La commande prédictive des systèmes hybrides

Sylvain Leirens, Jean Buisson

Chapitre 4

Commande prédictive des systèmes hybrides

4.1. Introduction

Le terme *systèmes hybrides* se réfère à des systèmes dont les modèles font intervenir simultanément des phénomènes continus et discrets et pour lesquelles des approches purement continues ou discrètes ne sont pas satisfaisantes. Si certains de ces systèmes ont fait l'objet d'études déjà anciennes (c'est le cas par exemple des systèmes à relais), ce n'est que récemment qu'une communauté rassemblant des automaticiens et des informaticiens s'est intéressée à développer des formalismes de modélisation, des outils d'analyse et de synthèse généraux [ZAY 01]. Les principales conférences consacrées à cette thématique sont "*Hybrid Systems : Computation and Control* (HSCC)" et "*Analysis and Design of Hybrid Systems* (ADHS)".

La difficulté majeure pour l'étude des systèmes hybrides provient de l'impossibilité de faire coexister mathématiquement, dans une théorie unifiée simple, des variables continues et discrètes (y compris pour ce qui concerne le temps). La résolution des problèmes passe en général par des approches algorithmiques. La commande prédictive est ainsi tout particulièrement indiquée pour résoudre le problème de commande optimale pour ce type de système.

Ce chapitre présente tout d'abord les formalismes PWA (*PieceWise Affine*) et MLD (*Mixed Logical Dynamical*) en section 4.2, en mettant l'accent sur leurs possibilités et limitations. La mise en œuvre de la commande prédictive avec ces deux formalismes est ensuite détaillée en section 4.3, notamment par l'étude des techniques de résolution du problème d'optimisation associé.

Chapitre rédigé par Sylvain LEIRENS et Jean BUISSON.

4.2. Formalismes

4.2.1. *Systèmes PWA*

4.2.1.1. *Description*

Un système PWA est décrit par un ensemble de modèles affines associé à un partitionnement de l'espace d'état-commande. Classiquement, ce découpage résulte en un nombre fini de régions polyédrales disjointes. Une partition est alors définie par un ensemble d'inégalités linéaires qui constituent des contraintes additionnelles du modèle d'état affine.

En temps discret, un système PWA prend la forme suivante :

$$\mathbf{x}(k+1) = \mathbf{A}_i\mathbf{x}(k) + \mathbf{B}_i\mathbf{u}(k) + \mathbf{a}_i \tag{4.1a}$$

$$\mathbf{y}(k) = \mathbf{C}_i\mathbf{x}(k) + \mathbf{D}_i\mathbf{u}(k) + \mathbf{c}_i \tag{4.1b}$$

où l'indice i représente le mode $i(k) \in \mathbb{I} \subset \mathbb{N}$ à l'instant k défini par l'appartenance à la région χ_i de l'espace d'état-commande :

$$(\mathbf{x}(k), \mathbf{u}(k)) \in \chi_i = \left\{(\mathbf{x}, \mathbf{u}) \,|\, \mathbf{F}_i\mathbf{x} + \mathbf{G}_i\mathbf{u} \leq \mathbf{f}_i\right\} \tag{4.2}$$

$\mathbf{x} \in \mathbb{X} \subset \mathbb{R}^{n_x}$, $\mathbf{u} \in \mathbb{U} \subset \mathbb{R}^{n_u}$ et $\mathbf{y} \in \mathbb{Y} \subset \mathbb{R}^{n_y}$ sont respectivement l'état, l'entrée et la sortie du système. Les inégalités définissant les régions de l'espace d'état-commande sont non strictes, ce qui signifie que les frontières sont communes. En toute rigueur, il faudrait utiliser aussi des inégalités strictes de telle sorte que le découpage constitue un partitionnement, *i.e.* des régions χ de l'espace d'état-commande telles que $\chi_i \cap \chi_j = \emptyset \; \forall i, j \in \mathbb{I}$ et $\bigcup_i \chi_i = \mathbb{X} \times \mathbb{U}$. Pour prendre en compte des discontinuités de l'état aux commutations (changement de mode), il faut ajouter aux équations ci-dessus des fonctions de saut (*reset maps*) pour réinitialiser l'état du système.

EXEMPLE.– Un système PWA peut résulter de l'approximation d'un système non-linéaire par un ensemble de modèle affines. L'exemple de la figure 4.1 est une cuve remplie de liquide et possédant une fuite. Soit S_f la section du tuyau de fuite, h la hauteur de liquide et h_{max} la hauteur maximale de liquide dans la cuve. Le débit de fuite q est donné par l'équation de Torricelli :

$$q = S_f\sqrt{2\,g\,h} \tag{4.3}$$

avec l'accélération de la pesanteur g. Cette relation est non-linéaire (douce ou *smooth*) à cause de la présence de la fonction racine carré.

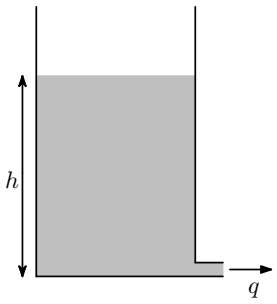


Figure 4.1. *Exemple : débit de fuite d'une cuve remplie de liquide*

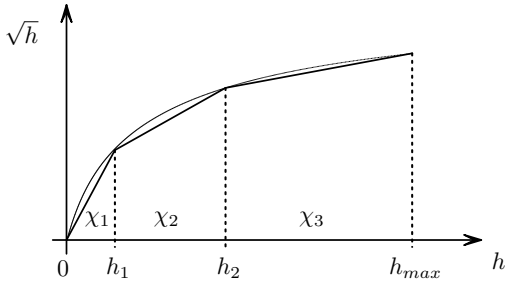


Figure 4.2. *Exemple : partitionnement de l'espace en trois régions*

L'espace engendré par la hauteur de liquide h est partitionné en trois régions comme illustré par la figure 4.2. Ainsi, $\sqrt{h}$ est approchée par :

$$\sqrt{h} = \begin{cases} \frac{h}{\sqrt{h_1}} & \text{si} \quad 0 \leq h \leq h_1 \\ \frac{\sqrt{h_2}-\sqrt{h_1}}{h_2-h_1}(h-h_1)+\sqrt{h1} & \text{si} \quad h_1 \leq h \leq h_2 \\ \frac{\sqrt{h_{max}}-\sqrt{h_2}}{h_{max}-h_2}(h-h_2)+\sqrt{h2} & \text{si} \quad h_2 \leq h \leq h_{max} \end{cases} \tag{4.4}$$

4.2.1.2. *Simulation des systèmes PWA*

L'algorithme suivant permet de simuler un système modélisé sous forme PWA :

1) Choisir l'état initial $\mathbf{x}(k)$;

2) Choisir la commande $\mathbf{u}(k)$;

3) Le couple $(\mathbf{x}(k), \mathbf{u}(k))$ (satisfaction des contraintes liées à la partition considérée) permet de connaître le mode $i(k)$;

4) Déterminer $\mathbf{x}(k+1)$ avec (4.1a) ;

5) Aller à 2 en prenant $k = k+1$.

Il est donc très simple de simuler un système PWA, puisque cela ne requiert aucun calcul autre que l'évolution de la trajectoire de l'état $\mathbf{x}$ et de la sortie $\mathbf{y}$.

REMARQUE.– Les systèmes PWA présentés ici ne peuvent être l'objet que de commutations autonomes (franchissement d'une frontière séparant deux régions). Il est en effet impossible de modéliser des commutations contrôlées avec cette formulation, dues par exemple à la présence d'un interrupteur ou d'une vanne tout-ou-rien (TOR) dans le système. Après la section suivante consacrée au formalisme MLD, nous reviendrons sur ce point avec une extension du formalisme PWA (*cf.* section 4.2.3).

4.2.2. *Systèmes MLD*

4.2.2.1. *Description*

La figure 4.3 est une représentation assez générale d'un système dynamique hybride qui consiste en l'interaction d'un sous-système (à état) discret et d'un sous-système(à état) continu interconnectés par l'intermédiaire d'interfaces continu/discret (C/D) et discret/continu (D/C). On retrouve ce type de structure dans beaucoup de systèmes où une partie commande (numérique) interagit avec une partie opérative (analogique) par l'intermédiaire d'interfaces analogique/numérique et numérique/analogique.

Le formalisme MLD repose sur la transformation de relations d'implication ($\rightarrow$) et d'équivalence ($\leftrightarrow$) en inégalités mixtes. Des relations de ces types interviennent aux interfaces et leur transformation en inégalités va permettre de les formaliser pour les inclure dans un modèle mathématique global. Le terme mixte que l'on retrouvera souvent dans la suite désigne la présence de variables à valeurs continues (réelles) et de variables à valeurs discrètes (logiques/binaires) dans les équations. Pour la partie dynamique continue, si les équations sont linéaires, les inégalités mixtes seront aussi linéaires. Dans les développements actuels du formalisme MLD, la dynamique continue est décrite par des équations linéaires à temps discret (équations aux différences/de récurrence). Le formalisme MLD repose donc sur les idées–clés suivantes :

– représenter le couplage entre les dynamiques discrètes et continues par des inégalités linéaires mixtes ;

– représenter les relations logiques sous forme d'inégalités linéaires binaires ;

– inclure des variables binaires dans des équations aux différences.

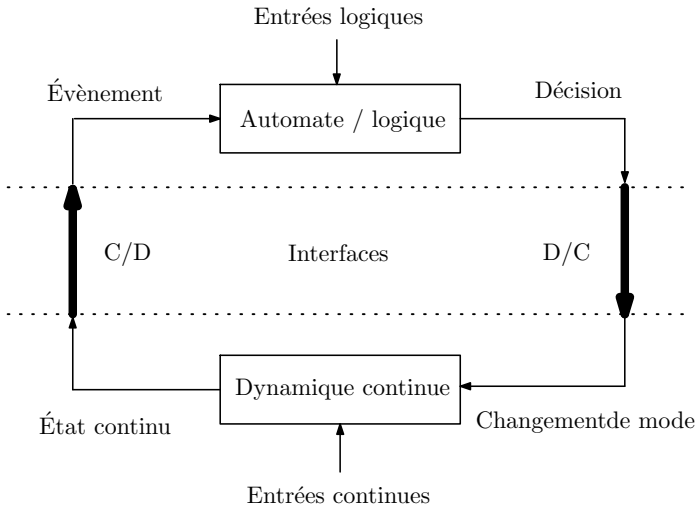


Figure 4.3. *Interaction entre des sous-systèmes discret et continu*

La **partie discrète** du modèle permet de décrire des dynamiques typiquement représentées par un automate d'état fini. Les relations logiques décrivant son fonctionnement peuvent être transformées en inégalités linéaires binaires via un ensemble de transformations élémentaires dont on donne un aperçu ci-dessous. Soient trois propositions logiques X_1, X_2 et X_3 et les trois variables binaires δ_1, δ_2 et δ_3 associées. La proposition logique $X_1 \rightarrow X_2$ qui s'écrit $[\delta_1 = 1] \rightarrow [\delta_2 = 1]$ se transforme en :

$$\delta_1 - \delta_2 \leq 0 \tag{4.5}$$

La proposition logique $(X_1 \wedge X_2) \leftrightarrow X_3$ qui s'écrit $[\delta_1 = 1] \wedge [\delta_2 = 1] \leftrightarrow [\delta_3 = 1]$ se transforme en :

$$\left\{ \begin{array}{lcl} -\delta_1 + \delta_3 & \leq & 0 \\ -\delta_2 + \delta_3 & \leq & 0 \\ \delta_1 + \delta_2 - \delta_3 & \leq & 1 \end{array} \right. \tag{4.6}$$

L'ensemble des transformations se trouve rassemblé dans [MIG 02]. En pratique, les inégalités linéaires sont obtenues en ayant recours au calcul d'enveloppe convexe pour lequel des algorithmes efficaces existent ou au calcul de formes normales conjonctives (CNF pour *Conjonctive Normal Form*). L'exemple suivant illustre ce propos.

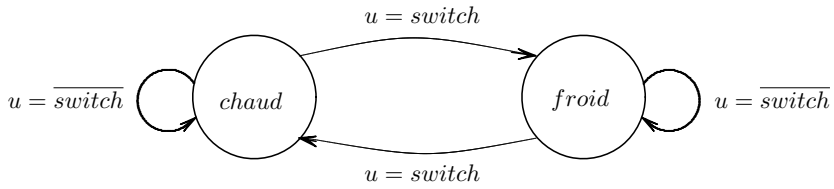


Figure 4.4. *Exemple : automate d'états fini*

EXEMPLE.– Considérons le petit automate de la figure 4.4 modélisant un système de chauffage.

Il est composé de deux états ($chaud$ et $froid$) et d'un interrupteur poussoir. Le comportement du système est très simple : il reste dans l'état courant tant que l'interrupteur n'est pas actionné. Le système change d'état quand on appuie sur l'interrupteur. Pour décrire ce comportement, il faut donc considérer une variable d'état logique $x(t) \in \{chaud,\ froid\}$ et une entrée de commande logique $u(t) \in \left\{\overline{switch},\ switch\right\}$. La dynamique de ce système est la suivante :

- si $x(t) = chaud$ et $u(t) = switch$ alors $x(t+1) = froid$
- si $x(t) = chaud$ et $u(t) = \overline{switch}$ alors $x(t+1) = chaud$
- si $x(t) = froid$ et $u(t) = switch$ alors $x(t+1) = chaud$
- si $x(t) = froid$ et $u(t) = \overline{switch}$ alors $x(t+1) = froid$

En introduisant les variables binaires $\delta_i \in \{0, 1\}$ suivantes :

$$\left\{\begin{array}{lcl} \delta_1 = 1 & \leftrightarrow & x(t) = chaud \\ \delta_2 = 1 & \leftrightarrow & u(t) = switch \\ \delta_3 = 1 & \leftrightarrow & x(t+1) = chaud \end{array}\right. \tag{4.7}$$

le comportement du système s'écrit :

$$\left\{\begin{array}{lcl} [\delta_1 = 1] \wedge [\delta_2 = 1] & \rightarrow & [\delta_3 = 0] \\ [\delta_1 = 1] \wedge [\delta_2 = 0] & \rightarrow & [\delta_3 = 1] \\ [\delta_1 = 0] \wedge [\delta_2 = 1] & \rightarrow & [\delta_3 = 1] \\ [\delta_1 = 0] \wedge [\delta_2 = 0] & \rightarrow & [\delta_3 = 0] \end{array}\right. \tag{4.8}$$

Les quatre inégalités ci-dessous forment dans $[0, 1]^3$ l'enveloppe convexe du polytope défini par les sommets $(1, 1, 0)$, $(1, 0, 1)$, $(0, 1, 1)$ et $(0, 0, 0)$:

$$\left\{\begin{array}{rcl} -\delta_1 - \delta_2 + \delta_3 & \leq & 0 \\ -\delta_1 + \delta_2 - \delta_3 & \leq & 0 \\ \delta_1 - \delta_2 - \delta_3 & \leq & 0 \\ \delta_1 + \delta_2 + \delta_3 & \leq & 0 \end{array}\right. \tag{4.9}$$

Les **interfaces** permettent de représenter les interactions entre la partie discrète et la partie continue du modèle. Prenons quelques exemples : lorsqu'une grandeur continue atteint un certain seuil, un *évènement* d'alarme est généré (passage continu vers discret) ; l'arrêt d'urgence a été demandé, les vannes de vidanges doivent s'ouvrir (changement de mode : passage discret vers continu).

À l'interface continu vers discret (C/D), l'introduction de variables auxiliaires binaires δ est nécessaire pour représenter les évènements (ex. : franchissement d'un seuil). La relation

$$[f(x) \leq 0] \leftrightarrow [\delta = 1] \tag{4.10}$$

est convertie en deux inégalités mixtes :

$$\left\{ \begin{array}{lcl} f(x) & \leq & M(1-\delta) \\ f(x) & \geq & \epsilon + (m-\epsilon)\delta \end{array} \right. \tag{4.11}$$

$\epsilon > 0$ représente une tolérance permettant d'obtenir des inégalités non-strictes, ce qui est nécessaire pour les utiliser comme contraintes lors de mise en forme de problèmes d'optimisation (*cf.* sections 4.2.2.3 et 4.3.2). f est une fonction de $x \in \mathbb{R}$ telle que $f(x) \in [m,\, M]$ où m et M sont respectivement un minorant et un majorant de f.

De même, à l'interface discret vers continu (D/C), des variables auxiliaires réelles z seront introduites pour prendre en compte des changements de mode (ex. : vanne ouverte). Le produit mixte

$$z = \delta\, f(x) \tag{4.12}$$

est convertie en quatre inégalités mixtes :

$$\left\{ \begin{array}{lcl} z & \geq & m\delta \\ z & \leq & M\delta \\ z & \leq & f(x) - m(1-\delta) \\ z & \geq & f(x) - M(1-\delta) \end{array} \right. \tag{4.13}$$

REMARQUE.– Dans ce qui précède, nous avons vu que la dynamique discrète pouvait être convertie en inégalités linéaires binaires. À l'interface, l'introduction de variables auxiliaires permet de décrire par des inégalités mixtes l'influence de l'état continu sur la dynamique discrète (évènements) et l'influence de l'état discret sur la dynamique continue (changement de mode). On retrouvera donc les variables auxiliaires binaires δ dans les équations de la dynamique discrète et les variables auxiliaires réelles z dans les équations de la dynamique continue. Les bases du formalisme MLD sont posées.

4.2.2.2. *Mise en équations*

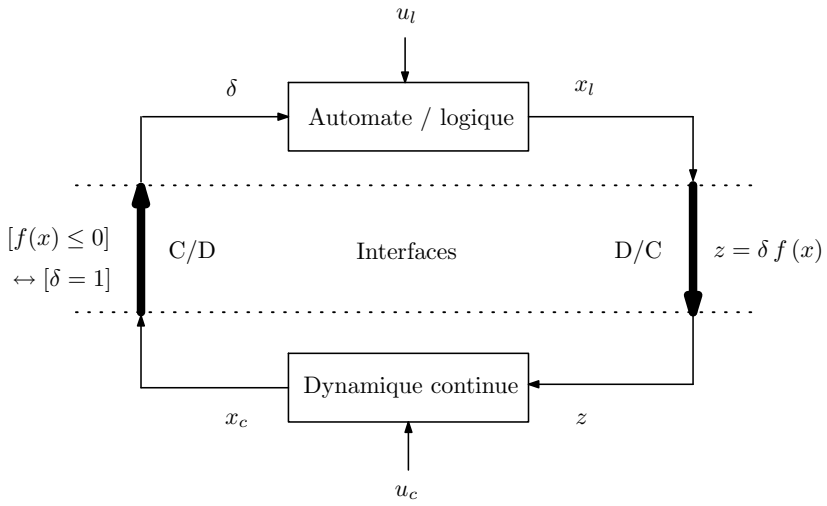


Figure 4.5. *Modèle MLD*

Si on considère des dynamiques continues linéaires à temps discret, l'utilisation des transformations précédentes permet de mettre en équations le système modélisé par la figure 4.5 sous forme d'un ensemble d'équations et de contraintes linéaires mixtes :

$$\mathbf{x}(k+1) = \mathbf{A}_1\mathbf{x}(k) + \mathbf{B}_1\mathbf{u}(k) + \mathbf{B}_2\boldsymbol{\delta}(k) + \mathbf{B}_3\mathbf{z}(k) \quad (4.14a)$$

$$\mathbf{y}(k) = \mathbf{C}_1\mathbf{x}(k) + \mathbf{D}_1\mathbf{u}(k) + \mathbf{D}_2\boldsymbol{\delta}(k) + \mathbf{D}_3\boldsymbol{z}(k) \quad (4.14b)$$

$$\mathbf{E}_2\boldsymbol{\delta}(k) + \mathbf{E}_3\mathbf{z}(k) \leq \mathbf{E}_1\mathbf{u}(k) + \mathbf{E}_4\mathbf{x}(k) + \mathbf{E}_5 \quad (4.14c)$$

avec l'état

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_c \\ \mathbf{x}_l \end{pmatrix} \text{ avec } \mathbf{x}_c \in \mathbb{R}^{n_{xc}},\ \mathbf{x}_l \in \{0,1\}^{n_{xl}},\ n_x \triangleq n_{xc} + n_{xl} \quad (4.15)$$

l'entrée de commande

$$\mathbf{u} = \begin{pmatrix} \mathbf{u}_c \\ \mathbf{u}_l \end{pmatrix} \text{ avec } \mathbf{u}_c \in \mathbb{R}^{n_{uc}},\ \mathbf{u}_l \in \{0,1\}^{n_{ul}},\ n_u \triangleq n_{uc} + n_{ul} \quad (4.16)$$

et la sortie

$$\mathbf{y} = \begin{pmatrix} \mathbf{y}_c \\ \mathbf{y}_l \end{pmatrix} \text{ avec } \mathbf{y}_c \in \mathbb{R}^{n_{yc}},\ \mathbf{y}_l \in \{0,1\}^{n_{yl}},\ n_y \triangleq n_{yc} + n_{yl} \tag{4.17}$$

Les variables auxiliaires $\boldsymbol{\delta} \in \{0,1\}^{r_l}$ et $\mathbf{z} \in \mathbb{R}^{r_c}$ ($r \triangleq r_c + r_l$) proviennent de la transformation des relations de la partie discrète et des interfaces en inégalités mixtes linéaires.

REMARQUE.– Si on prête attention à la structure des équations (4.14a)–(4.14c), on peut noter plusieurs choses :

– dans chaque mode les équations dynamiques sont linéaires mais le changement de mode est un aspect non-linéaire *caché* dans les inégalités mixtes (matrices $\mathbf{E}_{1,\ldots,5}$) ;

– la partie de la dynamique continue commune à tous les modes se trouve dans les matrices $\mathbf{A}_1$, $\mathbf{B}_1$, $\mathbf{C}_1$ et $\mathbf{D}_1$[1] ;

– la dynamique continue relative aux différents modes se trouve ainsi prise en compte par les variables auxiliaires réelles $\mathbf{z}$ et les matrices $\mathbf{B}_3$ et $\mathbf{D}_3$;

– la dynamique discrète est prise en compte par les variables discrètes, les variables auxiliaires binaires $\boldsymbol{\delta}$ et par l'intermédiaire des matrices $\mathbf{B}_2$ et $\mathbf{D}_2$ ainsi que les inégalités mixtes.

4.2.2.3. *Modèle MLD bien posé et simulation*

Un modèle MLD est dit *bien posé* si pour tout couple $(\mathbf{x}(k), \mathbf{u}(k))$, les variables $\mathbf{x}(k+1)$ et $\mathbf{y}(k)$ existent de façon unique, permettant d'obtenir une trajectoire unique de l'état $\mathbf{x}$ et de la sortie $\mathbf{y}$ pour un état initial $\mathbf{x}(k)$ et une séquence de commandes $(\mathbf{u}(k), \cdots, \mathbf{u}(k+N))$. Un modèle MLD est dit *complètement bien posé* si le couple $(\boldsymbol{\delta}(k), \mathbf{z}(k))$ est également unique.

Pour tester l'existence du couple $(\boldsymbol{\delta}(k), \mathbf{z}(k))$, une façon de procéder est de formuler un problème d'optimisation mixte linéaire dit de type MILP (*Mixed Integer Linear Program*). Étant donné le couple $(\mathbf{x}(k), \mathbf{u}(k))$, il s'agit de résoudre

$$\min_{(\boldsymbol{\delta}(k),\, \mathbf{z}(k))} \sum_{i=1}^{r_l} \delta_i(k) + \sum_{i=1}^{r_c} z_i(k) \tag{4.18}$$

1. Toutefois, ce n'est pas requis par le formalisme. Si tous les modes partagent une partie de la dynamique continue, elle peut évidemment être prise en compte par les variables auxiliaires réelles $\mathbf{z}$, *cf.* remarque suivante.

sous les contraintes (4.14c) pour déterminer le couple $(\boldsymbol{\delta}(k), \mathbf{z}(k))$ solution. Pour tester l'unicité, le lecteur est invité à se reporter à [BEM 99] où est présenté un algorithme basé sur un test de faisabilité.

L'algorithme suivant permet de simuler un système modélisé sous forme MLD :

1) Choisir l'état initial $\mathbf{x}(k)$;

2) Choisir la commande $\mathbf{u}(k)$;

3) Pour le couple $(\mathbf{x}(k), \mathbf{u}(k))$, obtenir $\boldsymbol{\delta}(k)$ et $\mathbf{z}(k)$ en effectuant un test de faisabilité sur les contraintes (4.14c) ;

4) Déterminer $\mathbf{x}(k+1)$ avec (4.14a) ;

5) Aller à 2 en prenant $k = k + 1$.

Les contraintes sont indissociables des équations dans la formulation du modèle. Les relations aux interfaces ont été transformées en inégalités linéaires mixtes. Pour simuler un système MLD, il est donc nécessaire de résoudre à chaque instant k un problème d'optimisation de type MILP pour déterminer le couple $(\boldsymbol{\delta}(k), \mathbf{z}(k))$ nécessaire pour obtenir l'état suivant $\mathbf{x}(k+1)$ et la sortie $\mathbf{y}(k)$ à l'aide des équations (4.14a) et (4.14b).

4.2.2.4. *HYSDEL*

HYSDEL est l'acronyme de HYbdrid Systems DEscription Language [TOR 02]. C'est un outil de description de modèle qui permet d'obtenir la forme MLD (4.14a)–(4.14c) de façon systématique. La transformation *à la main* des relations aux interfaces en inégalités linéaires mixtes s'avère très fastidieuse. À partir des équations dynamiques, des équations logiques et des relations aux interfaces exprimées dans un langage proche du C, HYSDEL génère un script Matlab contenant les matrices du modèle MLD et des informations supplémentaires telles que les dimensions de ces matrices et des paramètres de contrôle.

L'appel du programme HYSDEL se fait depuis Matlab avec en paramètre le fichier source (fichier *.hys*). HYSDEL renvoie un fichier *.m* contenant les matrices du modèle

sous la forme suivante :

$$\begin{pmatrix}\mathbf{x}_c(k+1)\\ \mathbf{x}_l(k+1)\end{pmatrix} = \begin{pmatrix}\mathbf{A}_{cc} & \mathbf{A}_{cl}\\ \mathbf{A}_{lc} & \mathbf{A}_{ll}\end{pmatrix}\begin{pmatrix}\mathbf{x}_c(k)\\ \mathbf{x}_l(k)\end{pmatrix} + \begin{pmatrix}\mathbf{B}_{1cc} & \mathbf{B}_{1cl}\\ \mathbf{B}_{1lc} & \mathbf{B}_{1ll}\end{pmatrix}\begin{pmatrix}\mathbf{u}_c(k)\\ \mathbf{u}_l(k)\end{pmatrix} + \begin{pmatrix}\mathbf{B}_{2cl}\\ \mathbf{B}_{2ll}\end{pmatrix}\boldsymbol{\delta}(k) + \begin{pmatrix}\mathbf{B}_{3cc}\\ \mathbf{B}_{4lc}\end{pmatrix}\mathbf{z}(k) \quad (4.19a)$$

$$\begin{pmatrix}\mathbf{y}_c(k)\\ \mathbf{y}_l(k)\end{pmatrix} = \begin{pmatrix}\mathbf{C}_{cc} & \mathbf{C}_{cl}\\ \mathbf{C}_{lc} & \mathbf{C}_{ll}\end{pmatrix}\begin{pmatrix}\mathbf{x}_c(k)\\ \mathbf{x}_l(k)\end{pmatrix} + \begin{pmatrix}\mathbf{D}_{1cc} & \mathbf{D}_{1cl}\\ \mathbf{D}_{1lc} & \mathbf{D}_{1ll}\end{pmatrix}\begin{pmatrix}\mathbf{u}_c(k)\\ \mathbf{u}_l(k)\end{pmatrix} + \begin{pmatrix}\mathbf{D}_{2cl}\\ \mathbf{D}_{2ll}\end{pmatrix}\boldsymbol{\delta}(k) + \begin{pmatrix}\mathbf{D}_{3cc}\\ \mathbf{D}_{4lc}\end{pmatrix}\mathbf{z}(k) \quad (4.19b)$$

$$\mathbf{E}_2\boldsymbol{\delta}(k) + \mathbf{E}_3\mathbf{z}(k) \leq \mathbf{E}_1\begin{pmatrix}\mathbf{u}_c(k)\\ \mathbf{u}_l(k)\end{pmatrix} + \mathbf{E}_4\begin{pmatrix}\mathbf{x}_c(k)\\ \mathbf{x}_l(k)\end{pmatrix} + \mathbf{E}_5 \quad (4.19c)$$

Un fichier HYSDEL est divisé en deux parties. La première partie, appelée *INTERFACE*, a pour but de déclarer les variables (états, entrées, sorties) ainsi que les constantes. Elle comprend donc les sections *STATE*, *INPUT*, *OUTPUT* et *PARAMETER*. La deuxième partie, appelée *IMPLEMENTATION* est composée de huit sections *OUTPUT, AD, LOGIC, DA, LINEAR, CONTINUOUS, AUTOMATA, MUST* décrivant les relations entre les variables du modèle :

1) Section *OUTPUT* : elle permet de spécifier des relations linéaires et logiques pour la sortie, *i.e.* du type : $y_c = x_c + bu_c$, $y_l = x_l \vee u_l$;

2) Section *AD* : elle définit les variables auxiliaires binaires δ issues de des relations du type : si $ax_c \leq b$ alors $\delta = 1$;

3) Section *LOGIC* : cette section permet de spécifier des fonctions arbitraires sur les variables binaires (sélection de mode par ex.), *i.e.* des relations du type $\delta_2 = \delta_1 \vee u_l$;

4) Section *DA* : elle définit les variables auxiliaires continues z issues de conditions du type si-alors-sinon sur les variables binaires, *i.e.* elle définit les modes. On trouvera dans cette section des relations du type : si $\delta = 1$ alors $z = a_1x_c + b_1$ sinon $z = a_2x_c + b_2$;

5) Section *LINEAR* : elle permet de définir des variables auxiliaires qui sont des expressions affines des variables continues, *i.e.* de relations du type $z = ax_c + b$;

6) Section *CONTINUOUS* : elle décrit la dynamique continue, sous la forme d'équations aux différences, *i.e.* de relations du type $x_c^{+2} = ax_c + bu_c + cz$;

7) Section *AUTOMATA* : cette section décrit la dynamique discrète sous la forme d'équations logiques du type $x_l^+ = x_l \wedge u_l$;

2. L'exposant $+$ symbolise l'instant suivant.

8) Section *MUST* : cette section permet de spécifier des contraintes additionnelles (telles que des bornes sur la commande) sur les variables du modèle sous la forme d'inégalités linéaires et de relations logiques, par ex. $x_c^{min} \leq x_c \leq x_c^{max}$, $\delta_1 \wedge \delta_2$.

Un modèle MLD n'est pas unique en ce qui concerne le nombre de variables auxiliaires. Pour un même système, il y a autant de modèles MLD que de façons d'introduire les variables auxiliaires. Il est alors souhaitable de rechercher le modèle qui minimise le nombre de variables binaires afin de minimiser la complexité combinatoire. HYSDEL comporte un algorithme dédié à cette recherche. Pour en savoir plus sur les possibilités offertes par cet outil, le lecteur est invité à consulter le manuel d'utilisation d'HYSDEL [TOR 02]. Actuellement, l'écriture matricielle n'est pas prise en charge lors de la spécification du modèle dans HYSDEL, ce qui limite son utilisation à des systèmes de faible dimension. Les futurs développements prévoient d'inclure cette fonctionnalité.

4.2.3. *Équivalence PWA–MLD et extension du formalisme PWA*

4.2.3.1. *Équivalence entre les formalismes PWA et MLD*

Dans [HEE 01], l'équivalence entre les formalismes PWA et MLD a été montrée en supposant la présence de bornes sur les états et les entrées (PWA vers MLD).

Le passage de MLD vers PWA suppose que le modèle MLD soit (complètement) bien posé et se base sur l'idée suivante [BEM 00] : partitionner l'espace d'état-commande engendré par $\mathbf{x}_c \in \mathbb{R}^{n_{xc}}$ et $\mathbf{u}_c \in \mathbb{R}^{n_{uc}}$ en groupant en régions χ_i tous les $(\mathbf{x}_c^T \ \mathbf{u}_c^T)^T$ correspondant à une même valeur du vecteur $\boldsymbol{\delta}$. Pour simplifier, mais sans perte de généralité, on peut supposer que les variables binaires $\mathbf{x}_l$ et $\mathbf{u}_l$ sont aussi des variables auxiliaires (binaires). Avec $\boldsymbol{\delta}$ fixé, l'inégalité (4.14c) définit alors un polyèdre dans $\mathbb{R}^{n_{x_c}+n_{u_c}+r_c}$ et le modèle MLD étant bien posé, $\mathbf{z}$ est défini de façon unique, *i.e.* $\mathbf{z}$ peut s'écrire comme une fonction affine de $\mathbf{x}_c$ et $\mathbf{u}_c$.

4.2.3.2. *Extension du formalisme PWA*

Une évolution du formalisme PWA consiste à considérer la présence d'entrées continues et discrètes, *i.e.* l'entrée $\mathbf{u}$ est de nature mixte. Pour clarifier l'écriture, on note $\mathbf{u}_c \in \mathbb{U}_c \subset \mathbb{R}^{n_{uc}}$ le vecteur d'entrées continues et $\mathbf{u}_d \in \mathbb{U}_d$ le vecteur d'entrées discrètes avec $n_u = n_{uc} + n_{ud}$. Les équations du modèle (4.1a)–(4.2) sont alors modifiées comme suit :

$$\mathbf{x}(k+1) = \mathbf{A}_i\mathbf{x}(k) + \mathbf{B}_i\mathbf{u}_c(k) + \mathbf{a}_i \quad (4.20a)$$

$$\mathbf{y}(k) = \mathbf{C}_i\mathbf{x}(k) + \mathbf{D}_i\mathbf{u}_c(k) + \mathbf{c}_i \quad (4.20b)$$

où $\mathbf{x} \in \mathbb{X} \subset \mathbb{R}^{n_x}$ est le vecteur d'état, $\mathbf{u}_c \in \mathbb{U} \subset \mathbb{R}^{n_{uc}}$ est le vecteur des entrées continues et $\mathbf{y} \in \mathbb{Y} \subset \mathbb{R}^{n_y}$ est le vecteur de sortie du système. L'indice i dénote le mode $i(k) \in \mathbb{I} \subset \mathbb{N}$ à l'instant k qui est défini par une valeur du vecteur des entrées discrètes $\mathbf{u}_d(k)$ ($\mathbf{u}_d \in \mathbb{U}_d$ avec par ex., $\mathbb{U}_d \subset \mathbb{Z}^{n_{ud}}$) et l'appartenance à une partition χ_j ($j \in \mathbb{J} \subset \mathbb{N}$) de l'espace d'état-commande continu, indépendante des entrées discrètes $\mathbf{u}_d$[3] :

$$(\mathbf{x}(k), \mathbf{u}_c(k)) \in \chi_j = \left\{(\mathbf{x}, \mathbf{u}_c) \,|\, \mathbf{F}_j\mathbf{x} + \mathbf{G}_j\mathbf{u}_c \leq \mathbf{f}_j\right\} \tag{4.21}$$

On définit alors une fonction $\varphi : \mathbb{X} \times \mathbb{U}_c \times \mathbb{U}_d \longrightarrow \mathbb{I}$ telle que :

$$i(k) = \varphi(\mathbf{x}(k), \mathbf{u}_c(k), \mathbf{u}_d(k)) \tag{4.22}$$

L'entrée de commande $\mathbf{u}$ du système se compose des entrées continues ($\mathbf{u}_c$) et discrètes ($\mathbf{u}_d$) avec $n_u = n_{uc} + n_{ud}$. Le nombre total de modes pour le système est donné par :

$$p = \text{card}(\mathbb{I}) = \text{card}(\mathbb{J}) \times \text{card}(\mathbb{U}_d) \tag{4.23}$$

Cette formulation permet à présent de prendre en compte dans le modèle des commutations contrôlées (ou commandées).

4.3. Mise en œuvre de la commande prédictive

4.3.1. *Commande prédictive des systèmes PWA*

4.3.1.1. *Optimisation mixte*

Le principe de la commande prédictive consiste à résoudre un problème de commande optimale en boucle ouverte sur un horizon de prédiction (fini) glissant. La boucle est fermée par l'utilisation d'une nouvelle mesure de l'état du système à chaque instant d'échantillonnage. La prédiction est réalisée à l'aide d'un modèle du système et pour être le plus général, le modèle PWA étendu (4.20a)–(4.21) est utilisé dans cette section. Le modèle (PWA) de prédiction est stationnaire, on peut donc utiliser l'origine des temps comme instant courant sans perte de généralité dans les équations.

3. C'est un choix de modélisation. On peut aussi faire un choix de partitionnement dépendant de la valeur de $\mathbf{u}_d$ pour l'espace d'état-commande continu.

Soit $\mathbf{U}_N$ la séquence de commandes sur l'horizon de prédiction, choisi de longueur N :

$$\mathbf{U}_N = (\mathbf{u}^T(0)\ \mathbf{u}^T(1)\ \cdots\ \mathbf{u}^T(N-1))^T \tag{4.24}$$

avec $\mathbf{u} = (\mathbf{u}_c^T\ \mathbf{u}_d^T)^T$ et soit la fonction de coût suivante :

$$J_N(\mathbf{x}(0), \mathbf{U}_N) = F(\mathbf{x}(N)) + \sum_{k=0}^{N-1} L(\mathbf{x}(k), \mathbf{u}(k)) \tag{4.25}$$

En pratique, la fonction de coût (ou critère à optimiser) inclut un terme basé sur les écarts de la sortie $\mathbf{y}$ par rapport à une référence et un terme basé sur l'entrée de commande $\mathbf{u}$. Soient :

$$L(\mathbf{x}(k), \mathbf{u}(k)) = \|\mathbf{y}(k) - \mathbf{y}_r\|_{\mathbf{Q}_y} + \|\mathbf{u}(k)\|_{\mathbf{Q}_u} \tag{4.26a}$$

$$F(\mathbf{x}(N)) = \|\mathbf{x}(N) - \mathbf{x}_r\|_{\mathbf{Q}_f} \tag{4.26b}$$

où $\mathbf{y}_r$ et $\mathbf{x}_r$ sont respectivement les références pour la sortie et l'état (final). Les matrices de pondération sont symétriques et telles que $\mathbf{Q}_f \geq 0$, $\mathbf{Q}_y \geq 0$ et $\mathbf{Q}_u > 0$ (en prenant en compte des contraintes sur $\mathbf{u}$, typiquement des contraintes sur les actionneurs, il suffit que la matrice $\mathbf{Q}_u$ soit définie semi-positive.). On peut aussi ajouter un terme sur l'état $\|\mathbf{x}(k+1) - \mathbf{x}_r\|_{\mathbf{Q}_x}$ avec $\mathbf{Q}_x \geq 0$.

À chaque instant d'échantillonnage, le problème d'optimisation $\mathcal{P}_N$ suivant doit être résolu, l'exposant o signifiant l'optimalité :

$$\mathcal{P}_N(\mathbf{x}(0))\ :\ J_N^o(\mathbf{x}(0)) = \min_{\mathbf{U}_N} J_N(\mathbf{x}(0), \mathbf{U}_N) \tag{4.27}$$

sous les contraintes du modèle (4.20a)–(4.21). Une contrainte égalité sur l'état final $\mathbf{x}(N) = \mathbf{x}_f$ peut être ajoutée, notamment pour garantir la stabilité du système bouclé [MAY 00].

Une particularité du problème d'optimisation ainsi formulé est sa nature mixte. La présence d'entrées de commande continues ($\mathbf{u}_c$) et discrètes ($\mathbf{u}_d$) conduit à considérer la séquence de commandes recherchée $\mathbf{U}_N$ comme une séquence de commandes

continues, notée $\mathbf{U}_{cN}$, et une séquence de commandes discrètes, notée $\mathbf{U}_{dN}$, sur l'horizon de prédiction N. On a donc :

$$J_N^o(\mathbf{x}(0)) = \min_{\mathbf{U}_{dN}} \left(\min_{\mathbf{U}_{cN}} J_N(\mathbf{x}(0), (\mathbf{U}_{cN}, \mathbf{U}_{dN})) \right) \tag{4.28}$$

sous les contraintes du modèle, pour $k = 0, 1, \cdots N-1$:

$$\mathbf{x}(k+1) = \mathbf{A}_i \mathbf{x}(k) + \mathbf{B}_i \mathbf{u}_c(k) + \mathbf{a}_i \tag{4.29a}$$

$$\mathbf{y}(k) = \mathbf{C}_i \mathbf{x}(k) + \mathbf{D}_i \mathbf{u}_c(k) + \mathbf{c}_i \tag{4.29b}$$

$$\mathbf{F}_i \mathbf{x}(k) + \mathbf{G}_i \mathbf{u}_c(k) \leq \mathbf{f}_i \tag{4.29c}$$

avec $i(k) = \varphi(\mathbf{x}(k), \mathbf{u}_c(k), \mathbf{u}_d(k))$.

On note $\mathbf{I}_N = \begin{pmatrix} i(0) \; i(1) \; \cdots \; i(N-1) \end{pmatrix}^T \in \mathbb{I}^N$ une séquence de modes sur l'horizon N. Elle caractérise une séquence $\mathbf{U}_{dN}$ et N ensembles de contraintes (4.29a)–(4.29c) pour $k = 0, 1, \cdots N-1$. Le problème $\mathcal{P}_N$ peut se reformuler ainsi :

$$J_N^o(\mathbf{x}(0)) = \min_{\mathbf{I}} \left(\min_{\mathbf{U}_{cN}} J_N(\mathbf{x}(0), (\mathbf{U}_{cN}, \mathbf{I}_N)) \right) \tag{4.30}$$

Pour une séquence de modes donnée $\mathbf{I}_N$, le coût :

$$J_N^*(\mathbf{x}(0), \mathbf{I}_N) = \min_{\mathbf{U}_{cN}} J_N(\mathbf{x}(0), (\mathbf{U}_{cN}, \mathbf{I}_N)) \tag{4.31}$$

est le coût optimal trouvé en résolvant un sous-problème d'optimisation continu (la séquence $\mathbf{I}_N$ – donc $\mathbf{U}_{dN}$ – est fixée) sous contraintes. En utilisant la norme 2, la fonction de coût est quadratique[4], *i.e.* $\|\mathbf{w}\|_{\mathbf{Q}} \triangleq \mathbf{w}^T \mathbf{Q} \mathbf{w}$, ce sous-problème peut être transformé en un problème standard de programmation quadratique (QP) moyennant quelques étapes de calcul supplémentaires. L'exposant $*$ signifie l'optimalité vis à vis d'une séquence de commandes discrètes donnée. Ainsi, la séquence de commandes continues notée $\mathbf{U}_c^*$ est dite optimale en regard de la séquence de commandes discrètes $\mathbf{U}_d$ donnée[5]. Toutefois, le problème d'optimisation sous contraintes (4.31) n'est pas nécessairement faisable, *i.e.* pour la séquence discrète donnée, aucune solution ne satisfait les contraintes du modèle (4.29a)–(4.29c).

4. Si on utilise la norme 1 ou ∞ et moyennant l'ajout de variables auxiliaires, le sous-problème d'optimisation est de type LP (pour *Linear Program*).

5. Pour alléger l'écriture, on emploiera dans la suite l'expression raccourcie *séquence discrète* (resp. *continue*) pour *séquence de commandes discrètes* (resp. *continues*).

4.3.1.2. *Énumération exhaustive*

À partir de ces considérations, la méthode la plus simple pour trouver l'optimum recherché consiste à énumérer toutes les séquences de modes possibles sur l'horizon de prédiction puis résoudre les sous-problèmes QP associés aux séquences correspondantes.

Pour une séquence de modes donnée, deux cas de figure peuvent se présenter :

– il n'y a pas de solution satisfaisant les contraintes, le sous-problème QP est infaisable ;

– le sous-problème QP est faisable, une séquence continue optimale est trouvée.

Pour chaque séquence de modes $\mathbf{I}_N$ où le sous-problème QP correspondant est faisable, une séquence continue optimale $\mathbf{U}_{cN}^*$ en regard de la séquence discrète $\mathbf{U}_{dN}$ associée à $\mathbf{I}_N$ et une valeur de la fonction de coût (4.31) sont obtenues. L'optimum recherché est donc donné par la séquence de modes qui minimise (4.31) :

$$J_N^o(\mathbf{x}(0)) = \min_{\mathbf{I}_N} \left(J_N^*(\mathbf{x}(0), \mathbf{I}_N)\right) \tag{4.32}$$

Simple en apparence, cette méthode, que l'on qualifie d'énumération exhaustive, est rapidement mise en défaut lorsque le nombre de modes et (ou) la longueur de l'horizon de prédiction augmentent. C'est un phénomène d'explosion combinatoire : le nombre de possibilités à énumérer croît exponentiellement avec le nombre de modes et la longueur de l'horizon. Soit p le nombre de modes possibles pour le système. Une énumération exhaustive des séquences de modes conduit à considérer p^N sous-problèmes QP.

La figure 4.6 représente l'arbre des possibilités : chaque feuille (symbolisée par un point noir) est un sous-problème QP à résoudre et l'une d'elle est l'optimum recherché. La profondeur de l'arbre est fixée par l'horizon N et à une profondeur donnée, sa largeur est fixée par le nombre de modes possibles du système.

Tous les sous-problèmes QP sont de dimension identique, *i.e.* la dimension du vecteur d'optimisation est $\dim(\mathbf{U}_{cN}) = n_{uc} \times N$.

REMARQUE.– Nous avons vu que nombre total de modes pour le système est donné par $p = \text{card}(\mathbb{J}) \times \text{card}(\mathbb{U}_d)$. Toutefois, certains couples $(\chi_j, \mathbf{u}_d)$ ne sont pas *permis* et ne définissent donc pas un mode possible pour le système. La prise en considération de ces modes *interdits* permet de réduire le nombre de modes à énumérer.

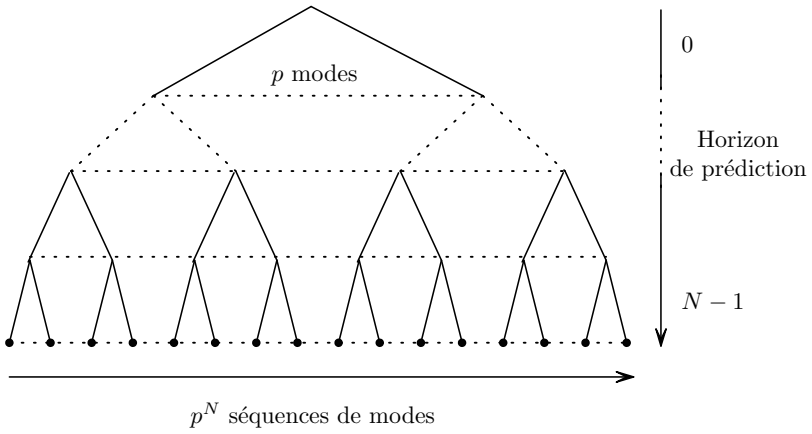


Figure 4.6. *Énumération exhaustive*

4.3.1.3. *Énumération partielle*

Dans cette section, un algorithme d'énumération partielle est proposé afin de réduire la dimension et le nombre de sous-problèmes QP à résoudre pour trouver l'optimum du problème mixte (4.28). L'énumération exhaustive consiste à parcourir la totalité de l'arbre des possibilités de la figure 4.6. L'idée clé de l'algorithme d'énumération partielle est, connaissant un sous-optimum de (4.28), d'évaluer des coûts partiels afin d'*élaguer* l'arbre, *i.e.* couper les branches qui ne peuvent pas conduire à l'optimum. La méthode se compose d'une stratégie de descente pour parcourir l'arbre des possibilités et d'un critère d'élimination de branches.

Stratégie de descente : Supposons être $P\ (< N)$ pas d'échantillonnage dans le futur, *i.e.* à une profondeur P dans l'arbre des possibilités. On définit le coût partiel J_P à ce nœud :

$$J_P(\mathbf{x}(0), \mathbf{U}_P) = \sum_{k=0}^{P-1} L(\mathbf{x}(k), \mathbf{u}(k)) \tag{4.33}$$

La stratégie proposée est de type *meilleur d'abord* :

– calculer les coûts optimaux J_{P+1} correspondant aux sous-problèmes faisables pour les différents choix possibles du mode ;

– choisir la branche donnant le coût minimal sur l'horizon $P+1$ pour continuer l'exploration.

Élimination de branches : Supposons connaître un premier sous-optimum[6]. Parcourir l'arbre en éliminant les branches pour lesquelles :

– soit le coût optimal sur un horizon réduit est supérieur au coût du sous-optimum connu ;

– soit le sous-problème est infaisable[7].

Éliminer une branche signifie éliminer toutes les autres branches issues de cette dernière. Le sous-optimum *connu* est actualisé à chaque fois qu'une feuille est évaluée et que le coût obtenu est inférieur à celui du sous-optimum précédent.

Soit une séquence de modes $\mathbf{I}_N$[8] où l'indice N signifie que la séquence est de longueur N. Soit un horizon P avec $P < N$. Les notations suivantes sont utilisées :

– $\mathbf{I}_P^{(N)}$ est la séquence des P premiers modes extraite de $\mathbf{I}_N$;

– $\mathbf{U}_{cP}^{(N)}$ est la séquence continue de longueur P extraite de $\mathbf{U}_{cN}$.

On rappelle que l'exposant $*$ signifie l'optimalité vis à vis d'une séquence de modes donnée : *i.e.* la séquence $\mathbf{U}_{cN}^{*}$ est optimale vis à vis d'une séquence de modes $\mathbf{I}_N$ donnée. Toutefois, la séquence extraite $\mathbf{U}_{cP}^{*(N)}$ n'est pas nécessairement optimale sur l'horizon P.

Le système est stationnaire et pour alléger les notations, on pose $k = 0$. Le théorème suivant s'énonce ; il est à la base du critère d'élimination de branches.

THÉORÈME.– *Soit une séquence de modes $\mathbf{I}_N$ donnée, pour tout $P < N$, le coût optimal obtenu pour la séquence $\mathbf{I}_N$ est supérieur au coût optimal obtenu pour une séquence extraite $\mathbf{I}_P^{(N)}$:*

$$\forall\, P < N,\;\; J_N^*\big(\mathbf{x}(0), \mathbf{I}_N\big) \geq J_P^*\big(\mathbf{x}(0), \mathbf{I}_P^{(N)}\big) \tag{4.34}$$

6. Toutes les feuilles de l'arbre pour lesquelles le sous-problème QP associé est faisable sont des sous-optima. L'optimum recherché est le meilleur sous-optimum.

7. On rappelle qu'un problème d'optimisation sous contraintes est dit infaisable lorsqu'il n'existe pas de solution satisfaisant les contraintes du problème.

8. On fait l'hypothèse que le sous-problème QP associé à la séquence $\mathbf{I}_N$ est faisable.

PREUVE.– Le coût (4.31) se divise en deux termes, pour tout $P < N$:

$$J_N^*\big(\mathbf{x}(0), \mathbf{I}_N\big) = J_P\big(\mathbf{x}(0), (\mathbf{U}_{cP}^{*(N)}, \mathbf{I}_P^{(N)})\big) + \sum_{l=P}^{N-1} L\big(\mathbf{x}(l), (\mathbf{u}_c^*(l), i(l))\big) + F(\mathbf{x}(N)) \tag{4.35}$$

Le coût est additif et les fonctions L et F sont définies positives donc :

$$J_N^*\big(\mathbf{x}(0), \mathbf{I}_N\big) \geq J_P\big(\mathbf{x}(0), (\mathbf{U}_{cP}^{*(N)}, \mathbf{I}_P^{(N)})\big) \tag{4.36}$$

Le coût $J_P\big(\mathbf{x}(0), (\mathbf{U}_{cP}^{*(N)}, \mathbf{I}_P^{(N)})\big)$ est le coût calculé avec une séquence extraite $\mathbf{U}_{cP}^{*(N)}$ et est donc sous-optimal sur l'horizon P :

$$\underbrace{J_P\big(\mathbf{x}(0), (\mathbf{U}_{cP}^{*(N)}, \mathbf{I}_P^{(N)})\big)}_{\text{coût sous-optimal sur } P} \geq \underbrace{J_P^*\big(\mathbf{x}(0), \mathbf{I}_P^{(N)}\big)}_{\text{coût optimal sur } P} \tag{4.37}$$

□

Supposons être à une profondeur $P < N$ et que le coût associé J_P est supérieur à un sous-optimum[9] connu, en vertu du théorème énoncé précédemment, la branche correspondante et toutes les branches qui en sont issues peuvent être coupées.

REMARQUE.–

1) La présence d'une contrainte égalité sur l'état final $\mathbf{x}(N) = \mathbf{x}_r$ (resp. un terme sur l'état final $F(\mathbf{x}(N))$ n'est à considérer que lors de l'évaluation d'un coût sur l'horizon complet (N), *i.e.* une feuille de l'arbre de la figure 4.6. Cette contrainte (resp. ce terme) n'existe pas lors de l'évaluation d'un coût partiel ($P < N$).

2) Dans le cas d'un problème d'optimisation purement combinatoire, le coût d'un nœud dans l'arbre des possibilités est obtenu par la somme des coûts associés aux branches menant à ce nœud. Le problème considéré dans cette section est de nature mixte. Le coût d'une branche n'est pas connu à priori puisqu'il dépend de la commande continue $\mathbf{u}_c^*$ trouvée en résolvant le sous-problème QP associé au chemin emprunté dans l'arbre. Soit un nœud à une profondeur $P < N$, le coût associé à ce nœud n'est pas la somme des coûts de branche mais le coût obtenu en résolvant un sous-problème QP sur l'horizon P.

9. Un sous-optimum est une feuille de l'arbre dont le coût est obtenu en résolvant un sous-problème QP sur l'horizon N.

Les différentes étapes de l'algorithme vont être illustrées graphiquement. Pour simplifier l'explication, tous les sous-problèmes sont supposés faisables[10]. On rappelle que les feuilles de l'arbre sont représentées par des points noirs. Pour compléter, les symboles graphiques suivants sont utilisés :

– l'évaluation d'un coût partiel est représentée par un rond ;

– la descente vers le premier sous-optimum est illustrée par des traits gras et des ronds grisés ;

– le premier sous-optimum et un meilleur sous-optimum suivant sont représentés par un rond en trait gras ;

– lorsqu'un coût partiel est supérieur au meilleur sous-optimum connu (horizon N), il est barré d'une croix (coupure de branches) ;

– l'optimum est repéré par un triangle.

La stratégie de descente proposée est illustrée sur la figure 4.7 pour l'obtention du premier sous-optimum. Partant de la racine (en haut de l'arbre), les p nœuds sont évalués et on choisit la branche (*i.e.* le mode) pour laquelle le coût optimal J_1^* est minimal (branche 1). On recommence la même procédure jusqu'à atteindre le bas de l'arbre, *i.e.* l'horizon N (branches numérotées de 2 à 4). Un premier sous-optimum est donc obtenu en considérant la séquence de N modes repérée par les branches 1 à 4.

L'élimination de branches est illustrée figure 4.8. Il y a de multiples façons de parcourir l'arbre pour évaluer des coûts partiels afin de couper des branches. À l'issue de l'étape précédente de descente, un certain nombre de nœuds ont déjà été évalués (nœuds repérés par un rond) et l'arbre peut être parcouru à rebours jusqu'à trouver (ou non) une nouvelle possibilité de descente, *i.e.* un coût partiel inférieur au meilleur sous-optimum connu.

Sur la figure 4.8, en remontant d'un pas de temps, on suppose pour l'exemple qu'il n'y a pas de *bon candidat* (branche 5), *i.e.* pas de nœud pour lequel le coût partiel est inférieur au meilleur sous-optimum connu. Une croix symbolise alors l'abandon de toutes les branches attenantes. On remonte à nouveau d'un pas de temps et on suppose que la branche 6 fournit une possibilité de descente (branches 7 et 8).

Les branches issues de la branche 9 (*cf.* figure 4.9) sont coupées car le coût partiel à ce nœud est supérieur au dernier sous-optimum connu et on remonte jusqu'à atteindre

10. C'est un cas très particulier à but pédagogique. Les sous-problèmes infaisables sont intéressants du point de vue de la réduction de complexité combinatoire puisqu'ils permettent de couper les branches restantes sans considération de coût.

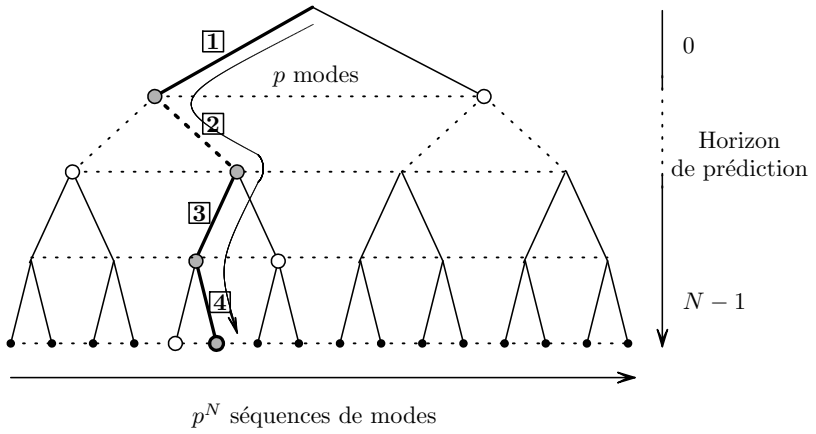


Figure 4.7. *Énumération partielle : stratégie de descente et premier sous-optimum*

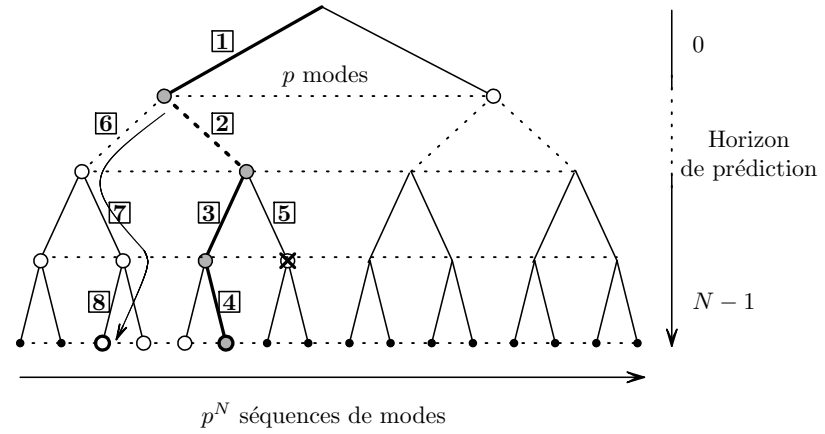


Figure 4.8. *Énumération partielle : élimination de branches et descente vers un meilleur sous-optimum*

la racine car il n'y a plus de possibilité de descente (branche 10). Le dernier sous-optimum trouvé est donc l'optimum recherché (symbolisé par un triangle). Seuls les nœuds marqués d'un rond ont été évalués.

Dans cette approche par énumération partielle, la dimension des sous-problèmes QP à résoudre commence à n_{uc} au début de l'arbre (prédiction à un pas) pour croître avec l'horizon jusqu'à $n_{uc} \times N$ en bas de l'arbre (horizon N).

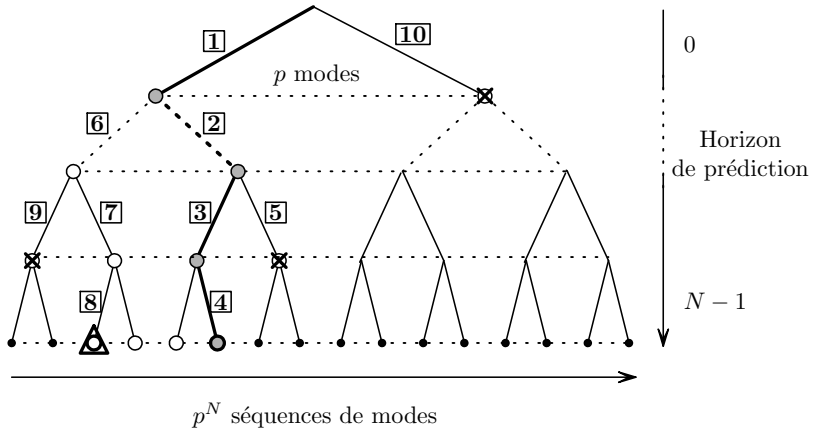


Figure 4.9. *Énumération partielle : élimination de branches et obtention de l'optimum*

REMARQUE.–

1) L'algorithme d'énumération partielle est un algorithme de type branch & bound qui conduit à la solution optimale en exploitant la structure particulière du problème d'optimisation associé à la commande prédictive : la hauteur de l'arbre est fixée par la longueur de l'horizon de prédiction. Les techniques de programmation mixte entière à base de branch & bound (*cf.* section 4.3.2.2) n'exploitent pas cette caractéristique. Elles ont été développées pour résoudre des problèmes standards, purement combinatoires ou mixtes.

2) La stratégie de descente proposée est une heuristique qui permet d'obtenir un premier sous-optimum qu'on espère de bonne qualité, *i.e.* peu éloigné de l'optimum. Le caractère sous-optimal provient du choix de la séquence de modes (stratégie du meilleur d'abord à un pas de prédiction) mais le sous-optimum est obtenu en résolvant un sous-problème QP sur l'horizon complet N.

4.3.1.4. *Discussion*

L'algorithme d'énumération exhaustive est simple à implémenter mais impossible à utiliser pour des problèmes non-triviaux. L'algorithme d'énumération partielle permet d'éviter cette recherche exhaustive et convient pour des problèmes de taille plus importante. Au lieu d'essayer une par une les solutions possibles (feuilles de l'arbre), l'arbre est parcouru *intelligemment* à la recherche du meilleur sous-optimum. La rapidité de convergence est liée aux caractéristiques du problème, notamment par la fonction de coût. La comparaison des deux algorithmes est résumée ci-après :

Énumération exhaustive :

+ (très) simple à mettre en œuvre
– explosif
– dimension fixe ($n_{uc} \times N$) des sous-problèmes QP
– algorithme simpliste (il faut tout essayer)

Énumération partielle :

+ simple à mettre en œuvre
+ exploration *intelligente*
+ recherche de sous-optima
+ dimension variable (de n_{uc} à $n_{uc} \times N$) des sous-problèmes QP
– le parcours complet de l'arbre n'est pas exclu

Le dernier point est un cas particulier (peu probable) : si pour chaque nœud à une profondeur $P < N$, le coût (partiel) est inférieur au dernier sous-optimum connu, l'arbre sera parcouru dans sa totalité.

4.3.2. *Commande prédictive des systèmes MLD*

4.3.2.1. *Mise en forme*

Une fois le système modélisé par une forme MLD (4.14a)–(4.14c), le problème d'optimisation associé à la commande prédictive peut s'écrire sous la forme d'un problème standard d'optimisation mixte. Le système est stationnaire, on peut donc poser $k = 0$ sans perte de généralité.

Soit $\mathbf{U} = (\mathbf{u}^T(0)\ \mathbf{u}^T(1) \cdots\ \mathbf{u}^T(N-1))^T$ la séquence de commandes sur l'horizon de prédiction. À chaque pas d'échantillonnage, il faut résoudre le problème suivant :

$$\begin{aligned}\min_{\mathbf{U}} J(\mathbf{x}(0), \mathbf{U}) = \min_{\mathbf{U}} \sum_{l=0}^{N-1} &\|\mathbf{x}(l+1) - \mathbf{x}_e\|_{\mathbf{Q}_x} + \|\mathbf{y}(l) - \mathbf{y}_e\|_{\mathbf{Q}_y} \\ &+ \|\mathbf{u}(l) - \mathbf{u}_e\|_{\mathbf{Q}_u} + \|\boldsymbol{\delta}(l) - \boldsymbol{\delta}_e\|_{\mathbf{Q}_\delta} + \|\mathbf{z}(l) - \mathbf{z}_e\|_{\mathbf{Q}_z} \end{aligned} \tag{4.38}$$

sous les contraintes (pour $l = 0$ à $N-1$)

$$\mathbf{x}(l+1) = \mathbf{A}_1\mathbf{x}(l) + \mathbf{B}_1\mathbf{u}(l) + \mathbf{B}_2\boldsymbol{\delta}(l) + \mathbf{B}_3\mathbf{z}(l) \tag{4.39a}$$

$$\mathbf{y}(l) = \mathbf{C}_1\mathbf{x}(l) + \mathbf{D}_1\mathbf{u}(l) + \mathbf{D}_2\boldsymbol{\delta}(l) + \mathbf{D}_3\boldsymbol{z}(l) \tag{4.39b}$$

$$\mathbf{E}_2\boldsymbol{\delta}(l) + \mathbf{E}_3\mathbf{z}(l) \leq \mathbf{E}_1\mathbf{u}(l) + \mathbf{E}_4\mathbf{x}(l) + \mathbf{E}_5 \tag{4.39c}$$

en ajoutant éventuellement une contrainte sur l'état final

$$\mathbf{x}(N) = \mathbf{x}_e \tag{4.40}$$

Le couple $(\boldsymbol{\delta}_e, \mathbf{z}_e)$ est déterminé à partir du couple $(\mathbf{x}_e, \mathbf{u}_e)$, appelé paire d'équilibre, en résolvant un problème MILP de manière similaire au test de faisabilité (*cf.* section 4.2.2.3). $\mathbf{y}_e$ est la consigne pour la sortie. N est la longueur de l'horizon de prédiction. Les matrices de pondération ont les propriétés suivantes : $\mathbf{Q}_{x,u}$ sont symétriques définies positives et $\mathbf{Q}_{\delta,z,y}$ sont symétriques définies semi-positives.

On remarquera que le problème d'optimisation (critère et contraintes) fait intervenir les variables de commande $\mathbf{u}$ mais aussi les variables auxiliaires $\boldsymbol{\delta}$ et $\mathbf{z}$. C'est le prix à payer pour les différentes transformations en inégalités linéaires mixtes. De ce fait le vecteur à optimiser est augmenté : du point de vue de la commande, on cherche $\mathbf{U}$ optimal sur l'horizon de prédiction mais du point de vue de l'optimisation, on cherche $(\mathbf{U}^T\ \boldsymbol{\Delta}^T\ \mathbf{Z}^T)^T$ où $\boldsymbol{\Delta}$ et $\mathbf{Z}$ sont les séquences de variables $\boldsymbol{\delta}$ et $\mathbf{z}$ sur l'horizon ! La dimension du problème d'optimisation est donc liée non pas seulement à la dimension du vecteur de commandes, mais aussi à la dimension des vecteurs de variables auxiliaires binaires (complexité combinatoire) et continues (dimension des sous-problèmes continus dans la technique de résolution branch & bound). C'est l'inconvénient majeur de l'approche MLD.

En utilisant la norme quadratique dans le critère de (4.38), le problème d'optimisation peut se mettre sous la forme (moyennant quelques étapes de calcul) :

$$\min_{\mathbf{V}} \frac{1}{2}\mathbf{V}^T\mathbf{H}\mathbf{V} + \mathbf{f}^T\mathbf{V} \tag{4.41}$$

sous les contraintes

$$\mathbf{A}_{in}\mathbf{V} \leq \mathbf{b}_{in} \tag{4.42}$$

$$\mathbf{A}_{eq}\mathbf{V} = \mathbf{b}_{eq} \tag{4.43}$$

où $\mathbf{V} = (\mathbf{U}^T\ \boldsymbol{\Delta}^T\ \mathbf{Z}^T)^T$. C'est un problème dit de type MIQP (*Mixed Integer Quadratic Program*)[11].

11. Si la norme utilisée est 1 ou ∞ et moyennant l'ajout de variables supplémentaires, le critère à optimiser est alors linéaire et le problème d'optimisation est de type MILP.

4.3.2.2. *Programmation mixte entière*

Pour résoudre le problème d'optimisation MIQP défini précédemment, la technique couramment employée est le branch & bound (ou méthode de séparation et évaluation) qui a pour but d'éviter une énumération exhaustive des variables entières (ou binaires dans le cas des systèmes MLD) du problème. Si le problème considéré est de type MILP, la technique est identique et il suffit de remplacer quadratique par linéaire (et QP/MIQP par LP/MILP) dans le texte ci-dessous.

La relaxation consiste à modifier les contraintes sur les entiers, de manière à transformer un problème quadratique mixte (MIQP) en un problème quadratique standard (QP). Quatre cas possibles sont à envisager :

1) le problème QP est sans solution (il est dit infaisable) donc le problème MIQP est infaisable car l'ensemble des solutions du problème MIQP est contenu dans celui du problème QP obtenu par relaxation des variables entières ($MIQP \subseteq QP$) ;

2) le problème QP a une solution, mais au moins une contrainte d'intégrité (contrainte imposant à une variable d'être entière) n'est pas satisfaite, auquel cas le problème MIQP n'est pas (encore) résolu ;

3) le problème QP a une solution et toutes les contraintes d'intégrité sont satisfaites, auquel cas le problème MIQP est résolu ;

4) le problème QP n'est pas borné.

Le dernier cas (4) est critique et correspond à un problème mal posé. Dans le cas des systèmes MLD, les variables $\mathbf{u}$ sont en général bornées (contraintes sur les actionneurs par ex.) et les variables auxiliaires $\boldsymbol{\delta}$ et $\mathbf{z}$ sont bornées par définition : ce dernier cas ne peut donc pas se produire. Les cas (1) et (3) sont simples : soit le problème est sans solution, soit la solution existe et elle a été trouvée. Le cas (2) est plus complexe et fait intervenir le concept de séparation pour continuer la résolution.

Quand une ou plusieurs contraintes d'intégrité ne sont pas satisfaites, la technique du branch & bound consiste alors à parcourir un arbre dont chaque nœud est un sous-problème MIQP. Soit le problème MIQP standard suivant :

$$\min_{\mathbf{x}} \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{f}^T\mathbf{x} \tag{4.44}$$

sous les contraintes

$$\mathbf{A}\mathbf{x} \leq \mathbf{b} \tag{4.45}$$

$$\mathbf{x} = \begin{pmatrix} \mathbf{x_c} \\ \mathbf{x_d} \end{pmatrix},\ \mathbf{x_c} \in \mathbb{R}^{n_c}, \mathbf{x_d} \in \mathbb{N}^{n_d} \tag{4.46}$$

La technique du branch & bound est décrite par l'algorithme suivant qui comporte quatre étapes principales.

Algorithme du branch & bound :

1) Initialiser une liste de sous-problèmes avec le problème MIQP original à résoudre. Prendre le coût optimal $J_{opt} = +\infty$ et la solution optimale $\mathbf{x}_{opt} = (+\infty, \cdots, +\infty)^T$;

2) - Si la liste n'est pas vide, sélectionner et extraire un sous-problème dans la liste. Relaxer puis résoudre ce sous-problème. On note le coût obtenu J_r[12] et la solution correspondante $\mathbf{x}_r$;

- Sinon (la liste est vide), terminer avec J_{opt} et x_{opt}. Si $J_{opt} = +\infty$, le problème MIQP est infaisable ;

3) - Si le sous-problème est infaisable ou si $J_r \geq J_{opt}$. Aller au point 2 ;

- Sinon, si $\mathbf{x}_r$ satisfait toutes les contraintes d'intégrité alors mettre à jour l'optimum courant : $J_{opt} = J_r$, $\mathbf{x}_{opt} = \mathbf{x}_r$ et aller au point 2 ;

4) ($J_r < J_{opt}$ et au moins une contrainte d'intégrité n'est pas satisfaite) Générer deux nouveaux sous-problèmes à partir du sous-problème courant, les ajouter à la liste des sous-problèmes à résoudre (étape de séparation) et aller à l'étape 2.

Dans cet algorithme, les étapes 2 et 4 font intervenir des choix de stratégie qui influencent fortement les performances. Un *bon* algorithme branch & bound doit rapidement élaguer l'arbre, *i.e* passer souvent par l'étape 3, ce qui permet d'éviter la résolution d'un grand nombre de sous-problèmes. Le choix de la variable de séparation à l'étape 4 correspond à une **stratégie de branchement**. Plusieurs stratégies sont possibles et parmi les plus courantes, on peut citer :

– Première variable libre : parmi les variables entières relaxées, choisir celle de plus petit indice ;

– Branchement suivant la partie fractionnaire : à l'issue de la résolution QP, les variables entières relaxées ont en général une partie fractionnaire. Choisir la variable qui a la plus petite ou la plus grande distance à la valeur entière la plus proche ;

– Priorité de l'utilisateur : la séquence de branchement est spécifiée par l'utilisateur. Dans le cadre de la commande prédictive des systèmes MLD, ce choix permet de brancher en priorité sur les variables binaires intervenant au début de l'horizon de prédiction.

La sélection du sous-problème à résoudre à l'étape 2 se fait suivant une **stratégie de descente** (parcours d'arbre). Comme précédemment, plusieurs stratégies sont possibles :

12. En utilisant un vocabulaire propre à l'optimisation, J_r est aussi appelé borne inférieure. De même, J_{opt} est appelé borne supérieure.

– Parcours en profondeur (*depth first*) : les sous-problèmes sont sélectionnés suivant une règle du type dernier entré–premier sorti (LIFO pour *Last In–First Out*) ;

– Parcours en largeur (*breadth first*) : les sous-problèmes à une profondeur P sont résolus lorsque tous les sous-problèmes à la profondeur $P-1$ ont été résolus ;

– Meilleur d'abord (*best bound*) : le sous-problème choisi est celui dont la relaxation donne le coût minimal.

L'algorithme du branch & bound est illustré avec l'exemple ci-dessous. Il s'agit de résoudre un problème purement combinatoire à 4 variables binaires, *i.e.* $\mathbf{x} \in \{0,1\}^4$. Sur les figures 4.10 à 4.13, une $\star$ indique une variable relaxée, *i.e.* prenant ses valeurs dans l'intervalle $[0,1]$. La stratégie de branchement est la première variable libre et la stratégie de descente est un parcours en largeur.

Figure 4.10 : le problème MIQP est relaxé (1), il est faisable et deux sous-problèmes sont générés par séparation de la première variable. Le premier sous-problème relaxé (2) donne une solution entière et une première valeur pour J_{opt} (borne supérieure). La résolution du sous-problème relaxé (3) fournit une solution ne satisfaisant pas les contraintes d'intégrité et telle que $J_r < J_{opt}$, il faut *brancher*.

Figure 4.11 : la séparation du sous-problème (3) sur la deuxième variable génère deux nouveaux sous-problèmes. Le premier sous-problème relaxé (4) ne donne pas une solution entière et $J_r < J_{opt}$, il faudra donc développer ce nœud. Le deuxième sous-problème relaxé (5) est infaisable, il est donc inutile de continuer dans cette direction.

Figure 4.12 : la séparation du sous-problème (4) sur la troisième variable génère deux nouveaux sous-problèmes. Le premier sous-problème relaxé (6) ne donne pas une solution entière avec un coût associé $J_r > J_{opt}$, il est inutile de développer ce nœud. Le deuxième sous-problème relaxé (7) est faisable ($J_r < J_{opt}$) et il faut brancher.

Figure 4.13 : la séparation du sous-problème (7) sur la dernière variable donne deux sous-problèmes (supposés) faisables. La première solution (8) est meilleure que celle trouvée lors de la résolution du sous-problème (2) et fournit une nouvelle valeur pour J_{opt}. Le dernier sous-problème (9) n'apporte pas une meilleure solution, l'optimum est donc $(1\,0\,1\,0)$.

Une énumération exhaustive des solutions conduirait, dans cet exemple simple, à la résolution de $2^4 = 16$ sous-problèmes QP. L'algorithme du branch & bound permet ici de n'en résoudre que 9. Les problèmes MI(Q)P sont dits NP-difficiles car leur complexité dépend exponentiellement du nombre de variables entières. Même si rien ne garantit qu'il ne faudra pas parcourir la totalité de l'ensemble des solutions, en pratique les algorithmes de ce type permettent d'envisager la résolution de problèmes de grande taille en un temps raisonnable. Comme cela a été évoqué précédemment, le paramétrage de l'algorithme via les choix de stratégie (branchement et descente)

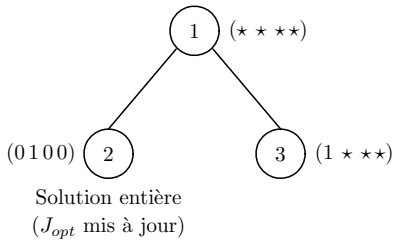


Figure 4.10. *Branch & bound (1/4)*

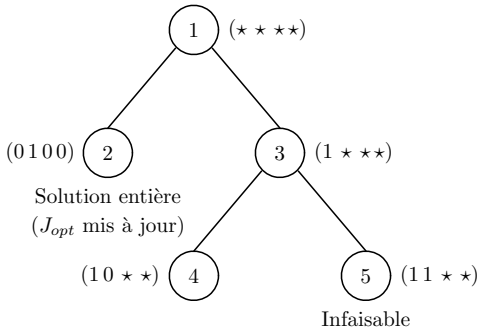


Figure 4.11. *Branch & bound (2/4)*

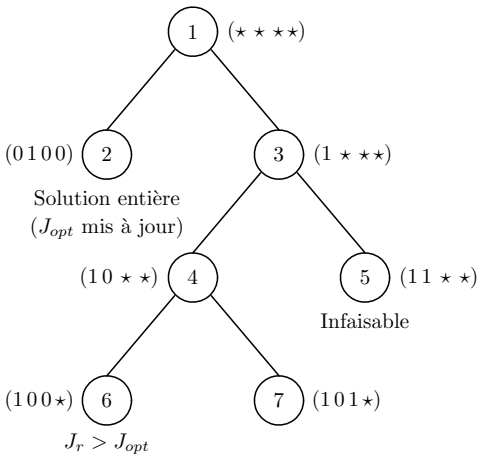


Figure 4.12. *Branch & bound (3/4)*

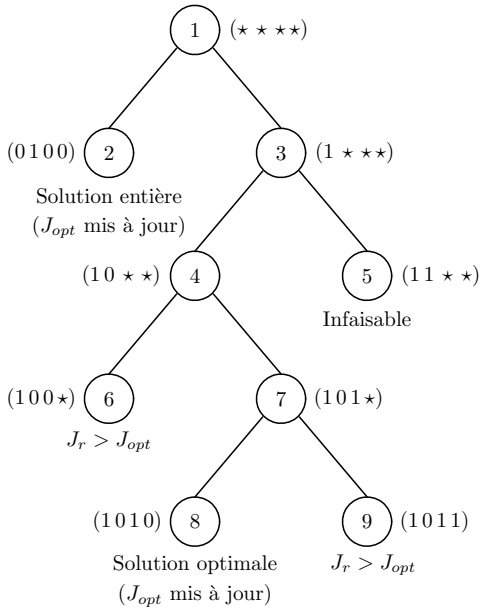


Figure 4.13. *Branch & bound (4/4)*

a une grande influence sur les performances. Les solveurs évolués disposent aussi de pré-traitements pour supprimer ou resserrer des contraintes. On peut aussi initialiser l'algorithme avec une solution entière satisfaisant les contraintes du problème, si on en connaît une... ce qui n'est en général pas évident à priori. Certains solveurs comportent une heuristique de recherche d'une première solution, *i.e.* un premier sous-optimum, afin de démarrer le branch & bound avec une borne supérieure si possible peu éloignée du minimum recherché. Dans le cadre d'une commande prédictive, on peut par exemple initialiser l'algorithme avec la solution trouvée à l'instant précédent sous réserve que cette dernière satisfasse les contraintes du nouveau problème.

4.3.3. *Exemple d'application*

Pour illustrer les formalismes et techniques présentés dans ce chapitre, l'exemple d'application choisi est un cas d'étude classique de l'automatique hybride : le système à trois cuves [LUN 98] dont un schéma est représenté figure 4.14. Le système est composé de trois cuves numérotées de 1 à 3 de hauteur h_{max}. Les cuves 1 et 2 sont alimentées en liquide par deux pompes dont les débits respectifs Q_1 et Q_2 peuvent varier entre 0 et Q_{max}. Quatre vannes V_{13l}, V_{23l}, V_{13u} et V_{23u} permettent de contrôler les flux entre les cuves. On suppose qu'elles ne peuvent prendre que deux états : ouvert (1) ou fermé (0). Les vannes supérieures sont situées à une hauteur h_v. Les vannes V_{1f},

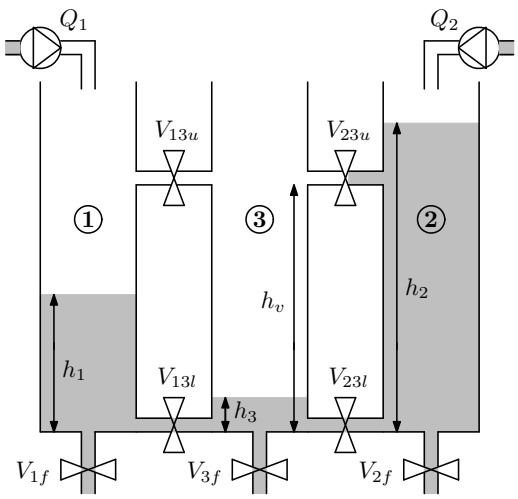


Figure 4.14. *Système à trois cuves*

V_{2f} et V_{3f} fixent les flux sortants. Les niveaux de liquide dans les trois cuves, qui sont les grandeurs à contrôler, sont notés h_1, h_2 et h_3 respectivement.

La conservation de la masse dans les cuves permet d'écrire les équations différentielles suivantes :

$$\dot{h}_1 = \frac{1}{S_c}(Q_1 - Q_{13l} - Q_{13u} - Q_{1f}) \tag{4.47a}$$

$$\dot{h}_2 = \frac{1}{S_c}(Q_2 - Q_{23l} - Q_{23u} - Q_{2f}) \tag{4.47b}$$

$$\dot{h}_3 = \frac{1}{S_c}(Q_{13l} + Q_{13u} + Q_{23l} + Q_{23u} - Q_{3f}) \tag{4.47c}$$

où S_c est la section des cuves et les grandeurs Q sont les différents flux entre les cuves et flux sortants. La loi de Torricelli en donne les expressions qui sont linéarisées et présentées ci-dessous :

$$Q_{13l} \approx k_l V_{13l}(h_1 - h_3) \tag{4.48a}$$

$$Q_{23l} \approx k_l V_{23l}(h_2 - h_3) \tag{4.48b}$$

$$Q_{if} \approx k_l V_{if} h_i \text{ pour } i \in \{1, 2, 3\} \tag{4.48c}$$

avec $k_l = aS_v\sqrt{2g/h_{max}}$. a est une constante liée au liquide, S_v est la section des vannes et g est l'accélération de la pesanteur. Pour le débit Q_{13u}, il faut considérer les différents cas possibles pour les niveaux de liquide h_1 et h_3 :

– si $h_1 \geq h_v$ et $h_3 \geq h_v$ alors $Q_{13u} = k_u V_{13u}(h_1 - h_3)$;
– si $h_1 \leq h_v$ et $h_3 \geq h_v$ alors $Q_{13u} = k_u V_{13u}(h_v - h_3)$;
– si $h_1 \geq h_v$ et $h_3 \leq h_v$ alors $Q_{13u} = k_u V_{13u}(h_1 - h_v)$;
– si $h_1 \leq h_v$ et $h_3 \leq h_v$ alors $Q_{13u} = 0$.

avec $k_u = aS_v\sqrt{2g/(h_{max} - h_v)}$. Il en va de même pour le débit Q_{23u} dont l'expression, qui est analogue à Q_{13u}, n'est pas détaillée. L'espace d'état engendré par h_1, h_2 et h_3 en donc partitionné en huit régions contigües. En considérant les deux positions (ouverte ou fermée) des quatre vannes, le nombre total de modes est alors $8 \times 2^4 = 128$.

Un modèle MLD est obtenu avec HYSDEL en introduisant les variables auxiliaires suivantes :

$$\boldsymbol{\delta} = (\delta_1\ \delta_2\ \delta_3)^T \tag{4.49}$$

$$\mathbf{z} = (z_1\ z_2\ z_3\ \ z_{13l}\ z_{23l}\ \ z_{13u}\ z_{23u})^T \tag{4.50}$$

avec

$$[\delta_i = 1] \leftrightarrow [h_i \geq h_v] \quad \text{pour} \quad i = 1, 2, 3 \tag{4.51}$$

et

$$z_i = \delta_i(h_i - h_v) \quad \text{pour} \quad i = 1, 2, 3 \tag{4.52a}$$

$$z_{i3l} = V_{i3l}(h_i - h_3) \quad \text{pour} \quad i = 1, 2 \tag{4.52b}$$

$$z_{i3u} = V_{i3u}(z_i - z_3) \quad \text{pour} \quad i = 1, 2 \tag{4.52c}$$

L'état et la sortie du système sont composés des hauteurs de liquide dans les trois cuves $\mathbf{x} = \mathbf{y} = (h_1\ h_2\ h_3)^T$, donc de dimension 3. L'entrée du système est $\mathbf{u} = (Q_1\ Q_2\ \ V_{13l}\ V_{23l}\ V_{13u}\ V_{23u}\ )^T$ de dimension 6. Le modèle MLD comporte 44 inégalités linéaires mixtes. Les équations (4.47a)–(4.47c) sont discrétisées par la

Paramètre	Valeur	Description
a	1	Constante liée au liquide
h_{max}	1	Hauteur maximale (m)
h_v	0.8	Hauteur des vannes supérieures (m)
Q_{max}	$2\,10^{-4}$	Débit maximal des pompes (m^3/s)
S_c	$3.14\,10^{-2}$	Section des cuves (m^2)
S_v	$1.76\,10^{-4}$	Section des vannes (m^2)
$V_{if,\ i=1,2,3}$	0.1	Ouverture des vannes de fuite
T_s	10	Période d'échantillonnage (s)

Tableau 4.1. *Paramètres pour le système à trois cuves*

méthode d'Euler afin d'obtenir des modèles à temps discret. Les valeurs numériques utilisées pour les simulations sont rassemblées dans le tableau 4.1.

L'état initial est :

$$\mathbf{x} = (0.4\ 0.9\ 0.1)^T \tag{4.53}$$

Les consignes pour les niveaux de liquide sont :

$$\mathbf{y}_r = \mathbf{x}_r = (0.6\ 0.8\ 0.5)^T \tag{4.54}$$

Les pondérations utilisées dans la fonction de coût sont :

$$\mathbf{Q}_u = \text{diag}(1\ 1\ \ 1\ 1\ 1\ 1) \tag{4.55a}$$

$$\mathbf{Q}_y = \mathbf{Q}_x = \text{diag}(10^2\ 10^2\ 10^3) \tag{4.55b}$$

et $\mathbf{Q}_\delta = \mathbf{Q}_z = 0$ (MLD). On ne considère pas de coût ni de contrainte égalité sur l'état final.

Des simulations ont été effectuées pour différents horizons de prédiction ($N = 2, 3, 4$). Les figures 4.15, 4.16 et 4.17 représentent respectivement les niveaux de liquide dans les trois cuves, les commandes continues et discrètes, et la trajectoire dans l'espace d'état obtenus pour $N = 3$. L'état à atteindre $\mathbf{x}_r$ n'est pas un état d'équilibre pour le système et les niveaux des cuves 1 et 3 oscillent autour de leur valeur de consigne respective suivant l'ouverture et la fermeture de la vanne inférieure V_{13l} qui permet le passage du liquide entre ces deux cuves.

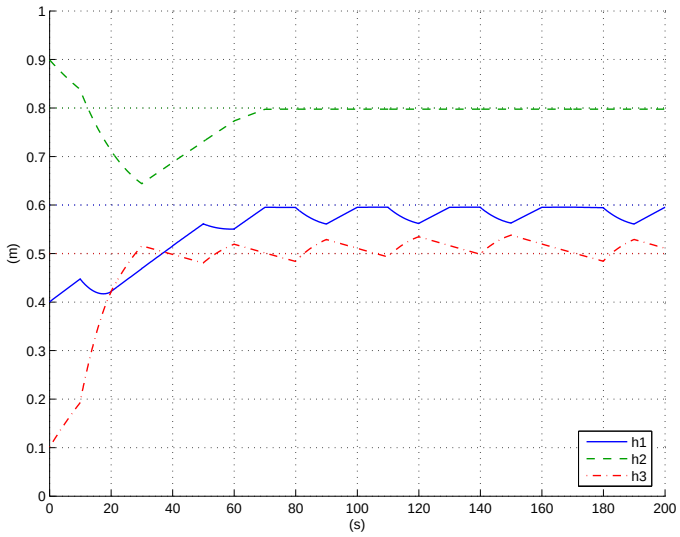


Figure 4.15. *Niveaux de liquide ($N = 3$)*

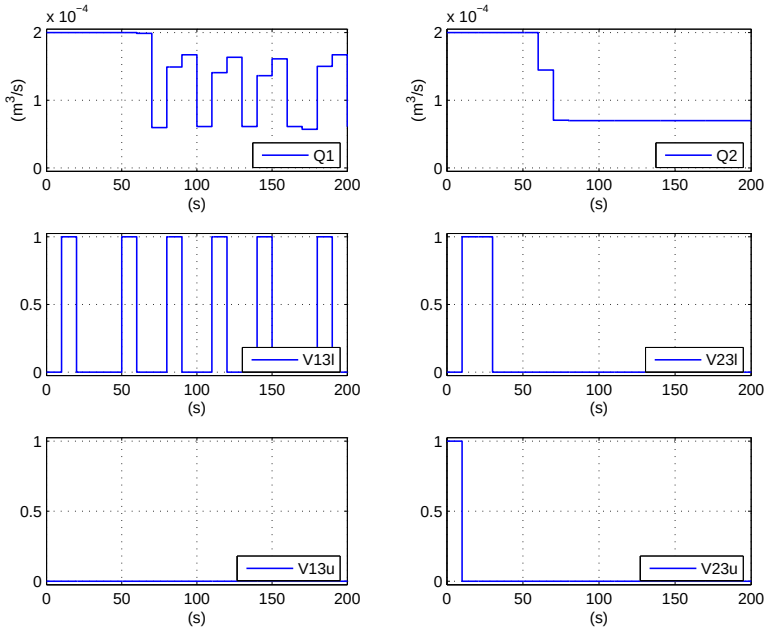


Figure 4.16. *Entrées de commande ($N = 3$)*

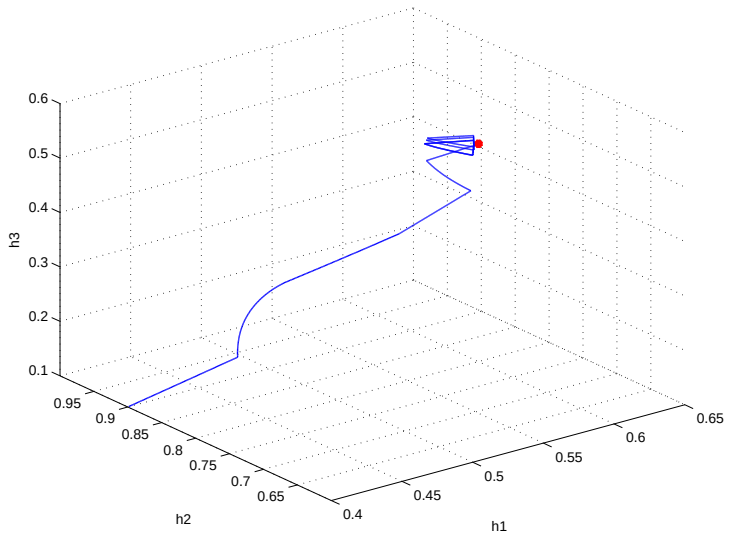
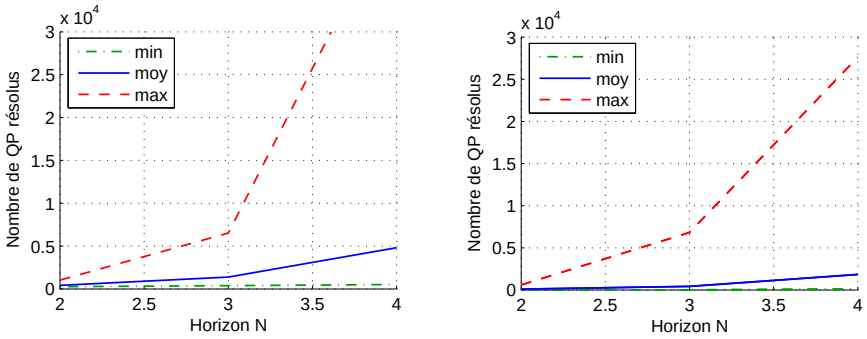


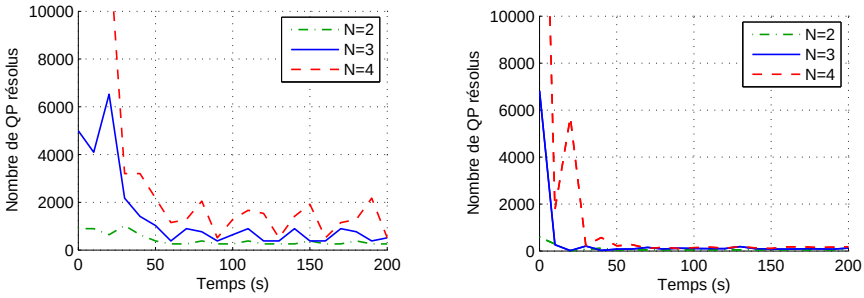
Figure 4.17. *Trajectoire dans l'espace d'état ($N = 3$)*



(a) PWA (b) MLD

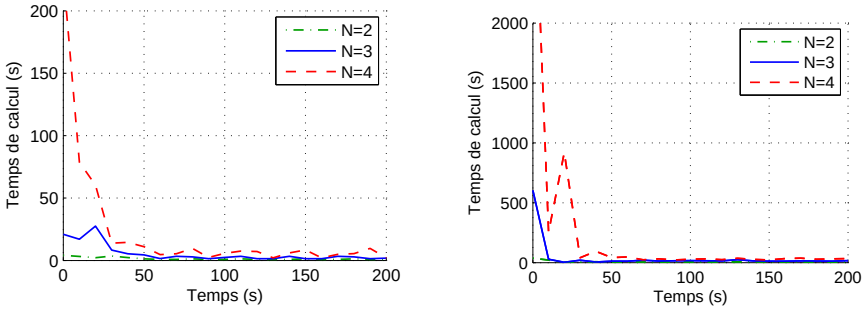
Figure 4.18. *Nombre de sous-problèmes QP (minimum, moyen et maximum) résolus en fonction de l'horizon de prédiction N*

La figure 4.18 représente les nombres minimaux, moyens et maximaux de sous-problèmes QP résolus pour atteindre l'optimum recherché en fonction de la longueur N de l'horizon de prédiction. On peut observer que la complexité reste exponentielle mais que les algorithmes d'optimisation utilisés – énumération partielle (PWA) et branch & bound (MLD) – permettent d'éviter l'explosion combinatoire.



(a) PWA (b) MLD

Figure 4.19. *Nombre de sous-problèmes QP résolus en fonction du temps ($N = 2, 3, 4$)*



(a) PWA (b) MLD

Figure 4.20. *Temps de calcul utilisé en fonction du temps ($N = 2, 3, 4$) – Les échelles en ordonnée diffèrent d'un facteur 10*

La figure 4.19 représente le nombre de sous-problèmes QP résolus en fonction du temps, *i.e.* à chaque instant d'échantillonnage lors des simulations, pour les différents horizons de prédiction considérés ($N = 2, 3, 4$). La figure 4.20 représente le temps de calcul utilisé pour résoudre ces sous-problèmes à chaque instant[13]. Outre le nombre de sous-problèmes QP résolus pour trouver l'optimum, le temps de calcul nécessaire est aussi lié à la dimension des sous-problèmes. Le formalisme MLD requiert l'ajout de $r_l = 3$ variables auxiliares binaires et $r_c = 7$ variables auxiliaires réelles. La

13. Ces résultats ont éte obtenus avec un ordinateur de type PC équipé d'un processeur Centrino cadencé à 2 GHz et de la version 7.0 du logiciel Matlab. Les algorithmes utilisés ont été développés en code Matlab et le *solveur* utilisé pour résoudre les sous-problèmes QP est " quadprog " issu de la boîte à outils " Optimization ".

	énum. partielle (PWA)		branch & bound (MLD)	
N	QP max	temps (s)	QP max	temps (s)
2	1024	4.34	589	35.1
3	6528	27.8	6821	591
4	44928	233.3	27609	3583

Tableau 4.2. *Performances comparées*

dimension maximale d'un sous-problème est alors $(n_{uc} + n_{ul} + r_c + r_l) \times N$. Pour l'énumération partielle, la dimension maximale d'un sous-problème est $n_{uc} \times N$. Sur la figure 4.20, les échelles en ordonnée (temps de calcul) diffèrent d'un facteur 10 ! Le tableau 4.2 rassemblent les nombres maximaux de sous-problèmes QP résolus et les temps de calculs associés.

4.4. Conclusion

Ce chapitre a présenté deux approches pour la commande prédictive des systèmes hybrides, le choix de l'une où l'autre dépendant en partie des outils disponibles. Le formalisme PWA consiste en un ensemble de modèles et de domaines de validité associés. Le formalisme MLD rassemble dans un unique (et gros) modèle mathématique tous les modèles propres à chaque mode. La mise en œuvre de la commande prédictive pour un modèle MLD aboutit à un problème standard de taille importante qui requiert l'utilisation de solveurs puissants (et coûteux) comportant un algorithme de type branch & bound, par exemple : CPLEX, XPRESS. L'algorithme d'énumération partielle exploite la structure particulière du problème sans avoir recours à l'ajout de variables auxiliaires. L'exemple d'application de la section précédente a mis en lumière ces aspects.

Les travaux de recherche en cours visent à maîtriser la complexité : déport de calculs hors ligne faisant appel à la programmation multi-paramétrique [BEM 02], recherche de sous-optimums en utilisant des heuristiques et des algorithmes génétiques [THO 04a, THO 04b], élimination de branches par calculs d'atteignabilité [PEñ 03]. Les approches décrites dans ce chapitre ont été appliquées à des cas d'étude de réseaux d'énergie électrique [LEI 05a, LEI 05b, LEI 05c].

Chapitre 5

Bibliographie

[BEM 99] BEMPORAD A., MORARI M., « Control of Systems Integrating Logic, Dynamics, and Constraints », *Automatica*, vol. 35, n°3, p. 407–427, 1999.

[BEM 00] BEMPORAD A., FERRARI-TRECATE G., MORARI M., « Observability and Controllability of PieceWise Affine and Hybrid Systems », *IEEE Transactions on Automatic Control*, vol. 45, n°10, p. 1864–1876, 2000.

[BEM 02] BEMPORAD A., MORARI M., DUA V., PISTIKOPOULOS E. N., « The explicit linear quadratic regulator for constrained systems », *Automatica*, vol. 38, p. 3–20, 2002.

[HEE 01] HEEMELS W. P. M. H., DE SCHUTTER B., BEMPORAD A., « Equivalence of Hybrid Dynamical Models », *Automatica*, vol. 37, n°7, p. 1085-1091, 2001.

[LEI 05a] LEIRENS S., Approche hybride pour la commande prédictive en tension d'un réseau d'énergie électrique, PhD thesis, Université de Rennes I–Supélec, Rennes, France, 2005.

[LEI 05b] LEIRENS S., BUISSON J., BASTARD P., COULLON J.-L., « An Efficient Algorithm for Solving Model Predictive Control of Switched Affine Systems », *Proceedings of the Scientific Computation, Applied Mathematics and Simulation World Congress*, Paris, France, 2005.

[LEI 05c] LEIRENS S., BUISSON J., BASTARD P., COULLON J.-L., « A Hybrid Approach for Voltage Stability of Power Systems », *Proceedings of the Power Systems Computation Conference*, Liège, Belgium, 2005.

[LUN 98] LUNZE J., Laboratory Three Tanks System Benchmark for the Reconfiguration Problem, Rapport, Technical University of Hamburg-Harburg, Institute of Control Engineering, Hamburg, Germany, 1998.

[MAY 00] MAYNE D. Q., RAWLINGS J. B., RAO C. V., SCOCAERT P. O. M., « Constrained Model Predictive Control : Stability and Optimality », *Automatica*, vol. 36, n°6, p. 789–814, 2000.

[MIG 02] MIGNONE D., Control and Estimation of Hybrid Systems with Mathematical Optimization, PhD thesis, Swiss Federal Institute of Technology (ETH), Zürich, 2002.

[PEñ 03] PEÑA M., CAMACHO E. F., PIÑÓN S., « Hybrid Systems for Solving Model Predictive Control of Piecewise Affine System », *Proceedings of the IFAC Conference on Analysis and Design of Hybrid Systems*, Saint-Malo, France, 2003.

[THO 04a] THOMAS J., Estimation et Commande Prédictive à Horizon Glissant de Systèmes Hybrides, PhD thesis, Université Paris XI–Supélec, Gif-Sur-Yvette, France, 2004.

[THO 04b] THOMAS J., BUISSON J., DUMUR D., GUÉGUEN H., « Model predictive control for hybrid systems under a state partition based MLD approach (SPMLD) », *Proceedings of the International Conference on Informatics in Control, Automation and Robotics*, Setúbal, Portugal, 2004.

[TOR 02] TORRISI F., BEMPORAD A., MIGNONE D., Hysdel - A Tool for Generating Hybrid Models, Rapport n°AUT00-03, Automatic Control Laboratory, ETH, Zürich, Switzerland, 2002.

[ZAY 01] ZAYTOON J., *Systèmes Dynamiques Hybrides*, Traité IC2 - Systèmes Automatisés, Hermès, 2001, ISBN 2-7462-0247-6.