for mixed-integer optimal control problems*. Der Andere Verl, Tönning, 2005.

[55] Shangming Wei, Kasemsak Uthaichana, Miloš Žefran, Raymond A. DeCarlo, and Sorin Bengea. Applications of numerical optimal control to nonlinear hybrid systems. *Nonlinear Hybrid Control Systems*, 1(2) :264–279, June 2007.

[56] Sebastian Sager, Hans Georg Bock, and Moritz Diehl. The integer approximation error in mixed-integer optimal control. *Mathematical programming*, 133(1-2) :1–23, 2012.

[57] John T Betts. *Practical methods for optimal control using nonlinear programming*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2001.

[58] John T. Betts, Neil Biehn, Stephen L. Campbell, and William P. Huffman. Compensating for order variation in mesh refinement for direct transcription methods. *Journal of Computational and Applied Mathematics*, 125(1-2) :147–158, December 2000.

[59] Luís Tiago Paiva and Fernando A. C. C. Fontes. Adaptive time–mesh refinement in optimal control problems with state constraints. *Discrete and Continuous Dynamical Systems*, 35(9) :4553–4572, 2015.

[60] Fredrik Magnusson and Johan Åkesson. Dynamic Optimization in JModelica.org. *Processes*, 3(2) :471–496, June 2015.

[61] E. F. Camacho and C. Bordons. *Model Predictive Control in the Process Industry. Second Edition.* Springer-Verlag, London, England, 2004.

[62] Jan Maciejowski. *Predictive Control with Constraints.* Prentice Hall, 2000.

[63] Frank Allgöwer, Rolf Findeisen, and K.Nagy Zoltan. Nonlinear model predictive control : From theory to application. *Journal of Chinese Institute for Chemical Engineering*, 35(3) :299–315, 2004.

[64] Moritz Diehl, Hans Joachim Ferreau, and Niels Haverbeke. Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation. In Manfred Morari, Manfred Thoma, Lalo Magni, Davide Martino Raimondo, and Frank Allgöwer, editors, *Nonlinear Model Predictive Control*, volume 384, pages 391–417. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[65] Magdalena Axelsson. Nonlinear Model Predictive Control in JModelica.org. MSc Thesis, Department of Automatic Control Lund University, Lund, 2015.

[66] Quoc Tran Dinh, Carlo Savorgnan, and Moritz Diehl. Real-time sequential convex programming for nonlinear model predictive control and application to a hydro-power plant. pages 5905–5910. IEEE, December 2011.

[67] Francesco Borrelli, Paolo Falcone, Tamas Keviczky, and Jahan Asgari. MPC-based approach to active steering for autonomous vehicle systems. *International Journal of Vehicle Autonomous Systems*, 3(2) :265–291, 2005.

[68] D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control : Stability and optimality. *Automatica*, 36(6) :789–814, June 2000.

[69] Jamal Daafouz and Jacques Bernussou. Parameter dependent Lyapunov functions for discrete time systems with time varying parametric uncertainties. *Systems & control letters*, 43(5) :355–359, 2001.

[70] Ricardo C. L. F. Oliveira and Pedro L. D. Peres. Time-varying discrete-time linear systems with bounded rates of variation : Stability analysis and control design. *Automatica*, 45(11) :2620 – 2626, 2009.

[71] Timm Faulwasser and Rolf Findeisen. A model predictive control approach to trajectory tracking problems via time-varying level sets of Lyapunov functions. pages 3381–3386. IEEE, December 2011.

[72] S. V. Raković and D. Q. Mayne. A simple tube controller for efficient robust model predictive control of constrained linear discrete-time systems subject to bounded disturbances. In *in Proceedings of the 16th IFAC World Congress*, 2005.

[73] Saša V Rakovic, Basil Kouvaritakis, Rolf Findeisen, and Mark Cannon. Simple homothetic tube model predictive control. In *Proceedings of the 19th International Symposium on Mathematical Theory of Networks and Systems–MTNS*, volume 5, 2010.

[74] T. Manrique, M. Fiacchini, T. Chambrion, and G. Millerioux. MPC tracking under time-varying polytopic constraints for real-time applications. pages 1480–1485. IEEE, June 2014.

[75] Michael Grant and Stephen Boyd. *CVX : Matlab Software for Disciplined Convex Programming, version 2.1.* March 2014.

[76] Michael Grant and Stephen Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning*

and Control, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008. http ://stanford.edu/ boyd/graph_dcp.html.

[77] D. Limon, T. Alamo, F. Salas, and E. F. Camacho. Input to State Stability of Min-max MPC Controllers for Nonlinear Systems with Bounded Uncertainties. *Automatica*, 42(5) :797–803, May 2006.

[78] Maxime Penet. *Robust Nonlinear Model Predictive Control based on Constrained Saddle Point Optimization : Stability Analysis and Application to Type 1 Diabetes.* Theses, Supélec, October 2013.

[79] D. Limon, T. Alamo, Davide Martino Raimondo, D. Muñoz de la Peña, J.M. Bravo, A. Ferramosca, and E. F. Camacho. Input-to-State Stability : A Unifying Framework for Robust Model Predictive Control. In Lalo Magni, Davide Martino Raimondo, Frank Allgöwer, Manfred Morari, and Manfred Thoma, editors, *Nonlinear Model Predictive Control*, volume 384 of *Lecture Notes in Control and Information Sciences*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[80] P. Maybeck. *Stochastic Models, Estimation and Control*, volume 2. Academic press edition, 1982.

[81] International Association for the Properties of Water and Steam. Release on the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam. Technical report, IAPWS, Erlangen, Germany, 1997.

[82] C. Bissuel, C. Goude, and B. Pechine. Heat Load Forecasting. Krakow, 2013. Taler, J. (ed.).

[83] Michel Bierlaire. *Introduction a l'optimisation differentiable.* Presses polytechniques et universitaires romandes, Lausanne, 2006.

[84] Lorenz T. Biegler. Chapter 4. Concepts of Constrained Optimization. In *Nonlinear programming : concepts, algorithms, and applications to chemical processes*, MOS-SIAM series on optimization. Society for Industrial and Applied Mathematics : Mathematical Programming Society, Philadelphia, 2010.

[85] Tim Hoheisel, Christian Kanzow, and Alexandra Schwartz. Theoretical and numerical comparison of relaxation methods for mathematical programs with complementarity constraints. *Mathematical Programming*, 137(1-2) :257–288, 2013.

Annexes

ANNEXE A

Optimisation en dimension finie

On commence ici par certains rappels sur la résolution de problèmes d'optimisation sous contrainte. Le vecteur des variables d'optimisation est noté $\mathbf{x}$. On s'intéresse au problème suivant :

$$\min_{\mathbf{x}} J(\mathbf{x}) \tag{A.1}$$
$$\text{s.c.} h_I(\mathbf{x}) \leq 0 \tag{A.2}$$
$$h_E(\mathbf{x}) = 0 \tag{A.3}$$

A.1 Conditions nécessaires d'optimalité

Définition (contraintes actives) Les contraintes actives en un point $\bar{\mathbf{x}}$ sont les contraintes inégalités indexées par i pour lesquelles $h_I^{(i)}(\bar{\mathbf{x}}) = 0$. On note $\mathcal{A}(\bar{\mathbf{x}})$ l'ensemble de ces indices.

Une solution $\mathbf{x}^*$ est optimale si elle est admissible et qu'aucune variation autour de ce point n'améliore la fonction objectif J. Si l'objectif et les contraintes sont suffisamment régulières au voisinage du point optimal $\mathbf{x}^*$, alors une approximation linéaire est possible et permettra de dériver des conditions d'optimalité. Cette condition de régularité peut s'exprimer par diverses conditions sur les contraintes que l'on appelle qualification des contraintes. On introduit pour le problème (A.1) la fonction $\mathcal{L}$ nommée Lagrangien, qui consiste à adjoindre à la fonction objectif initiale une pénalisation de la violation des contraintes, avec des paramètres de pénalisation λ_i (contraintes d'égalité) et μ_i (contraintes d'inégalité), nommés multiplicateurs de Lagrange. On cherchera alors à minimiser le Lagrangien plutôt que la fonction objectif initiale. Le Lagrangien s'écrit

$$\mathcal{L}(\mathbf{x}, \lambda, \mu) = J(\mathbf{x}) + \lambda^T h_E(\mathbf{x}) + \mu^T h_I(\mathbf{x}) \tag{A.4}$$

Les vecteurs λ et μ sont des vecteurs appelés multiplicateurs de Lagrange. Chaque composante de ce vecteur est associée à une contrainte. Ils représentent le coût associé au respect de cette contrainte. Il est possible de dériver pour un point $\mathbf{x}^*$ des conditions nécessaires d'optimalité

pour le problème (A.1) : celles-ci sont nommées conditions de Karush Kuhn Tucker (KKT) [83][84]. Les premières conditions KKT sont obtenues d'après des approximations au premier ou second ordre du problème (A.1) autour de $\mathbf{x}^*$. Une condition dite de qualification des contraintes est alors présente pour garantir le bien fondé de ces approximations autour de $\mathbf{x}^*$. Les conditions KKT sont les suivantes :

- *Stationnarité du Lagrangien :* La condition de stationnarité est $\nabla_v \mathcal{L}(\mathbf{x}^*, \lambda^*, \mu^*) = 0$
- *Faisabilité :* Les contraintes $h_I(\mathbf{x}^*) \leq 0$, $h_e(\mathbf{x}^*) = 0$ sont satisfaites
- *Complémentarité :* Les contraintes d'inégalité inactives $h_I^j(\mathbf{x}) < 0$ (j^{me} contrainte) n'influent pas la solution et doivent avoir un multiplicateur associé λ_j nul. Pour les contraintes égalité et les contraintes inégalité actives $h_I^j = 0$, les multiplicateurs de Lagrange doivent interdire les déplacements vers l'extérieur du domaine admissible, c'est à dire dégrader la fonction objectif, d'où les conditions suivantes sur μ et λ : $h_I^j(\mathbf{x}^*)^T.\lambda^* = 0, \mu \geq 0$
- *Conditions de second ordre :* Afin de bien identifier un minimum local et pas un maximum ni un point selle, les conditions suivantes sont ajoutées. Elles traduisent le comportement de la solution pour un déplacement dans la direction $d > 0$ autour de $\mathbf{x}^*$. Ainsi, une courbure positive du Lagrangien dans toutes les directions admissibles d est garantie par

$$\begin{aligned}
&\nabla_{xx}\mathcal{L}(\mathbf{x}^*, \lambda^*, \mu^*) \simeq 0 \\
&\quad \nabla_v h_E(\mathbf{v}^*)^T d = 0 \quad \forall d > 0 \\
&\quad \nabla_v h_I^j(\mathbf{x}^*)^T d = 0, j \in \{j \mid h_I^j(\mathbf{x}^*) = 0, \mu_j^* > 0\} \\
&\quad \nabla_v h_I^j(\mathbf{x}^*)^T d \leq 0, j \in \{j \mid h_I^j(\mathbf{x}^*) = 0, \mu_j^* = 0\}
\end{aligned}$$

A.2 Qualification des contraintes

Les conditions d'optimalité KKT sont basées sur une approximation linéaire des contraintes actives autour d'un point. Or comme le problème original est non linéaire, il est nécessaire que les contraintes soient suffisamment régulières en son voisinage pour garantir le bon comportement des algorithmes d'optimisation. Plusieurs conditions sur les contraintes, plus ou moins contraignantes, permettent de garantir cette propriété. On peut citer

- la contrainte d'indépendance linéaire des contraintes actives en un point $\mathbf{x}^*$ (LICQ) qui requiert l'indépendance linéaire du gradient des contraintes actives et des contraintes égalité en $\mathbf{x}^*$:

$$\nabla h_E(\mathbf{x}^*), \quad \nabla h_I^{(i)}(\mathbf{x}^*) \quad i \in \mathcal{A}(\mathbf{x}^*)$$

  LICQ garantit l'existence de multiplicateurs de Lagrange bornés et uniques [39]
- la contrainte MFCQ, moins restrictive, qui ne requiert l'indépendance linéaire que pour le gradient des contraintes d'égalité et l'existence d'une direction de recherche admissible d autour du point $\mathbf{x}^*$ (on utilise ici une dérivée directionnelle pour les contraintes inégalité plutôt que le gradient, qui correspond à toutes les directions de recherche possibles).

$$\nabla h_E(\mathbf{x}^*)^T d < 0, \quad \nabla h_I^{(i)}(\mathbf{x}^*)^T d \quad i \in \mathcal{A}(\mathbf{x}^*)$$

La condition MFCQ garantit l'existence de multiplicateurs de Lagrange bornés (mais non leur unicité).

A.3 Qualification des contraintes pour les problèmes MPCC

Un problème MPCC est un problème d'optimisation contenant le jeu de contraintes suivant

$$\begin{aligned} G_1^i(x) &\geq 0 \\ G_2^i(x) &\geq 0 \\ G_1^i(x)G_2^i(x) &= 0, \quad \forall i \in [1..m_c] \end{aligned}$$

On l'abrègera parfois en $G_1^i(x) \perp G_2^i$.

Comme les qualifications habituelles des contraintes (LICQ, MFCQ...) échouent pour les problèmes de complémentarité, de nouvelles qualifications, plus faibles, sont introduites. Elles garantissent à certaines méthodes de converger vers des points stationnaires. Les qualifications des contraintes adaptées aux MPCC, ainsi que les preuves de convergence détaillés dans [46] se basent sur la définition des ensembles suivants

$$\begin{aligned} \mathcal{A} &= i \text{ t.q. } h_I(x) = 0 \\ \mathcal{A}_{00} &= i \text{ t.q. } G_1^i(x) = 0, G_2^i(x) = 0 \\ \mathcal{A}_{01} &= i \text{ t.q. } G_1^i(x) = 0, G_2^i(x) > 0 \\ \mathcal{A}_{10} &= i \text{ t.q. } G_1^i(x) > 0, G_2^i(x) = 0 \end{aligned}$$

On note ν_1 les multiplicateurs de KKT associés à $G_1(x)$ et ν_2 ceux associés à $G_2(x)$ et λ ceux associés aux autres contraintes $g_{in}(x)(x)$.

On définit différents types de points stationnaires pour les MPCC :

- les points faiblement stationnaires :

$$\nabla_x \mathcal{L} = 0, \quad \nu_2^i = 0 \quad \forall i \in \mathcal{A}_{01}, \quad \nu_1^i = 0 \quad \forall i \in \mathcal{A}_{10}, \quad \lambda^i > 0, \lambda^i g_{in} = 0 \; \forall i \in \mathcal{A}$$

- les points C-stationnaires : les points faiblement stationnaires vérifiant

$$\nu_1^i \nu_2^i \geq 0, \quad \forall i \in \mathcal{A}_{00}$$

- les points M-stationnaires : les points faiblement stationnaires vérifiant

$$\nu_1^i \nu_2^i = 0, \; \textbf{OU} \; \nu_1^i > 0 \; \textbf{OU} \; \nu_2^i > 0, \quad \forall i \in \mathcal{A}_{00}$$

- les points fortement stationnaires : les points faiblement stationnaires vérifiant

Plusieurs approches existent pour relaxer la condition de complémentarité [85], et générer une suite de problèmes convergents vers le problème initial. Les différents algorithmes de relaxation qui peuvent être mis en place pour résoudre le MPCC garantissent la convergence vers un type de point stationnaire.

ANNEXE B

Rappels mathématiques

B.1 Méthode de Newton

La méthode de Newton est une méthode de recherche locale du zéro d'une fonction différentiable $r : \mathbb{R}^x \to \mathbb{R}^y$. La méthode est illustrée en dimension 1 en Figure B.1.

Algorithme 7 Algorithme de Newton : recherche de la racine de $f(x) = 0$

Entrées: Valeur initiale x^0 , tolérance $\epsilon > 0$
  $k \leftarrow 0$
  Répéter
    Résoudre le système $\Delta x^k = -\nabla_x^{-1} f(x^k)$
    $x^{k+1} = x^k + \Delta x^k$
    $k \leftarrow k + 1$
  Jusqu'à $\|\Delta x^k\| \leq \epsilon$

B.2 Matrices

Définition (Matrice définie (semi-)positive) Une matrice A est définie semi-positive si $x^T Ax \geq 0, \quad \forall x$. Elle est définie positive si l'inégalité est stricte. On notera respectivement $A \succeq 0$ et $A \succ 0$.

La notation $\|A\|_N^2$ est utilisée pour décrire la norme N de A : $A^T NA$, avec N une matrice définie positive.

Lemme B.1. *(Complément de Schur) Soit X une matrice symétrique de la forme*

$$X = \begin{bmatrix} A & B \\ B^T & C \end{bmatrix}$$

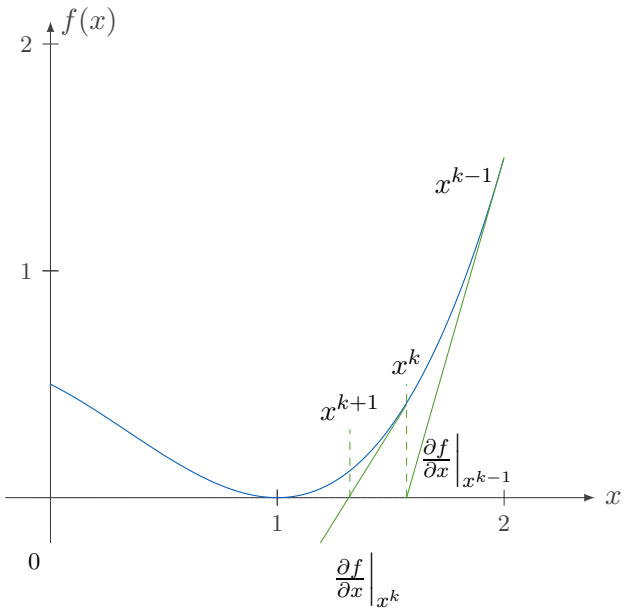


FIGURE B.1 – Méthode de Newton

Si A est non singulière, $S := C - B^T A^{-1} B$ est appelé le complément de Schur de A dans X et l'on a les équivalences suivantes :

$$X \succeq 0 \Leftrightarrow A \succ 0, S \succeq 0 \tag{B.1a}$$
$$X \preceq 0 \Leftrightarrow A \prec 0, S \preceq 0 \tag{B.1b}$$

Définition (Congruence) Deux matrices A et B de même dimension sont congruentes si pour une certaine matrice U non singulière on a

$$B = U^T A U$$

Lemme B.2. *Si A et B sont deux matrices congruentes, alors*

$$A \succeq 0 \Leftrightarrow B \succeq 0 \tag{B.2}$$

Lemme B.3. *La résolution simultanée de n inégalités matricielles $M_n \succeq 0$ revient à résoudre l'inégalité matricielle*

$$diag(M_0, M_1, \ldots, M_n) \succeq 0$$

Définition (Enveloppe convexe de matrices) L'ensemble de matrices sommets $\mathcal{M} = \{\mathbf{M}^m, m \in \overline{1..n_M}\}$ permet de définir l'enveloppe convexe notée $co(\mathcal{M})$ des matrices $\bar{M}$ de la façon suivante :

$$co(\mathcal{M}) = \{\bar{M} \text{ tel que } \bar{M} = \sum_{i=1}^{n_M} \xi_i \mathbf{M}^i, \text{ avec } \sum_{i=1}^{n_M} \xi_i = 1, \quad \xi \geq 0$$

B.3 Fonctions

Définition (Norme infinie) La norme infinie d'une fonction $f : E \subset \mathbb{R}^n \to \mathbb{R}^m$, de composantes $f_i(x)$ est définie comme

$$\|f\|_\infty = \max_{i \in \overline{1..m}} \sup_{x \in E} |f_i(x)|$$

Définition (Lipschitz) Une application $f : S \subseteq \mathbb{R}^m \to \mathbb{R}^n$ est Lipschitz si il existe une constante $\mathcal{L} \geq 0$ tel que $\forall (\mathbf{x}, \mathbf{y}) \in S^2$ on a

$$\|f(\mathbf{x}) - f(\mathbf{y})\| \leq \mathcal{L}\|\mathbf{x} - \mathbf{y}\| \tag{B.3}$$

On dira que la fonction $f(x_1, x_2, \ldots, x_m)$ est Lipschitz en x_1 si il existe une constante $\mathcal{L}_x \geq 0$ tel que $\forall (w, z) \in \mathbb{R}^2$ on a

$$\|f(w, x_2, \ldots, x_m) - f(z, x_2, \ldots, x_m)\| \leq \mathcal{L}_x\|w - z\| \tag{B.4}$$

On retiendra dans cette thèse la norme infinie.

Lemme B.4. *(Cas particulier du lemme de Grönwall) [56] Soient $\psi(t)$ et $y(t)$ deux fonctions continues à valeur positive sur un intervalle $[t_0, t_f]$. Si pour une constante $L \geq 0$ on a pour presque tout $t \in [t_0, t_f]$*

$$y(t) \leq \psi(t) + L \int_{t_0}^{t} y(\tau) d\tau$$

Alors pour presque tout $t \in [t_0, t_f]$:

$$y(t) \leq \psi(t) + L \int_{t_0}^{t} e^{L(t-\tau)} \psi(\tau) d\tau$$

Si de plus $\psi(.)$ est mesurable et bornée (apparient à $L^\infty([t_0; t_f], \mathbb{R})$), alors presque partout sur $[t_0, t_f]$ on a :

$$y(t) \leq \|\psi(.)\|_\infty^t e^{L(t-t_0)}$$

Définition Une fonction $f : \mathbb{R}^+ \to \mathbb{R}^+$ est de classe $\mathcal{K}$ si elle est continue, strictement croissante et $f(0) = 0$

Définition Une fonction $f : \mathbb{R}^+ \to \mathbb{R}^+$ est de classe $\mathcal{K}^\infty$ si elle est de classe $\mathcal{K}$ est n'est pas bornée

Définition Une fonction $g : \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R}^+$ est de classe $\mathcal{KL}$ si $p \to g(p, t)$ est de classe $\mathcal{K}$ pour tout t et $t \to g(p, t)$ est décroissante pour tout p

ANNEXE C

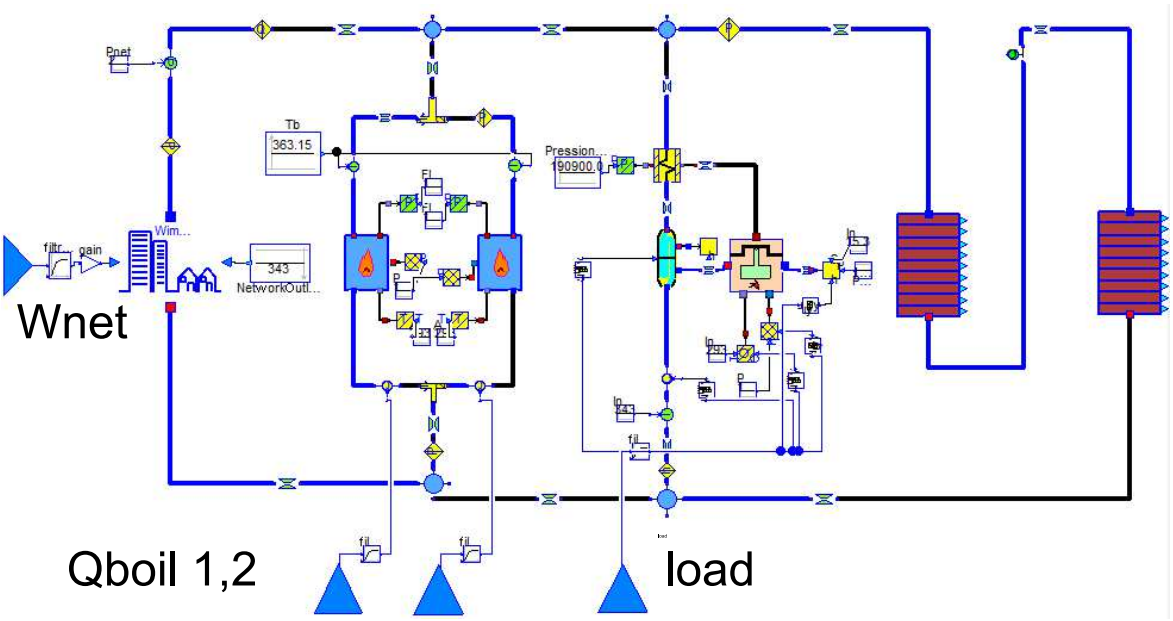
Implémentation logicielle dans JModelica

C.1 Implémentation logicielle

On présente ici les développements logiciels correspondants au Chapitre 3. Ces développements ont été réalisés dans la plateforme open-source JModelica dans le langage Python. JModelica propose un formalisme pour la modélisation des problèmes d'optimisation dynamique : Optimica.

C.1.1 Présentation d'Optimica

Optimica est une extension du langage Modelica permettant de décrire un problème d'optimisation ou de commande optimale sur un modèle Modelica. Comme les problèmes d'optimisations ne font pas partie des spécifications du langage Modelica, JModelica utilise des fichiers d'extension *.mop* pour décrire les modèles et problèmes d'optimisation (au lieu de *.mo* pour les modèles Modelica). La Figure C.1 montre comment s'effectue la description d'un problème d'optimisation.
Le modèle est tout d'abord décrit en langage Modelica. Les entrées qui seront considérées pour l'optimisation sont les variables introduites par le mot clé **input**. Puis le problème d'optimisation est décrit dans une classe **optimization**. L'utilisation du mot clé **extends** signifie que le problème d'optimisation hérite de toutes les équations du modèle. La fonction objectif est décrite entre parenthèses. Ici, la valeur de la variable *Cout* à la fin de l'horizon (**finalTime**) est considérée. On a pour cela introduit dans le modèle une nouvelle variable différentielle nommée *Cout*. Optimica permet également de décrire directement ce type d'objectif sans ajouter de nouvelle variable en utilisant **objectiveIntegrand** au lieu d'**objective** dans la définition de la classe. Des contraintes peuvent aussi être ajoutées dans le problème d'optimisation dans une section nommée **constraint**. Les contraintes qui y sont définies sont appelées *path constraints*. Il est également possible de décrire des contraintes de bornes pour les variables à l'aide des mots clés Modelica **min** et **max**. Il est d'ailleurs hautement recommandé d'utiliser ces attributs pour les contraintes de bornes, même si celles-ci peuvent être décrites dans la section **constraint**. La raison est qu'IPOPT traite différemment les contraintes de




```
optimization Tracking_ModeleSimple(objective=Cout(finalTime),finalTime=10)
    extends ModeleSimple;

    Real Cout;
    input Real x_ref; // Entrée imposée par l'extérieur--> traitement spécial
    initial equation
        Cout=0;
    equation
        der (Cout)=(x-x_ref)^2+0.1*u^2;
    constraint
        x^2<=5;
end Tracking_ModeleSimple;
```

FIGURE C.1 – Spécification d'un problème d'optimisation avec JModelica

bornes et les autres contraintes, et sera d'autant plus efficace que les bornes sont spécifiées (IPOPT étant un algorithme de points intérieurs, l'information des bornes permet de mieux cerner le domaine admissible pour une variable).

C.1.2 Architecture de JModelica

La partie de JModelica touchant à l'optimisation est développée en langage Python. JModelica fonctionne par défaut avec Python 2.7. L'installateur de JModelica propose d'ailleurs également d'installer Python 2.7 ainsi que les librairies utiles. JModelica utilise plusieurs logiciels tiers pour l'optimisation que l'on présente dans la Figure C.2 :

- CasADi est un logiciel de différentiation automatique [24]. La différentiation automatique permet de calculer de façon symbolique des informations sur les dérivées d'une fonction (dérivées directionnelles, gradients, Hessiens...) en un point donné, ce qui garantit à la fois une grande précision numérique et des temps de calcul très réduits. CasADi

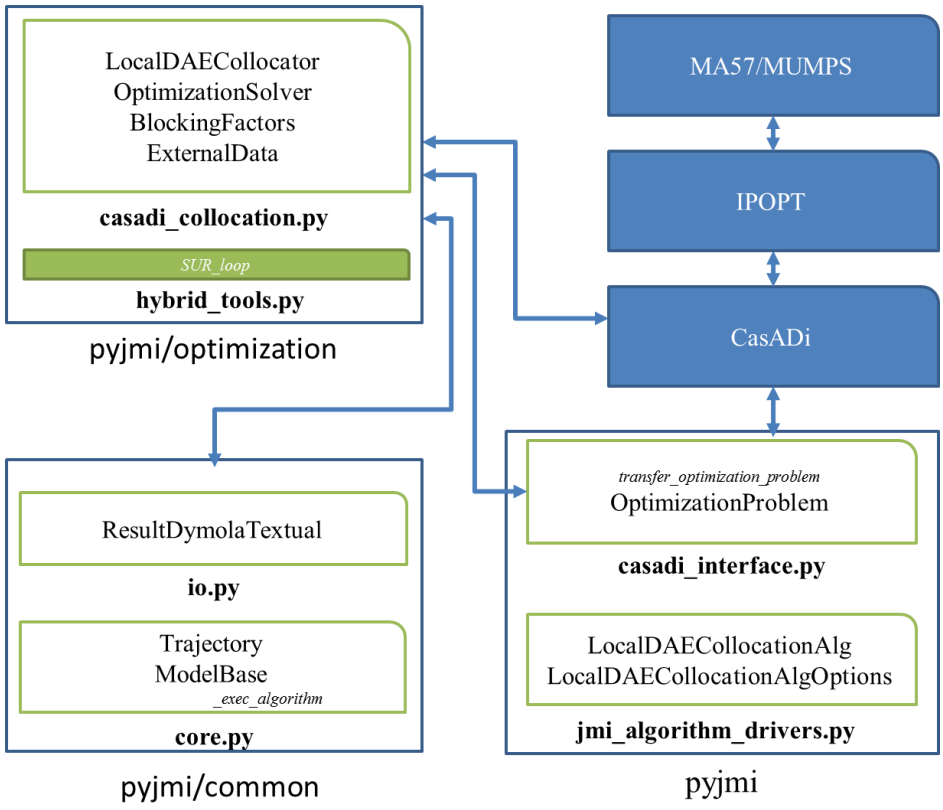


FIGURE C.2 – Fichiers utilisés pour l'optimisation dans JModelica

implémente également l'interface avec un certain nombre de solveurs d'optimisation. Nous utilisons par la suite IPOPT.

- IPOPT [45] est un solveur d'optimisation non linéaire utilisant un algorithme de points intérieurs. IPOPT est un standard en optimisation non linéaire, particulièrement adapté aux problèmes de très grande taille, ce qui est souvent le cas des NLP résultant de la méthode de collocation.
- Des solveurs linéaires. Le solveur venant avec le logiciel par défaut est MUMPS. Cependant, des gains substantiels en puissance et temps de calcul sont possibles avec d'autres solveurs tels que MA57 qui a été utilisé ici.

Les résultats obtenus sont stockés dans une instance de la classe *ResultDymolaTextual* fournie dans le fichier io.py. et sont également écrits sous formes de fichiers textes dans le même format que Dymola.

C.1.3 Modifications apportées au code de collocation

On présente ici les modifications apportées au code de collocation (casadi_collocation.py). Ces modifications concernent la classe *LocalDAECollocator* dans laquelle la représentation symbolique du problème de commande optimale est transformée en un problème NLP, et la classe *BlockingFactors* qui est utilisée pour décrire les entrées constantes par morceaux. Les deux autres classes importantes dans ce fichier sont *OptimizationSolver*, une classe utilisée pour extraire les informations du solveur non linéaire (IPOPT dans notre cas) et d'autres

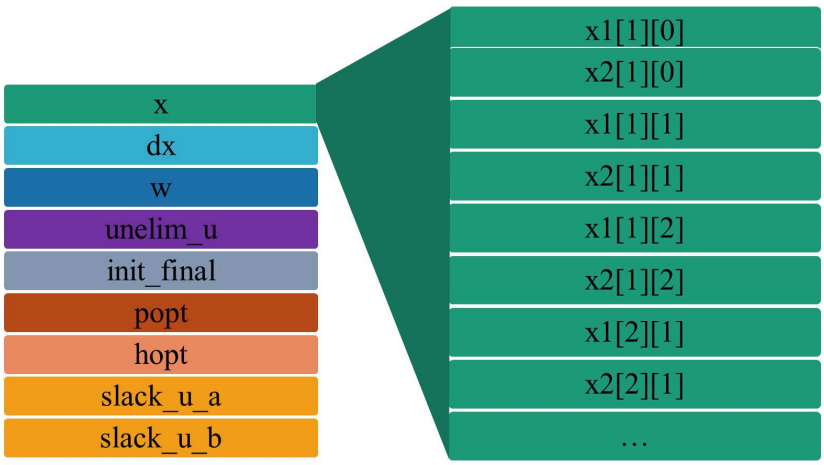


FIGURE C.3 – Structure des variables du problème NLP dans la méthode de collocation de JModelica

données pour l'analyse des résultats. Enfin la classe *ExternalData* permet quant à elle de fournir à l'algorithme des trajectoires d'entrées externes, comme par exemple les trajectoires des perturbations (courbe de demande dans notre cas), où encore des trajectoires mesurées pour renseigner un algorithme d'estimation d'état.

Définition des variables du NLP :

Les variables d'optimisation résultant de la méthode de collocation sont empilées dans un grand vecteur selon un ordre prédéfini. On présente ici l'aspect de ce vecteur dans la Figure C.3. On utilise ici les notations présentes dans le code Python. Les variables correspondent à plusieurs informations :

- Un **type** : état x, dérivée dx, variable algébrique w, entrée non éliminée $unelim\_u$, valeur initiale ou finale $init\_final$, paramètre pouvant être optimisé p_opt, longeur des éléments h_opt. On y ajoute les deux nouveaux types $slack\_u\_a$ et $slack\_u\_b$ correspondant aux variables sa et sb des équations (3.54)et (3.57) (les variables up ne sont pas ajoutées au problème mais calculées comme des fonctions de sa et sb.
- Un indice **var_ind** pour chacune des variables appartenant à *type*.
- Un élément **k** auquel elle est associée (si nécessaire)
- Un point de collocation **c** auquel elle est associée (si nécessaire)

Le vecteur est ainsi constitué de sous-vecteurs pour chaque *type*. Au sein de ces vecteurs, la progression temporelle est respectée : on parcourt d'abord les variables, puis les points de collocation, puis les éléments. De nombreuses structures de données permettent par ailleurs de retracer l'évolution d'une variable, ou des valeurs des variables en un point donné.

Définition des entrées :

La définition des coûts de démarrage (ou par extension des coûts à la hausse pour une entrée qui ne serait pas une entrée binaire) n'a de sens que si l'entrée en question est constante par morceaux.

Dans l'implémentation informatique de JModelica, de telles entrées peuvent être définies à l'aide de facteurs bloquants (*blocking factors*). Les facteurs bloquants pour une entrée u se présentent sous la forme d'un vecteur $\mathbf{b}_u$ dont les termes représentent le nombres de pas de

temps sur lesquels l'entrée u sera maintenue constante (bloquée). La somme des termes est ainsi égale au nombre d'éléments de discrétisation n_e utilisé dans la méthode de collocation. A titre d'exemples, pour $n_e = 5$, on pourra utiliser $\mathbf{b}_u = [1, 1, 1, 1, 1]$ ou encore $\mathbf{b}_u = [1, 2, 2]$.
Les facteurs bloquants pour un problème d'optimisation sont définis à l'aide de la classe *BlockingFactors* dans le fichier *casadi_collocation.py*. Cette classe définit les vecteurs $\mathbf{b}_u$ utilisés pour chacune des entrées bloquées.
Comme les entrées pour lesquelles sont définis des coûts de démarrage doivent être des entrées bloquées, la classe *BlockingFactors* a été modifiée de la façon suivante :

- Paramètre optionnel c_{up} pour une entrée u définissant le coût d'une unité à la hausse (équivalent au coût de démarrage si l'entrée ω est la variable qui définit l'état du composant)
- Paramètre optionnel c_d définissant le coût d'une unité à la baisse
- Paramètre optionnel *penCompl* définissant le coefficient de pénalisation ρ de la non-complémentarité dans (3.57).

Modification des contraintes et de la fonction objectif :
Deux types de contraintes ont été ajoutées dans la classe *LocalDAECollocator* : des contraintes de bornes sur les variables $sa \geq 0$, $sb \geq 0$, ainsi que les égalités reliant sa_k, sb_k, ω_k, ω_{k-1} décrites dans les équations (3.54-3.55). Ces équations sont en réalité adaptées pour prendre en compte les facteurs bloquants, qui font que les commutations ne sont possibles qu'en certains éléments de la grille de collocation.
La fonction objectif est enfin modifiée en y ajoutant les coûts de démarrage et la pénalisation de la non-complémentarité des variables *sa* et *sb* comme décrit en (3.51) et (3.57).

C.1.4 Présentation du code hybrid_tools

Tous les outils relatifs à la procédure Sum Up Rounding et au raffinement de la grille de collocation ont été réunis dans un script Python nommé *hybrid_tools.py*, placé dans le dossier **pyjmi/optimization** où se trouve également la méthode de collocation. Il contient les méthodes suivantes

- *SUR_loop* : il s'agit de la méthode principale. Elle prend en entrée un modèle d'optimisation ainsi que des options de configuration, et retourne les résultats des différents problèmes d'optimisation (relaxé, à entrées logiques fixées)
- *SUR_minT* : cette méthode implémente l'Algorithme 6 (SURminT) et est appelée par *SUR_loop*.
- *simu_opt* : cette méthode simule le modèle pour une trajectoire des variables de commande provenant de l'optimisation. Elle permet de comparer la trajectoire prévue par la méthode de collocation avec la trajectoire obtenue par simulation (plus précise).
- *collocation_error_romberg* : Cette méthode calcule l'erreur locale sur la grille de collocation d'après la méthode décrite en 3.4.3.1.
- *refine_grid* : cette méthode prend en entrée les trajectoires des variables binaires relaxées sur la grille courante et l'erreur locale sur cette grille. Elle fournit la grille raffinée pour la prochaine itération ainsi que les trajectoires des variables logiques arrondies par la méthode SUR décrite dans l'Algorithme 5.
- *with_fixed_bin* : cette méthode réalise l'optimisation du problème pour une trajectoire fixée des entrées binaires
- *plot_SUR_result* : cette méthode permet l'affichage de variables sélectionnées par l'uti-

lisateur. Elle prend en entrée une structure de donnée définissant les différentes figures à créer et la disposition des graphiques dans ces figures.

ANNEXE D

Quantification de l'erreur du modèle (TanLd) en l'absence de perturbations

On cherche ici à exprimer l'erreur du modèle interne (TanLd) utilisé dans le MPC par rapport au système réel (4.1). On a vu dans la partie 4.1.3 que plusieurs étapes sont nécessaires pour arriver au modèle interne (TanLd) : le système physique (4.1) est modélisé à l'aide du formalisme DAE (NLc), puis ce modèle non-linéaire est linéarisé (TanLc) et enfin discrétisé pour aboutir à (TanLd) (on note que l'utilisation du modèle augmenté consiste juste en un changement de variable et ne modifie pas (TanLd)).

On commence par introduire l'hypothèse suivante afin de quantifier l'erreur entre le système réel et son modèle non linéaire (NLc).

Hypothèse H15. *L'erreur de modélisation non linéaire est bornée*

$$\|f_p(x,u,w) - f(x,u,w)\| \leq e_{mod}$$

On rappelle ensuite que le modèle non linéaire en temps continu peut se décomposer en une partie linéaire (le modèle linéarisé tangent) et une nonlinéarité :
$f(x(t),u(t),w(t)) = A(t)\delta x(t) + B_u(t)\delta u(t) + B_w(t)\delta w(t) + g(\delta x(t),\delta u(t),\delta w(t)$

Hypothèse H16. *$g(\delta x,\delta u,\delta w)$ est Lipschitz en $(\delta x,\delta u,\delta w)$ de constantes L_x^g,L_u^g,L_w^g et satisfait $g(0,0,0)=0$. On en déduit que f est également Lipschitz en x,u,w de constantes L_x^f, L_u^f, L_w^f.*

Les modèles linéarisés variants dans le temps (LTV) utilisés dans le MPC proviennent de linéarisations du modèle non linéaire autour d'une trajectoire nominale décrite en (NLnom), puis d'une discrétisation par un schéma d'Euler avant des modèles linéarisés. On cherche ici à borner l'erreur entre l'état du modèle LTV discret utilisé dans le MPC et celui du modèle non linéaire, au bout d'un pas d'échantillonnage et en l'absence de perturbations. Cette erreur est définie par

$$\epsilon_{LTV} = x(x_{nom,k} + \delta x_k, u_{r,k} + \delta u_k; t_{k+1}) - \bar{x}_{k+1} \tag{D.1}$$

Avec $\bar{x}_{k+1} = x(x_{nom,k}, u_{r,k}; t_{k+1})$. On suppose que les commandes sont constantes sur chaque pas de temps. Le modèle exact pour l'écart au nominal est donné par

$$\dot{\delta x} = \delta f(\delta x, \delta u) = f(x_{nom} + \delta x, u_r + \delta u, w_r) - f(x_{nom}, u_r, w_r) \tag{D.2}$$

Théorème D.1. *L'erreur du modèle linéarisé tangent ϵ_{LTV} sur un pas avec l'hypothèse Zero Order Hold pour les entrées δu et en l'absence de perturbations ($\delta w = 0$) est en $O(T_s^2)$.*

Démonstration. Par la méthode d'Euler,on a :

$$\delta x_{k+1} = \bar{F}_k(\delta x, \delta u) = \left(I + T_s \left.\frac{\partial f}{\partial x}\right|_{(x_{nom,k}, u_{r,k})} \right) \delta x + \left(T_s \left.\frac{\partial f}{\partial u}\right|_{(x_{nom,k}, u_{r,k})} \right) \delta u$$

Sans perte de généralité, on pose $t_k = 0$ et on note $\delta x_{[j]}$ la $j-$ème composante de δx. $\bar{F}_k$ étant une fonction linéaire, on peut appliquer le théorème de la moyenne qui assure qu'il existe $s \in [0, T_s]$ tel que :

$$\delta x_{[j]}(T_s) = \delta x_{[j]}(0) + T_s \left.\frac{d\delta x_{[j]}(t)}{dt}\right|_{t=0} + \frac{s^2}{2} \left.\frac{d^2 \delta x_{[j]}(t)}{dt^2}\right|_{t=s}$$

L'erreur de troncature est donnée comme

$$\epsilon_{LTV} = \delta x_{[j]}(T_s) - \left(\delta x_{[j]}(0) + T_s \left.\frac{d\delta x_{[j]}(t)}{dt}\right|_{t=0} \right)$$

Cette erreur est bornée par le maximum de $\frac{s^2}{2} \left.\frac{d^2 \delta x_{[j]}(t)}{dt^2}\right|_{t=s}$, que l'on va chercher à déterminer. L'erreur de troncature ϵ_{LTV} à $(\delta x, \delta u)$ fixé vérifie ainsi sur un pas de temps, composante par composante :

$$\begin{aligned} \|\epsilon_{LTV,[j]}(\delta x, \delta u)\| &\leq \frac{T_s^2}{2} \max_{s \in [0,T_s]} \left| \left.\frac{d^2 \delta x_{[j]}(t)}{dt^2}\right|_{t=s} \right| \\ &= \frac{T^2}{2} \max_{s \in [0,T_s]} \left| \frac{d\dot{\delta x}_{[j]}(t)}{d\delta x} \frac{d\delta x_{[j]}(t)}{dt} + \frac{d\dot{\delta x}_{[j]}(t)}{d\delta u} \frac{d\delta u(t)}{dt} \right|_{t=s} \Bigg| \end{aligned}$$

On s'intéresse ici à l'erreur sur un pas de temps. Comme les entrées sont constantes par morceaux, sur un pas de temps, on a $\frac{d\delta u(t)}{dt} = 0$. On passe ensuite en norme infinie en considérant la valeur la plus grande parmi les composantes j du vecteur δx.

$$\|\epsilon_{LTV}(\delta x, \delta u)\|_\infty \leq \frac{T_s^2}{2} \left\| \frac{d\delta f(\delta x, \delta u, 0)}{d\delta x} \right\|_\infty \left\| \dot{\delta x}(\delta x_0, \delta_u; t) \right\|_\infty$$

Tout d'abord, une borne entre l'état initial et l'état courant peut être calculée :

$$\begin{aligned} \|x(s) - x(0)\| &\leq s \max_{\tau \in [0,s]} \|f(x(\tau), \bar{u}, 0)\| \\ &\leq s \max_{\tau} \left(\|f(x(0), \bar{u}, 0)\| + \|f(x(\tau), \bar{u}, 0) - f(x(0), \bar{u}, 0)\| \right) \\ &\leq s\|f(x(0), \bar{u}, 0)\| + sL_x^f \|x(s) - x(0)\| \\ \|x(s) - x(0)\| &\leq \frac{s}{1 - sL_x^f} \|f(x(0), \bar{u}, 0)\| \end{aligned}$$

Par ailleurs la dérivée exacte de l'état $\dot{\delta x}$ du modèle non linéaire peut se réécrire comme :

$$\begin{aligned}\dot{\delta x}(\delta x_0, \delta_u; t) =& f(x(t), u_r + \delta u, w_r) - f(x_{nom}(t), u_r, w_r)\\ =& f(x(t), u_r + \delta u, w_r) - f(x(0), u_r + \delta u, w_r) + f(x(0), u_r + \delta u, w_r)\\ & - f(x(0), u_r, w_r) + f(x(0), u_r, w_r) - f(x_{nom}(0), u_r, w_r)\\ & + f(x_{nom}(0), u_r, w_r) - f(x_{nom}(t), u_r, w_r)\end{aligned}$$

On a alors pour $T_s L_x^f < 1$ (qui est de toutes façon la condition de stabilité requise pour utiliser la discrétisation par la méthode d'Euler explicite) :

$$\begin{aligned}\|\dot{\delta x}(\delta x_0, \delta_u; T_s)\|_\infty \leq& L_x^f \|x(T_s) - x(0)\|_\infty + L_u^f \|\delta u\|_\infty + L_x^f \|\delta x_0\|_\infty + L_x^f \|x_{nom}(T_s) - x_{nom}(0)\|_\infty\\ \leq& \frac{T_s}{1 - L_x^f T_s} \left(\|f(x_{nom}(0), u_r, w_r)\|_\infty + \|f(x(0), u_r, w_r)\|_\infty\right) \dots\\ & \cdots + L_u^f \|\delta u\|_\infty + L_x^f \|\delta x_0\|_\infty\end{aligned}$$

$$\begin{aligned}\delta f(\delta x1, \delta u, 0) - \delta f(\delta x_2, \delta u, 0) &= A(t)(\delta x_1 - \delta x_2) + g(\delta x_1, \delta u, 0) - g(\delta x_2, \delta u, 0)\\ \|\delta f(\delta x_1, \delta u) - \delta f(\delta x_2, \delta u)\| &\leq (\sigma(A(t)) + L_x^g)\|\delta x_1 - \delta x_2\|\end{aligned}$$

En posant $\sigma(A_i)$ la plus grande valeur propre de $A(t)$ à l'instant t_i, on en déduit une constante de Lipschitz pour la fonction δf : $L_{\delta x} = \sigma(A_i) + L_x^g$

$$\begin{aligned}\left\| \frac{d\delta f(\delta x, \delta u, 0)}{d\delta x} \right\|_\infty &= \left\| \lim_{h \to 0} \frac{\delta f(\delta x + h, \delta u, 0) - \delta f(\delta x, \delta u, 0)}{h} \right\|_\infty\\ &\leq \left\| \lim_{h \to 0} \frac{L_{\delta x} h}{h} \right\|_\infty\\ &= L_{\delta x}\end{aligned}$$

En généralisant à un instant t_k quelconque, l'erreur sur δx due à la linéarisation puis discrétisation du modèle non linéaire est alors bornée par :

$$\begin{aligned}\|\epsilon_{LTV}(\delta x_k, \delta u_k)\|_\infty \leq \frac{L_{\delta x} T_s^2}{2} \Bigg(& \frac{T_s}{1 - L_x^f T_s} \left(\|f(x_{nom}(t_k), u_{r,k}, w_{r,k})\|_\infty + \|f(x(t_k), u_{r,k}, w_{r,k})\|_\infty\right)\\ & + L_u^f \|\delta u_k\|_\infty + L_x^f \|\delta x_k\|_\infty \Bigg) \quad \text{(D.3)}\end{aligned}$$

L'erreur de linéarisation est ainsi en $O(T_s^2)$. ∎

Si l'on fixe le pas d'échantillonnage T_s, en rappelant que $\|f(x, u, w)\| \leq M$, on peut mettre l'erreur de modélisation sous la forme suivante :

$$\|\epsilon_{LTV}(\delta x_k, \delta u_k)\|_\infty \leq c_1 + c_2 \|\delta u_k\|_\infty + c_3 \|\delta x_k\|_\infty$$

Avec $c_1 = \frac{M L_{\delta x} T_s^3}{1 - L_x^f T_s}$, $c_2 = \frac{L_{\delta x} L_x u^f T_s^2}{2(1 - L_x^f T_s)}$ et $c_3 = \frac{L_{\delta x} L_x^f T_s^2}{2(1 - L_x^f T_s)}$.

ANNEXE E

Expression des sorties du second modèle augmenté

Le second modèle augmenté (4.41) a pour état augmenté

$$x_{a,k} = \begin{pmatrix} \Delta\delta x_k \\ \delta u_k \end{pmatrix}$$

Le modèle d'état est le suivant :

$$\begin{aligned} \overbrace{\begin{pmatrix} \delta x_{k+1} \\ \delta u_k \end{pmatrix}}^{x_{a,k+1}} &= \overbrace{\begin{bmatrix} A_k & B_k \\ 0 & I \end{bmatrix}}^{\Psi_k} \begin{pmatrix} \delta x_k \\ \delta u_{k-1} \end{pmatrix} + \overbrace{\begin{bmatrix} B_{u,k} \\ I \end{bmatrix}}^{\Gamma_k} \Delta\delta u_k + \overbrace{\begin{bmatrix} B_{w,k} \\ 0 \end{bmatrix}}^{G_k} \delta w_k \\ \delta y_k &= \underbrace{\begin{bmatrix} c_k & 0 \end{bmatrix}}_{C_k} \begin{pmatrix} \delta x_k \\ \delta u_{k-1} \end{pmatrix} \end{aligned} \tag{E.1}$$

On cherche ici à dériver une expression pour la sortie au rang $i+k$. Sans perte de généralité, on suppose $i = 0$ pour alléger la notation et l'on donne un exemple de ce calcul pour le cas $k = 3$. On généralise ensuite cette expression pour $i + k$.

$$\begin{aligned} \delta y_3 &= C_3 x_{a,3} \\ &= C_3 \left[\Psi_2 x_{a,2} + \Gamma_2 \Delta\delta u_2 + G_2 \delta w_2\right] \\ &= C_3 \left[\Psi_2 \left(\Psi_1 \left(\Psi_0 x_{a,0} + \Gamma_0 \Delta\delta u_0 + G_0 \delta w_0\right) + \Gamma_1 \Delta\delta u_1 + G_1 \delta w_1\right) + \Gamma_2 \Delta\delta u_2 + G_2 \delta w_2\right] \\ &= C_3 \left[\Psi_2 \Psi_1 \Psi_0 x_{a,0} + \Psi_2 \Psi_1 \left(\Gamma_0 \Delta\delta u_0 + G_0 \delta w_0\right) + \Psi_2 \left(\Gamma_1 \Delta\delta u_1 + G_1 \delta w_1\right)\right. \\ &\quad \left. + \Gamma_2 \Delta\delta u_2 + G_2 \delta w_2\right] \end{aligned}$$

La formule qui généralise cette expression est :

$$\delta y_{i+k} = C_k \left[\left(\prod_{n=1}^{k} \Psi_{i+k-n}\right) x_{a,i} + \sum_{n=1}^{k} \left[\left(\prod_{m=1}^{k-n} \Psi_{i+k-m}\right) \left(\Gamma_{i+n-1} \Delta\delta u_{i+n-1} + G_{i+n-1} \Delta\delta w_{i+n-1}\right)\right]\right] \tag{E.2}$$

ANNEXE F

Test de la méthode de collocation hybride sur un cas complexe

Dans cette partie, on présente l'application de la méthode de collocation hybride sur un autre cas industriel qui a fait l'objet de l'étude [9] : il s'agit d'un réseau de chaleur alimenté par une cogénération à cycle vapeur. La chaleur peut également être produite par deux unités auxiliaires, dont une peut être activée ou inactivée. Un accumulateur d'eau chaude est par ailleurs présent dans le réseau. Le modèle de cogénération développé par la société Modelon est montré en Figure F.1 et le système complet (cogénération + réseau de chaleur + stockage + unités auxiliaires) en Figure F.2. La complexité du modèle réside au niveau de sa taille et de la modélisation des propriétés physiques de l'eau vapeur.

La Figure F.3 montre les résultats obtenus sur l'installation à la deuxième itération de l'algorithme. Les résultats de l'optimisation relaxée sont en vert et ceux de l'optimisation à variables logiques fixées sont en bleu. Les courbes en tirets correspondent aux trajectoires prédites par l'optimisation et les courbes en trait plein le résultat de simulation pour les trajectoires de commande optimales. Les traits verticaux matérialisent la grille temporelle à l'itération courante (trait plein) et pour la prochaine grille d'optimisation (trait tiret pointillé). Le graphique du haut correspond à la charge de l'unité pouvant être activée (de 12% à 100%) ou inactivée. Cet aspect hybride est traité par l'ajout d'une commande binaire ω. On constate que l'optimisation active cette unité de 3 h à 22 h environ. Le deuxième graphique montre le débit au réseau, le troisième graphique la pression au condenseur et le dernier graphique la pression à la turbine. L'optimisation respecte les contraintes sur le débit maximal dans la pompe et la pression minimale au condenseur. Dans le cas de la pression au condenseur, on observe cependant que le résultat de simulation et d'optimisation sont encore assez éloignés sur certains éléments, et qu'un raffinement supplémentaire de la grille d'optimisation est nécessaire pour prendre en compte la dynamique rapide de l'installation sur ces pas de temps. La valeur maximale de l'erreur relative sur la grille courante atteint en effet 38.4 %, ce qui est au delà de la tolérance fixée.
La Figure F.4 montre quant à elle dans l'ordre : la puissance thermique fournie par la cogénération, celle soutirée par les clients, la température en sortie de la cogénération et la température reçue par les clients. (On n'a pas représenté ici la puissance soutirée du stockage ni des unités auxiliaires). On constate que l'installation respecte la température minimale

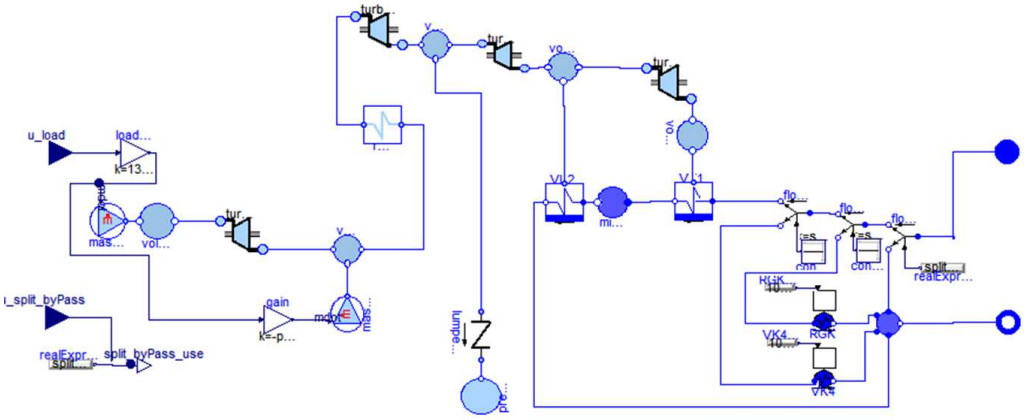


FIGURE F.1 – Cogénération à cycle vapeur

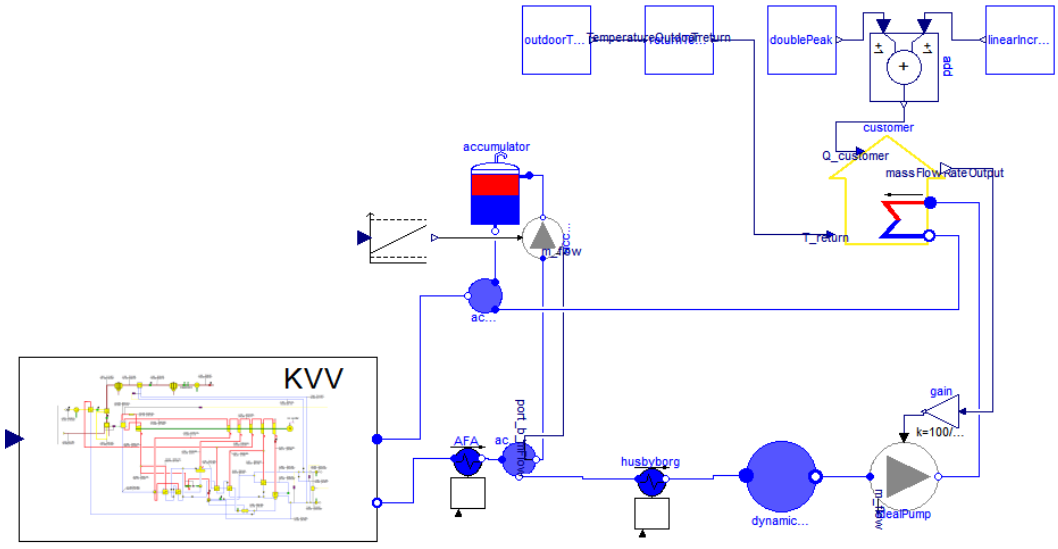


FIGURE F.2 – Système énergétique complet

injectée au réseau et celle fournie aux clients.

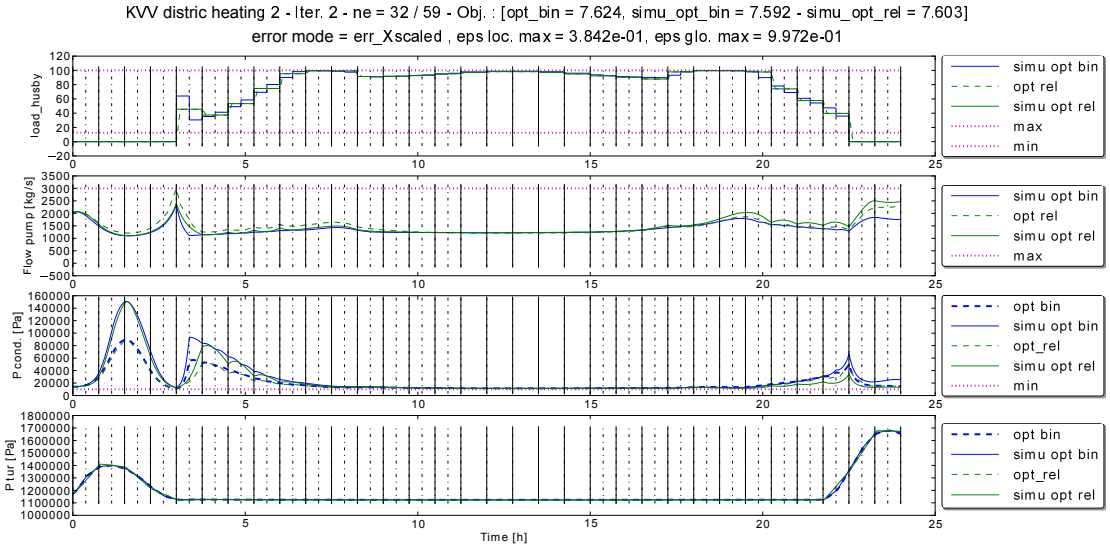


FIGURE F.3 – Optimisation d'une cogénération vapeur 1

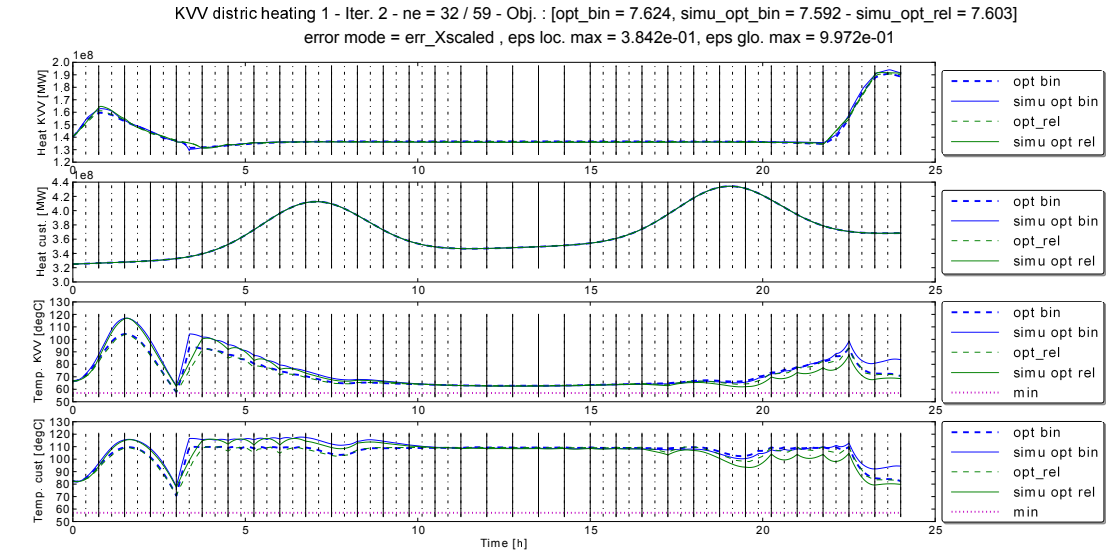


FIGURE F.4 – Optimisation d'une cogénération vapeur 2

Publications

Le travail présenté dans cette thèse a fait l'objet des publications suivantes

Conférences avec comité de lecture :

- **Fouquet, M.**, Guéguen, H., Dumur, D., Faille, D., 2015. Tracking of optimal trajectories for power plants based on physical models. IFAC-PapersOnLine 48, 373 – 378. doi :http ://dx.doi.org/10.1016/j.ifacol.2015.12.407
- Faille, D., **Fouquet, M.**, Péchiné, B., Ahallal, S., Ghaoui, L.E., Pauphilet, J., 2015. Robust planning for Combined Heat and Power production. IFAC-PapersOnLine 48, 391 – 396. doi :http ://dx.doi.org/10.1016/j.ifacol.2015.12.410
- Liu, S.J., Faille, D., **Fouquet, M.**, El-Hefni, B., Wang, Y., Zhang, J.B., Wang, Z.F., Chen, G.F., Soler, R., 2015. Dynamic Simulation of a 1MWe CSP Tower Plant with Two-level Thermal Storage Implemented with Control System. Energy Procedia 69, 1335 – 1343. doi :http ://dx.doi.org/10.1016/j.egypro.2015.03.139
- **Fouquet, M.**, Gueguen, H., Faille, D., Dumur, D., 2014. Hybrid dynamic optimization of power plants using sum-up rounding and adaptive mesh refinement. IEEE, pp. 316–321. doi :10.1109/CCA.2014.6981365
- Deneux, O., Péchiné, B., **Fouquet, M.**, 2013. Optimization of Design and Operation of a Combined Heat and Power Plant, by Use of a 1D-Physical Model. ASME, p. V002T08A006. doi :10.1115/POWER2013-98265

Article de revue (soumis à Journal of Process Control) :

- **Fouquet, M.**, Magnusson, F., Guéguen, H., Velut, S., Faille, D., Dumur, D., Henningsson, T., Hybrid dynamic optimization of power plants based on physical models and the collocation method

Titre : Commande prédictive non linéaire : Application à la production d'énergie

Cette thèse porte sur l'optimisation et la commande prédictive des centrales de production d'énergie en utilisant des modèles physiques des installations. Les modèles sont réalisés à l'aide du langage Modelica, un langage équationnel adapté à la modélisation de systèmes multi-physiques. La modélisation de systèmes physiques dans ce langage est présentée dans une première partie, ainsi que les traitements symboliques réalisés par les compilateurs Modelica pour mettre les modèles sous une forme adaptée à l'optimisation. On présente dans une seconde partie le développement d'une méthode d'optimisation dynamique hybride pour les centrales de production d'énergie, qui fournit une trajectoire optimisée de l'installation sur un horizon long. Les trajectoires calculées incluent les trajectoires des commandes continues ainsi que les décisions d'engagement des différents équipements. L'algorithme d'optimisation combine la méthode de collocation et une méthode nommée Sum Up Rounding (SUR) pour la prise en compte des décisions d'engagement. Un algorithme de commande prédictive (MPC) est enfin introduit afin de garantir le suivi des trajectoires optimales et de prendre en compte en temps réel la présence de perturbations et les erreurs du modèle d'optimisation. L'algorithme MPC utilise des modèles linéarisés tangents générés automatiquement à partir du modèle non linéaire.

Mots clés : Modèles physiques, Modelica, Optimisation dynamique hybride, Méthode de collocation, Sum Up Rounding, Commande prédictive non linéaire

---

Titre : Nonlinear Model Predictive Control : Application to power plants

This thesis deals with hybrid optimal control and Model Predictive Control (MPC) of power plants by use of physical models. Models of the facilities are developped with Modelica, an equation based language tailored for modelling multi-physics systems. Modeling of physical systems with Modelica is introduced in a first part, as well as some of the symbolic processing done by Modelica compilers that transform the original model to a form suited for optimization. Then, a method to solve optimal control problems on hybrid systems (such as power plants) is presented. This methods provides an optimal trajectory for the power plant on a long horizon. The optimal trajectory computed by the method includes the trajectories of continuous inputs as well as switching decisions for components in the plant. The optimization algorithm combines the collocation method and a method named Sum Up Rounding (SUR) for dealing with switches. Finally, a Model Predictive Controller is developped in order to follow this optimal trajectory in real time, and to cope with disturbances on the actual system and modelling errors. The proposed MPC uses tangent linear models of the plant that are derived automatically from the nonlinear model.

Keywords : Physical models, Modelica, Hybrid dynamic optimization, Collocation method, Sum Up Rounding, Nonlinear Model Predictive Control