UNIVERSITÉ DE NANTES

ÉCOLE DOCTORALE

SCIENCES ET TECHNOLOGIES
DE L'INFORMATION ET DES MATÉRIAUX

Année : 2006

Thèse de Doctorat

Diplôme délivré par l'Université de Nantes

Spécialité : AUTOMATIQUE ET INFORMATIQUE APPLIQUÉE

Présentée et soutenue publiquement par :

Thibault HILAIRE

le 8 juin 2006
à l'École Centrale de Nantes

Analyse et synthèse de l'implémentation de lois de contrôle-commande en précision finie

— Étude dans le cadre des applications automobiles sur calculateur embarqué —

JURY

Président du Jury	O. SENTIEYS	*Professeur, Université de Rennes (Enssat)*
Rapporteurs	D. ALAZARD	*Professeur, École nationale supérieure de l'aéronautique et de l'espace*
	J. WHIDBORNE	*Senior Lecturer, Cranfield University*
Examinateurs	F. AÏOUN	*Docteur-ingénieur, PSA Peugeot-Citroën*
	P. CHEVREL	*Professeur, École des mines de Nantes*
	Y. TRINQUET	*Professeur, Université de Nantes (IUT de Nantes)*
Invité	J-P. CLAUZEL	*Ingénieur, PSA Peugeot-Citroën*

Directeur de thèse : Yvon Trinquet
Laboratoire : IRCCyN
Co-directeur : Philippe Chevrel
Laboratoire : IRCCyN / EMN

Institut de Recherche en Communications et en Cybernétique de Nantes **N° ED : 366-258**

Je tiens tout d'abord à remercier Daniel Alazard, professeur à Supaéro, d'avoir accepté de rapporter ma thèse et de m'avoir fait part de ses remarques pour améliorer mon manuscrit. Je remercie aussi James Whidborne, Senior Lecturer à Cranfield University, qui m'a fait l'honneur d'accepter d'être rapporteur d'une thèse rédigée dans une langue qu'il maîtrise mais n'est pas la sienne, sur un sujet qu'il connaît particulièrement bien. J'ai particulièrement apprécié les conversations scientifiques nourries qui s'en sont suivies, sur la thématique *Précision Finie* et sur les perspectives intéressantes que l'on peut mener à la suite de ces travaux.

Je remercie aussi Olivier Sentieys, professeur à l'ENSSAT, d'avoir présidé mon jury de thèse et de m'avoir proposé de collaborer avec lui par la suite.

Je remercie aussi François Aïoun (représenté par Salim Benbouzid) et Jean-Philippe Clauzel d'avoir fait partie de mon jury et d'avoir suivi mon travail chez PSA Peugeot Citroën. Même si les emplois du temps de mes collègues et encadrants furent parfois chargés, l'ambiance de travail fut toujours agréable et détendue (merci genfi). Un remerciement aussi à Damien Lefebvre qui a consacré du temps pour me présenter son régulateur d'agrément de conduite et à tous les gens que j'ai rencontrés là-bas.

J'adresse un vif remerciement à Yvon Trinquet, mon directeur de thèse qui m'a permis de réaliser cette thèse CIFRE sur une thématique double *informatique* et *automatique*. Même si l'aspect *temporel* de l'implémentation n'a pu être développé, il est toujours resté présent et à l'écoute. Enfin, je remercie Philippe Chevrel, mon co-directeur de thèse, de m'avoir accompagné tout au long de cette aventure, d'avoir eu la patience de m'expliquer tous ces points de l'automatique qui m'étaient étrangers (et il y en a eu !) et de m'avoir laissé explorer cette problématique de l'impact *numérique*. J'ai beaucoup apprécié les longs échanges scientifiques, méthodologiques (et même parfois métaphysiques) qui sont survenus tout au long de ces trois années.

Ensuite, je remercie le personnel de l'IRCCyN, de l'École Centrale de Nantes, des Mines de Nantes et de l'IUT que j'ai pu rencontrer depuis 2003, ainsi que la *fine équipe* de doctorants/docteurs de l'IRCCyN, pour tous les repas du midi au RU, pour les longues pauses "gâteaux" à 16h, pour la finesse et la légèreté de nos éclats de rire, pour les sorties, les soirées, et pour les jeudis soirs où s'enchaînaient souvent *baby*, *fées*, *quick*, *absurde* et re-*fées* (comprendre championnat du monde de l'IRCCyN de football de table, collation pour hydrater les sportifs, repas gastronomique, sortie culturelle et intellectuelle, et nouvelle pause hydratation).

Enfin, pour terminer cette longue liste, une mention spéciale à ma famille, mes parents (ma maman qui a eu le courage de tout relire pour traquer les fautes), à GY!BE (et autres) pour avoir accompagné ces longues nuits de rédaction, aux Aventuriers du Rail, à W&G (GPPPPSV) et enfin aux

quelques amis, toujours proches malgré la distance, qui sont restés présents pendant la tempête.

Carrot Power !

À mes grands-parents.

"Calvin&Hobbes" [103]

Table des matières

Table des figures

Liste des tableaux

Notations

$\triangleq$	égal par définition
$\mathbb{R}$, $\mathbb{R}^+$, $\mathbb{R}^k$, $\mathbb{R}^{m\times n}$	respectivement l'ensemble des nombres réels, des nombres réels positifs, des vecteurs des réel $(k \times 1)$ et des matrices réelles $(m \times n)$
$\mathbb{C}$, $\mathbb{C}^k$, $\mathbb{C}^{m\times n}$	l'ensemble des nombres complexes, des vecteurs complexes $(k \times 1)$ et des matrices complexes $(m \times n)$.
$\mathbb{N}, \mathbb{Z}, \mathbb{B}$	respectivement l'ensemble des entiers naturels, des entiers relatifs et l'ensemble des valeurs d'un bit : $\mathbb{B} \triangleq \{0, 1\}$
i	$\sqrt{-1}$ sauf utilisé pour les indices
$[\![a, b]\!]$	ensemble des entiers compris entre a et b $[\![a, b]\!] \triangleq [a, b] \cap \mathbb{N}$
0_k, $0_{m\times n}$, 0	la matrice nulle de dimension $(k\times k)$, $(m\times n)$ ou de dimension définie par le contexte
I_k, $I_{m\times n}$, I	la matrice identité de dimension $(k\times k)$, $(m\times n)$ ou de dimension définie par le contexte
$\mathscr{E}_{p,q}$	la matrice de dimension $p \times q$ dont tous les éléments valent 1
$(M)_{i,j}$	l'élément (i, j) de la matrice M (i^{e} ligne, j^{e} colonne)

$M_{i,\bullet}, M_{\bullet,j}$	respectivement la i^{e} colonne de M et la j^{e} ligne de M
$M^\top$	la transposée de la matrice M
M^*	le conjugué de la matrice M
M^H	le transconjugué de la matrice M $M^H \triangleq (M^*)^\top$
$tr(M)$	la trace de la matrice M
$\lfloor x \rfloor, \lceil x \rceil, \lfloor x \rceil$	l'arrondi de x à l'entier inférieur, supérieur ou plus proche
C_n^p	Le nombre de combinaisons de p éléments parmi n $(n \geqslant p)$ $$C_n^p \triangleq \frac{n!}{p!(n-p)!}$$
$\|M\|_F$	la norme de Frobenius de la matrice M $$\|M\|_F \triangleq \sqrt{\sum_{i,j} \|M_{i,j}\|^2}$$
$\|M\|_{\max}$	la plus grande valeur absolue de M $$\|M\|_{\max} \triangleq \max_{i,j} \|M_{i,j}\|$$
$A \otimes B$	le produit de Kronecker de deux matrices (voir définition A.1) $$A \otimes B \triangleq \begin{pmatrix} a_{1,1}B & \cdots & a_{1,n}B \\ & & \vdots \\ a_{m,1} & & a_{m,n}B \end{pmatrix}$$
$\text{Vec}(M)$	l'opérateur Vec est l'opérateur qui transforme une matrice en un vecteur colonne (voir définition A.2)
$A \times B$	le produit direct de deux matrices A et B de même dimension (dit *produit de Schur* ou *produit d'Hadamard*) $$A \times B \triangleq \begin{pmatrix} a_{1,1}b_{1,1} & \cdots & a_{1,n}b_{1,n} \\ & & \vdots \\ a_{m,1}b_{m,1} & & a_{m,n}b_{m,n} \end{pmatrix}$$

$A \circledast B$

$$A \circledast B \triangleq \mathrm{Vec}(A).\left(\mathrm{Vec}\left(B^{\top}\right)\right)^{\top}$$

(voir définition 4.9 et annexe A)

Introduction

Ce mémoire présente une synthèse de mes travaux de thèse effectués à l'Institut de Recherche en Communications et en Cybernétique de Nantes (IRCCyN) et au service Conception Logiciel Organe (CLO) de PSA Peugeot Citroën (thèse CIFRE).

Ces travaux ont porté sur l'implémentation[1] de lois de contrôle/commande sous les contraintes de précision finie : en effet, les ressources et puissances limitées des calculateurs PSA embarqués imposent que les calculs réalisés, le plus souvent en virgule fixe, soient de précision limitée.
Les lois considérées proviennent de l'automatique ou du traitement de signal, et nous nous restreindrons aux formes linéaires à paramètres constants. Le processus d'implémentation – le passage de la loi mathématique, dont les performances ont été validées en simulation avec des calculs numériquement valables, à un code logiciel pour une cible particulière – amène de nombreuses dégradations de la loi, tant numériques que temporelles.
Les dégradations numériques, auxquelles nous nous intéressons ici, ont principalement deux origines :

- les bruits de quantification dans les calculs (assimilables à des bruits numériques) que l'on retrouve à chaque itération ; ils dépendent de la précision affectée à chaque calcul, et leur impact découle de la sensibilité de la loi aux bruits ;
- la quantification des coefficients intervenant dans les calculs (assimi-

[1] Même si certains considèrent *implémentation* comme un anglicisme, en lui préférant *implantation*, nous faisons la distinction entre ces deux termes : implanter signifie "porter un algorithme", le programmer, alors qu'implémenter ajoute, à notre sens, la possibilité d'adaptation, de reformulation de l'algorithme, comme nous le verrons tout au long de ce mémoire.

lables à des erreurs paramétriques) : cette quantification dépend du nombre de bits et du format requis pour stocker ces paramètres, et leur impact du choix de la réalisation.

Nous nous intéresserons plus particulièrement à l'impact de la quantification.

De plus, pour une loi (filtre ou régulateur) donnée, il existe une infinité de réalisations numériques possibles qui, bien que mathématiquement équivalentes, ne le sont plus en précision finie. Un exemple bien connu est celui de la réalisation dans l'espace d'état, soit la réalisation paramétrée par les matrices (A, B, C, D), exprimée par la relation

$$\begin{cases} X(k+1) &= AX(k) + BU(k) \\ Y(k) &= CX(k) + DU(k) \end{cases}$$

Toute réalisation $(T^{-1}AT, T^{-1}B, CT, D)$ avec T non singulière lui est mathématiquement équivalente, mais présente une sensibilité aux troncatures potentiellement différente. Une forme compagne, par exemple, est plus mal conditionnée qu'une forme équilibrée.

De plus, de nombreuses autres réalisations existent : réalisations en δ, en treillis, structures retour d'état/observateur, décompositions en cascade, en parallèle, etc.

Ce mémoire de thèse, après avoir présenté toutes ces possibilités, propose un formalisme mathématique – la forme implicite spécialisée – représentatif du code réalisé, permettant de décrire, de manière unifiée, un ensemble élargi d'implémentations. Celui-ci, bien que macroscopique, permet d'exprimer précisément les calculs à réaliser et les paramètres réellement mis en jeu. Différentes mesures appliquées à ce formalisme, permettant d'évaluer l'impact de la quantification (sous différentes hypothèses de virgule fixe et virgule flottante) et d'analyser la dégradation induite, sont ensuite proposées. Elles permettent, par exemple, d'évaluer la sensibilité de la fonction de transfert en fonction de la réalisation choisie, ou encore la sensibilité des pôles (que l'on peut relier au nombre de bits minimum nécessaires pour représenter les coefficients sans perdre la stabilité, lors d'une régulation).

Toutes les réalisations n'étant pas de robustesse égale vis-à-vis de la quantification, il appartiendra à ces mesures de les différencier en permettant d'analyser l'impact *a priori* de l'implémentation. Il est ainsi possible de poser un problème de synthèse de réalisation numérique, et de rechercher, selon différents critères, la ou les réalisations qui minimisent, pour une loi donnée, ces mesures de dégradation et qui présentent donc la meilleure robustesse face aux détériorations induites par les processus d'implémentation en précision finie. De nombreux exemples éclaireront la démarche.

Ce mémoire est divisé en six parties :

– le chapitre 1 présente le contexte automobile prescripteur de cette étude et situe la problématique ;

- le chapitre 2, quant à lui, montre l'étendue des possibilités de réalisations de lois, tant au niveau *automatique* qu'au niveau *logiciel*. Nous rappellerons tout d'abord les réalisations possibles les plus couramment utilisées : espace d'état, formes directes, réalisations en δ, etc. Nous indiquerons ensuite les possibilités d'aménagement d'algorithmes, principalement en terme de représentation des nombres et de leur impact ;
- le formalisme unificateur de la forme implicite spécialisée est exhibé au chapitre 3 autour de nombreux exemples. Les classes de réalisations équivalentes exprimées dans cette forme sont construites autour du *Principe d'Inclusion*. De plus, une représentation unifiée des nombres réels dans les calculateurs (virgule fixe, virgule flottante, ...) est développée ;
- le chapitre 4 présente des outils d'analyse permettant d'évaluer numériquement certaines propriétés (robustesse, coût de calcul, ...) de telle ou telle réalisation. Ces indicateurs nous aideront à distinguer des réalisations ;
- enfin, le chapitre 5 permet d'élaborer une synthèse de loi selon ces critères d'analyse en procédant à la recherche de réalisations optimales.

Pour terminer, nous conclurons par les perspectives possibles et les voies à suivre pour poursuivre le travail réalisé.

Ce travail a fait l'objet de quatre publications internationales :

- [44] Implicit state-space representation : a unifying framework for FWL implementation of LTI systems. In *IFAC05 World Congress*, Juillet 2005.
- [43] Designing low parametric sensitivity FWL realizations of LTI controllers/filters within the implicit state-space framework. In *CDC-ECC'05*, Décembre 2005.
- [40] Low parametric sensitivity realization design for FWL implementation of MIMO controllers : Theory and application to the active control of vehicle longitudinal oscillations. In *CAO'O6*, Avril 2006.
- [41] Pole sensitivity stability related measure of FWL realization with the implicit state-space formalism. In *Proc. ROCOND'06*, Juillet 2006.

Chapitre

1 Contexte et problématique

Résumé :
Ce premier chapitre resitue le domaine d'étude de cette thèse dans son contexte industriel – l'embarqué automobile – et expose la problématique telle qu'elle s'est présentée au début de ces travaux.
Le processus d'implémentation, qui vise à transformer une loi en un code logiciel s'exécutant sur un calculateur embarqué entraîne une modification de la loi, et nous listons ici les contraintes de ce processus ainsi que les sources de dégradations induites.

Sommaire

1.1 L'*embarqué* en automobile

1.1.1 Contexte automobile

Le domaine de l'automobile a beaucoup évolué ces trente dernières années. Initialement purement mécaniques, les fonctionnalités d'une automobile ont, depuis lors, intégré en leur sein de plus en plus d'électronique, puis d'informatique.
Cette informatique embarquée a désormais pour mission de mettre en œuvre des *lois* permettant d'améliorer, de manière générale, les performances d'une automobile (augmentation de la sécurité, du confort, diminution de la consommation, augmentation de l'efficacité des systèmes mécaniques, etc.). L'automatique est à la base du développement de ces nouvelles fonctionnalités.
A partir d'une modélisation des processus physiques (ainsi que des dispersions associées), des *lois de contrôle/commande* peuvent être conçues afin de piloter au mieux ces processus. Une partie du logiciel embarqué met en œuvre de telles lois, le reste s'attachant à gérer les différents modes de fonctionnement, la sécurité d'ensemble, ainsi qu'à assurer l'aide aux diagnostics.

L'informatique embarquée dans l'automobile poursuit son essor. Les nouvelles normes anti-pollution, les demandes du public en terme de sécurité et de confort, et la montée en puissance des technologies électroniques sont en train de transformer la conception automobile. Si l'électronique représente aujourd'hui de 20% à 25% du coût total du véhicule, cette proportion atteindra près de 40% d'ici 2010 [2]. L'électronique s'avère nécessaire pour contrôler toute la dynamique du véhicule (contrôle moteur, boîte de vitesses , suspension, direction, ABS, ...), les fonctions de carrosserie (phares, essuie-glaces, portes, ...) et les fonctions de confort de l'habitacle (affichage, autoradio, climatisation, ...) ; l'ensemble est coordonné par des superviseurs. En tout, on peut compter jusqu'à une quarantaine de calculateurs différents dans une même voiture. Ce nombre est toutefois à la baisse chez certains constructeurs, dont PSA, car les fonctionnalités sont petit à petit regroupées au sein d'un nombre réduit de calculateurs.
Si le développement de l'électronique est important, c'est bien le logiciel (et non la puissance de calcul) qui constitue l'élément stratégique. Son rôle est primordial : il détermine le comportement du calculateur ou du superviseur auquel il est associé. Comme les fonctions électroniques traitent de plus en plus d'informations, le logiciel s'étoffe pour prendre en compte toutes les configurations possibles. En 1980, la CX Citroën embarquait 1.1Ko, tandis qu'en 2000, la 607 embarque 2Mo. L'automobile suit un rythme de progression du logiciel identique à celui de l'avionique avec un décalage d'une décennie environ.

Les fonctions implémentées dans les calculateurs sont de plus en plus complexes, comme nous allons le voir au paragraphe suivant.

1.1.2 Les différentes lois embarquées

Nous n'allons pas détailler ici toutes les stratégies de contrôle-commande existantes (ou à venir), ni les multiples champs d'applications de la commande dans le domaine automobile. On pourra trouver dans [3, 90], notamment, de nombreux exemples détaillés.

Les deux grands domaines que l'automatique a principalement investis sont le contrôle du groupe motopropulseur et le contrôle dynamique de châssis.

Le groupe motopropulseur, appelé GMP, est composé de l'ensemble des éléments mécaniques permettant de produire le couple pour faire avancer le véhicule : producteurs de couple (comme le moteur thermique ou électrique), de limitateur de couple (embrayage, etc.) et de démultiplicateur (boîte de vitesses, etc.) [64]. Le contrôle moteur a pour mission de gérer le moteur afin de satisfaire au mieux la fonction de production d'énergie : il pilote les organes électriques commandables du moteur (injection, allumage pour moteur essence, pompe et vanne pour moteur diesel, etc.) afin que celui-ci fournisse la puissance demandée par le conducteur via la pédale d'accélérateur. De plus, le contrôle moteur devra tenir compte au mieux des demandes du conducteur, des contraintes environnementales et des autres périphériques (actifs ou passifs) auquel le calculateur est relié, tels que le contrôle actif de stabilité (ESP[1]), le système de freinage piloté (ABS[2]), la direction assistée, etc. La figure 1.1 résume les différents éléments fonctionnels d'un contrôle moteur essence classique (on trouvera plus de détails dans [3]).

Le contrôle dynamique du châssis vise à maîtriser le comportement longitudinal et latéral du véhicule en fonction des *desiderata* du conducteur et du comportement routier de la voiture. En agissant par application de forces sur un ou plusieurs pieds de roues, et en maîtrisant/connaissant l'interface roue-sol, la suspension, le freinage, la direction, il est possible de contrôler la dynamique longitudinale (l'ABS, le contrôle anti-glissement à l'accélération TCS[3], etc.), la dynamique transversale (direction assistée), la dynamique verticale (suspension active) ou encore une combinaison de ces dynamiques (contrôle du lacet, contrôle anti-roulis, etc.).

1.2 Méthodologie de conception

La conception des lois de contrôle/commande (réalisée par *l'Automaticien*) et le développement logiciel (réalisée par *l'Informaticien*) suivent

[1] *Electronic Stability Program.*
[2] *Antilock Braking System.*
[3] *Traction Control System.*

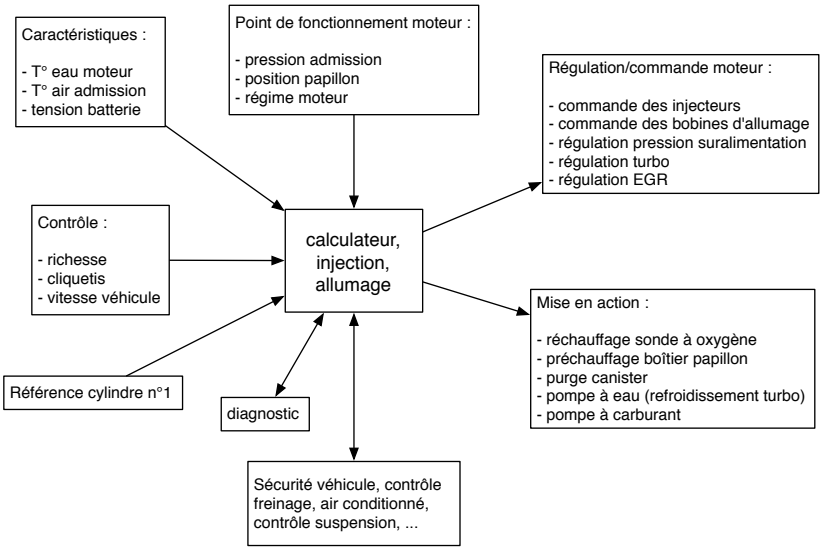


FIG. 1.1 – Entrées/Sorties et fonctionnalités d'un calculateur série de moteur essence

tous deux une méthodologie bien précise. Leur cycle de développement est illustré, classiquement, par la figure 1.2. Il prévoit plusieurs étapes clés

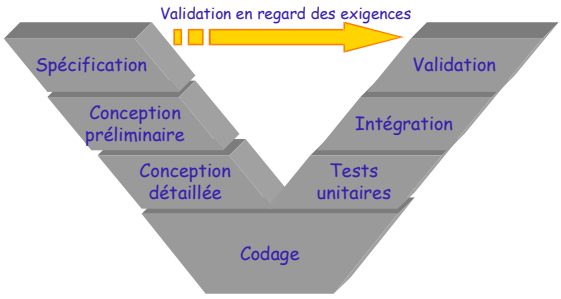


FIG. 1.2 – Cycle de développement

indispensables (dans l'ordre chronologique) :

- la spécification ;
- la conception ;
- la réalisation ;
- l'intégration ;
- la validation.

Lors des deux premières étapes, des spécifications techniques (spécifications techniques de besoins (STB), spécifications techniques générales (STG), spécifications techniques détaillées (STD), spécifications techniques de réalisation (STR)) sont élaborés pour permettre de passer d'une étape à une autre, ainsi que des plans de tests afin de valider chaque transition. Ils garantissent la bonne adéquation du résultat avec le besoin exprimé en entrée du cycle en V.
Ce cycle en V assure aussi une bonne maîtrise du développement et une traçabilité efficace entre les différentes phases. Toutefois, ce cycle peut être remis en cause, et un cycle de développement itératif, tel que celui présenté dans [59], peut dans certains cas être utilisé afin de réduire les allers-retours entre chaque étape, avec une plus grande souplesse et rapidité d'exécution.

En entrée du cycle de développement logiciel, un cahier des charges *fonctionnel* et *opérationnel* est élaboré à partir d'un modèle de la loi de contrôle/commande ; les contraintes matérielles et les contraintes de sûreté de fonctionnement s'y ajoutent.
Il est alors nécessaire d'insérer une étape entre la partie *Conception* et la partie *Implémentation* pour permettre la transcription des exigences de l'*Automaticien* en spécifications pour l'*Informaticien*. Cela passe, en particulier, par la définition de critères d'acceptation, par le choix des réalisations (paramétrisations) et par une analyse numérique (mesure de l'impact numérique de cette implémentation).

D'une manière générale, on dissociera les lois de commande décrites sous forme d'automates d'états (les automates, en nombre fini, représentant généralement des comportements différents pour une même loi, tels que les modes dégradés, etc. ; et les transitions entre ces états représentant les événements discrets provenant du conducteur ou de facteurs externes) des lois du domaine de l'automatique ou du signal. On ne considérera ici que des lois linéaires, à paramètres constants.
Pour les lois nécessitant l'évaluation de fonctions complexes en différents points (comme par exemple l'évaluation d'un sinus ou de toute autre fonction un tant soit peu complexe dont l'évaluation de la description mathématique nécessiterait un calcul coûteux), on se reportera, par exemple, aux les travaux de J.M. Muller sur les méthodes à base de table que sont les *cartographies*. Plusieurs implémentations possibles existent, permettant différents compromis autour de la taille mémoire utilisée, du coût de calcul ou encore de la précision du résultat, telles que les interpolations linéaires ou polynomiales par morceaux, les *bi-partite* ou *multipartite methods*, etc. [79, 81], [87], [93], ...

Nous nous focaliserons sur cette étape de transition et supposerons qu'une loi de commande *valide* a été préalablement conçue, quelle que soit la méthode de synthèse utilisée, et l'utiliserons comme loi de référence à implé-

menter, c.-à-d. à réaliser en logiciel sur une cible particulière.

1.3 Problématique

Afin de cerner plus précisément la problématique de la thèse, et de la centrer dans son contexte industriel, un rapport interne PSA [37] a été rédigé en début de mes travaux. Il a permis, lors d'échanges avec les différents acteurs impliqués, de lister les contraintes dues à l'implémentation et de définir les sources de dégradation qu'elle entraîne.

1.3.1 Exemple

Il peut être intéressant d'évoquer les problèmes de l'implémentation par un exemple PSA, révélateur de la problématique générale : l'estimateur d'état de charge de batterie.
Cette fonctionnalité a été conçue par PSA [85] et a pour but d'estimer, en cours de fonctionnement, l'état de charge d'une batterie automobile. Un modèle de cet estimateur a été construit sous Matlab/Simulink, puis testé en simulation et en conditions réelles (sur plusieurs batteries). Il a donné des résultats très convaincants.
L'implémentation de cette fonctionnalité au sein d'un calculateur PSA a posé davantage de problèmes : ce calculateur ne possédant pas d'unité de calcul flottant, il ne pouvait donc pas réaliser, tel quel, certains des calculs présents dans les *planches* Simulink (calculs matriciels, exponentiels, etc.). De plus, les calculs ne pouvant être réalisés qu'avec des nombres entiers, les coefficients du modèle devaient être quantifiés, induisant un impact non négligeable sur le comportement de la loi.
Une simplification du modèle a alors été effectuée, en deux grandes étapes :

- simplification des calculs, utilisation de tables (cartographies) pour pouvoir réaliser les opérations complexes ;
- une quantification des coefficients et des calculs, afin de ne faire intervenir que des calculs entiers sur 16 bits (réalisables sur le calculateur)

Au fur et à mesure de ces étapes, a eu lieu une comparaison avec le modèle initial, servant alors de référence : l'impact de chaque modification était alors étudié pour voir dans quelle mesure il entraînait une dégradation et dans quelle mesure celle-ci était acceptable.

Après cette étape de *reformulation*, la nouvelle réalisation de la fonction estimateur d'état de charge de batterie a pu finalement être implémentée facilement. Le module logiciel correspondant accomplit la fonctionnalité attendue, avec la précision requise. Dans ce cas, l'implémentation a entraîné une dégradation des résultats, mais cette dégradation a pu être maîtrisée et reste inférieure à la tolérance que l'on pouvait admettre.

Cela dit, ces problèmes récurrents se posent à chaque nouvelle implémentation et montrent la nécessité d'étudier des outils et des méthodes adaptés à cette problématique.

1.3.2 Contraintes

Les problèmes d'implémentation sont principalement dus à la différence de *puissance* (nous allons la détailler après) qui existe entre les ordinateurs (de bureau), sur lesquels sont conçues et testées les lois de commande et les calculateurs embarqués de série. Les contraintes, liées à la plate-forme de calcul – le calculateur, le système d'exploitation et l'application – sont principalement de deux natures : numériques et temporelles.

Précisément, les contraintes matérielles (c.-à-d. imposées par le choix du calculateur) dépendent des points suivants :

- les ressources matérielles du calculateur conditionnent les possibilités de calcul *hardware* : seules les opérations "natives" (calcul entier sur un processeur ne possédant que des unités de calcul entier, calcul entier et flottant sur un processeur avec unités de calcul flottant, etc.) seront possibles sans avoir recours à une émulation (par exemple, il est toujours possible d'émuler des opérations de calcul flottant sur un processeur ne disposant pas d'unité de calcul flottant, mais cela se fait au détriment d'un temps de calcul très important) ;
- les fonctions mathématiques complexes (trigonométrie, exponentiels, etc.) ne pouvant être réalisées par les ressources matérielles devront l'être en logiciel, ce qui nécessite la disponibilité de bibliothèques mathématiques adaptées au processeur cible pour être efficaces ;
- la taille du code de la ROM et de la RAM impliquent une taille maximale pour le code et pour les données (comme les cartographies) stockées ;
- certains calculateurs peuvent posséder des fonctions spécifiques réalisées matériellement (au lieu d'une réalisation logicielle, parfois émulée sur d'autres calculateurs) : unités de calcul spécialisées (unités MAC comparable à celles de DSP qui effectue des calculs de traitement du signal), unités réalisant des fonctionnalités spécifiques au contrôle moteur (par exemple, un module de calcul d'angle de came, disponibles en *matériel* sur certains calculateurs, alors que cette fonction n'est réalisée qu'en logiciel ailleurs), etc.

Pour le système d'exploitation, on notera que :

- le caractère *temps réel*[4] de l'exécution implique une réactivité im-

[4]"*Est qualifié de temps réel le comportement d'un système informatique lorsqu'il est assujetti à l'évolution dynamique d'un procédé qui lui est connecté, et qu'il doit piloter ou*

portante : la fréquence d'échantillonnage impose les temps de calculs maximum pour chaque loi ;
- la communication réseau avec d'autres calculateurs implique des contraintes temporelles : le calculateur est susceptible de recevoir des informations provenant d'autres calculateurs : les valeurs attendues pour le calcul de la loi peuvent ne pas arriver à temps (ou ne pas arriver du tout) ;
- le processeur n'est pas dédié à 100% à la réalisation de la loi : le logiciel de base peut utiliser de 10 à 20% du temps de calcul, et d'autres fonctionnalités sont très souvent implémentées sur le même calculateur.

Enfin, l'application réalisant numériquement la loi amène aussi quelques contraintes :
- les contraintes logicielles sont surtout d'ordre méthodologique : le code développé doit pouvoir être réutilisé facilement. Il doit donc être lisible (un code performant, car astucieux, pourra ne pas être retenu, s'il est difficile à comprendre), documenté, découpé en modules. Le code doit, également, être suffisamment portable pour être utilisé avec d'autres processeurs ;
- Le temps consacré au développement logiciel est relativement réduit : il n'est donc pas souvent possible de ré-optimiser, a posteriori, le logiciel, de chercher quels procédés algorithmiques, ou techniques, pourraient améliorer sensiblement son temps d'exécution. Une fois opérationnels et validés, il n'est pas toujours possible de peaufiner certains points de programmation, faute de temps.

1.3.3 L'implémentation entraîne une dégradation

Dans la majorité des cas, l'implémentation d'une loi de contrôle/commande sur un calculateur , si puissant soit-il, entraîne une dégradation de cette loi. En effet, le caractère fini d'un calculateur ne permet pas la représentation, le calcul, la manipulation des nombres transcendants, comme le sont la plupart des réels. Comme nous le verrons, seuls certains ensembles de nombres (les entiers, les rationnels, ...) peuvent être manipulés avec une précision infinie (avec une programmation adaptée).
De plus, les temps de calcul, souvent variants, entraînent des gigues entre l'acquisition des données en entrée et la production des sorties.

De par la nature de l'implémentation – calculs qui peuvent être inexacts et temps de calcul non nuls ou non maîtrisés – l'exécution d'une loi de commande réalisée par un algorithme sur un calculateur série peut différer

suivre en réagissant *à tous ses changements d'état*" d'après [29].

(et de beaucoup[5]) du comportement de référence.

1.3.4 Les sources de dégradation

Il peut être intéressant de pouvoir lister les sources de dégradations induites par le processus d'implémentation, afin de les analyser plus finement et de les prendre en considération.

Nous distinguerons les sources de dégradations selon leur nature : numérique ou temporelle.

La dégradation est de nature temporelle lorsqu'un des éléments qui réalise la loi de contrôle/commande amène un retard temporel sur la sortie. Ces retards peuvent être dus à :

- un délai opératoire : le temps d'exécution d'un processus n'est pas nul (temps de calcul, temps de recherche d'une valeur dans une cartographie, etc.) ;
- une donnée d'entrée nécessaire qui est véhiculée par le réseau mais qui n'arrive pas à temps (on ne peut connaître a priori le temps maximal mis par une donnée pour arriver à son destinataire, à cause des collisions entre trames, les priorités, etc., qui peuvent retarder l'envoi d'une donnée) ;
- au logiciel de base (qui comprend l'OS, les drivers de communication, les mécanismes de sûreté de fonctionnement, etc.) : celui-ci peut interrompre l'exécution logique du calcul de la loi pour effectuer une tâche prioritaire (traiter une valeur réseau, un ordre important, etc.).

La dégradation est de nature numérique lorsque le résultat d'un calcul n'est pas celui attendu. Les dégradations numériques consécutives à la réalisation d'une loi de contrôle/commande sont de plusieurs types [31, 111] :

- les erreurs paramétriques : elles sont dues à la quantification des coefficients de la loi. Les coefficients ne peuvent pas être exprimés de manière exacte dans le calculateur (lorsque celui-ci ne possède pas la même précision, en terme de représentation que l'ordinateur sur lequel a été conçue la loi), la loi en elle-même est changée. C'est alors une autre loi (espérée suffisamment proche), qui est implémentée. Parfois, les pôles d'une loi peuvent être déplacés, en dehors même du cercle de stabilité. Il est donc préférable, que la loi soit peu sensible à la modification de ses paramètres (sensibilité paramétrique) ;
- les erreurs d'arrondi lors des calculs (quantification des résultats de calculs intermédiaires).

[5] On se référera à l'exemple de la section 4.4.5 qui montre un régulateur dont la stabilité n'est plus assurée dès que les coefficients qui le représentent sont un tant soit peu tronqués pour être utilisés dans un calculateur.

Nous nous intéresserons plus particulièrement, dans cette thèse, aux erreurs paramétriques et à leur impact.

1.4 Conclusion

Ce court chapitre de présentation du contexte et de la problématique nous a permis d'expliciter les contraintes matérielles et logicielles et leur répercussion sur la qualité d'une implémentation, et finalement, de poser le cadre de travail. L'inventaire des sources de dégradation d'une "loi" lors de sa mise en œuvre numérique éclaire sur les critères à prendre en compte afin d'évaluer l'impact du processus d'implémentation.

Chapitre

2 Un large panel d'implémentations

Résumé :
L'objectif de ce chapitre de thèse est de montrer que, pour une loi donnée, un très large panel d'implémentations existe. Nous étudierons tout d'abord les nombreuses possibilités d'implémentation de filtres et/ou régulateurs de type LTI[a], qui amènent à autant d'algorithmes différents. Ensuite, nous étudierons les moyens informatiques pour représenter un nombre (représentation en entier, virgule fixe, virgule flottante) ainsi que les impacts numériques que cela peut avoir sur les calculs, et donc sur les algorithmes de mise en œuvre de lois de contrôle/commande.

[a] *Linear Time Invariant* : filtre/régulateur linéaire à coefficients invariants dans le temps.

Sommaire

2.1 Différentes paramétrisations possibles

Pour une loi de contrôle-commande donnée, qu'il s'agisse d'un filtre ou d'un contrôleur intervenant dans une régulation ou un asservissement, il existe classiquement de nombreuses réalisations numériques possibles, une infinité en fait. Nous allons ici détailler les plus connues. Pour une loi donnée, celles-ci sont bien évidemment mathématiquement équivalentes, mais ne le sont plus dès que l'on passe en précision finie.
Nous considérerons aussi bien les filtres/régulateurs SISO[1] que MIMO[2].
Ces différentes possibilités d'implémentation découlent de deux approches différentes : celle du traitement du signal, précurseur dans le domaine, mais sur un champ restreint, et celle de l'automatique.
Les lois considérées peuvent provenir de synthèse de filtres (filtres RIF[3] ou RII[4] : ils sont définis par leurs paramètres intrinsèques ou par leur fonction de transfert), de régulateurs, d'observateurs, etc.
Nous verrons, de manière entremêlée, des implémentations découlant d'une approche entrée(s)/sortie(s), comme les formes directes ou en treillis, ainsi que des implémentations explicitant l'*état* interne du système considéré (approche classique en automatique pour des problèmes de régulation ou de poursuite de trajectoire). Enfin nous verrons différents opérateurs qui modifient la paramétrisation mise en jeu, ainsi que l'utilisation de décompositions de lois en lois élémentaires.

2.1.1 Graphe de fluence

On a souvent l'habitude, et c'est particulièrement vrai en traitement du signal, de décrire une réalisation de loi sous la forme d'un graphe de calcul, d'un diagramme, détaillant les interconnexions entre les différents opérateurs agissant sur les entrées et les états[5]. Dans le cas des systèmes linéaires, on peut se limiter aux opérateurs d'addition/soustraction, à la multiplication par une constante (scalaire ou matricielle) et à l'opérateur retard q^{-1} (cf. figure 2.1), qui consiste à mémoriser une valeur puis à la restituer au pas d'échantillonnage suivant.
La notion de *graphe de fluence* est alors utilisable pour représenter un algorithme de loi de contrôle-commande. Il s'agit d'un graphe pour lequel les noeuds représentent les étapes de calcul et pour lequel les branches valuées

[1] S*ingle* I*nput* S*ingle* ***O****utput*.
[2] M*ultiple* I*nput* M*ultiple* O*utput*.
[3] Filtre à *R*éponse *I*mpulsionnelle *F*inie : $y(k) = \sum_{i=0}^{q} h_i u(k-i)$.
[4] Filtre à *R*éponse *I*mpulsionnelle *I*nfinie : $y(k) = \sum_{i=0}^{q} b_i u(k-i) - \sum_{i=1}^{p} a_i y(k-i)$.
[5] Simulik est un parfait exemple de logiciel permettant de décrire une loi sous la forme d'un graphe de calculs.

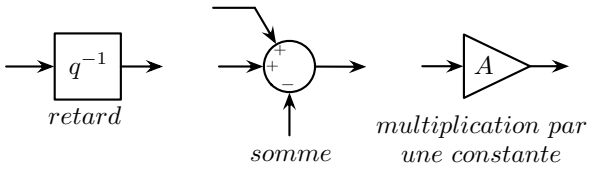


FIG. 2.1 – Les différents opérateurs d'un graphe

et orientées symbolisent une opération (multiplication ou opérateur retard). La convergence de plusieurs branches sur un même noeud représente une addition. Il s'agit donc d'une notation équivalente, mais légèrement différente, à celle présentée à la figure 2.1 et utilisée dans les logiciels de représentation graphique des processus, tels que Simulink. La figure 2.2 est un exemple de graphe de fluence représentant un algorithme d'implémentation d'un filtre du second ordre [28]. Celui-ci caractérise la relation entrée/sortie du système. Il correspond à l'algorithme

1: $X_1(k+1) \leftarrow a_{12}X_2(k) + a_{11}\left(X_1(k) + U(k)\right) + U(k)$
2: $X_2(k+1) \leftarrow a_{22}X_2(k) + a_{21}\left(X_1(k) - U(k)\right)$
3: $Y(k) \leftarrow c_1X_1(k) + c_2X_2(k) + dU(k)$

On remarquera que l'ordre des calculs et le parenthésage sont, en précision finie, très important : cet algorithme n'est pas équivalent, lorsque l'on considère les paramètres utilisés à

1: $X_1(k+1) \leftarrow a_{12}X_2(k) + a_{11}X_1(k) + (a_{11}+1)\,U(k)$
2: $X_2(k+1) \leftarrow a_{22}X_2(k) + a_{21}X_1(k) - a_{21}U(k)$
3: $Y(k) \leftarrow c_1X_1(k) + c_2X_2(k) + dU(k)$

Dans le 2e algorithme, un nouveau coefficient $(a_{11}+1)$ apparaît, qui pourra lui aussi être modifié lors de l'opération de la quantification.
De plus, pour du calcul 2 :

$$X_2(k+1) = a_{22}X_2(k) + a_{21}\left(X_1(k) - U(k)\right) \quad (2.1)$$
$$= a_{22}X_2(k) + a_{21}X_1(k) - a_{21}U(k) \quad (2.2)$$

les valeurs $X(k)$ et $U(k)$ peuvent être additionnées entre elles puis multipliées par a_{11} (1er algorithme équation (2.1)) ou bien multipliées par a_{11} et ensuite additionnées (équation (2.2), 2e algorithme). À chaque étape de calcul peut (mais ce n'est pas obligatoire) intervenir une quantification et

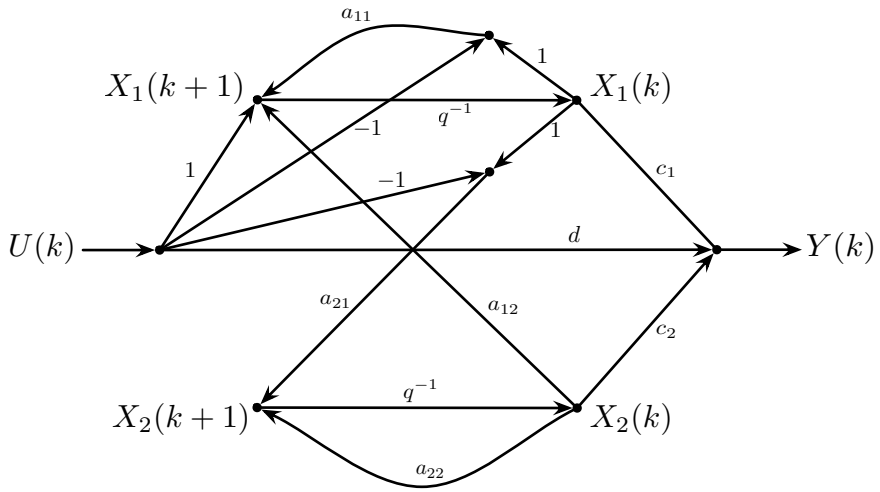


FIG. 2.2 – Un exemple de graphe de fluence d'une réalisation d'un filtre du second ordre

donc un arrondi, et les résultats obtenus à l'aide de ces deux algorithmes risquent d'être, selon l'implémentation, différents.

La théorie des graphes [27] permet de modifier un graphe sans changer le comportement de l'algorithme qu'il décrit et peut être utilisée pour la recherche de formes intéressantes. Les formes transposées (voir 2.1.4.3) en sont un exemple.

2.1.2 Représentation d'*état*

La représentation d'état est couramment utilisée en *automatique* alors que son usage est marginal en *signal*. Elle consiste à choisir un vecteur d'état caractéristique de l'*état* instantané du système considéré et à exprimer l'évolution de cet état à l'aide d'équation différentielles ou récurrentes du 1$^{\text{er}}$ ordre.
Ici, nous ne nous intéresserons qu'aux lois temporellement discrètes : la 1$^{\text{re}}$ relation relie donc l'état à l'instant $k+1$, l'état à l'instant k et l'entrée, tandis que la sortie s'exprime comme une combinaison linéaire des composants de l'état et de l'entrée. Formellement, cela s'écrit

$$\begin{cases} qX(k) &= AX(k) + BU(k) \\ Y(k) &= CX(k) + DU(k) \end{cases} \tag{2.3}$$

où $X(k) \in \mathbb{R}^n$ est le vecteur d'état, $U(k) \in \mathbb{R}^m$ le vecteur d'entrées et $Y(k) \in \mathbb{R}^p$ le vecteur de sorties, à l'instant k. Ces notations seront adoptées

par la suite.
q est l'opérateur *avance* défini par

$$qX(k) \triangleq X(k+1) \tag{2.4}$$

Les matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$ et $D \in \mathbb{R}^{p\times m}$ définissent entièrement ce système (ces matrices sont constantes car il s'agit d'un système LTI ; dans le cas d'un système LPV[6], elles dépendent d'un paramètre θ variant dans le temps). Dans le cas SISO, on a évidemment $p = m = 1$.
L'équation (2.3) peut se représenter sous la forme du graphe 2.3.

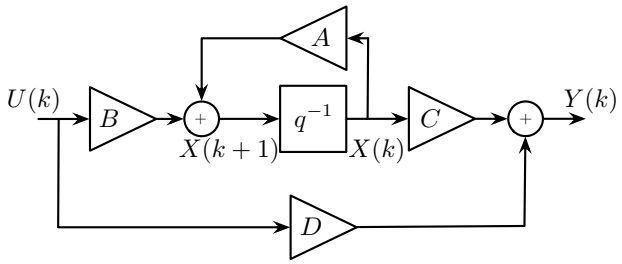


FIG. 2.3 – Système sous forme espace d'état

La fonction de transfert H du système décrit par l'équation (2.3) s'écrit

$$H(z) = C\left(zI_n - A\right)^{-1} B + D \qquad \forall z \in \mathbb{C} \tag{2.5}$$

Il est bien connu qu'il n'y a pas unicité de l'écriture sous forme espace d'état. Ainsi, si l'on considère une matrice $T \in \mathbb{R}^{n\times n}$ non singulière, alors tout système défini par les matrices $(T^{-1}AT, T^{-1}B, CT, D)$ est mathématiquement équivalent au système décrit par (2.3). Les matrices (A, B, C, D) définissent une **paramétrisation** du régulateur considéré. Celui-ci est alors défini par au plus $n^2 + n(p+m) + pm$ paramètres, à comparer aux $n(m+p) + mp$ paramètres nécessaires pour décrire le système sous une forme canonique, comme la fonction de transfert.

Parmi les représentations classiques, on notera les 4 formes canoniques[55] (la forme *Observateur*, *Contrôleur*, *d'Observabilité* et *de Contrôlabilité*) que l'on peut écrire dans le cas SISO, SIMO et MISO (les cas SISO sont traités aux sections 2.1.4.2 et 2.1.4.3).
La forme équilibrée, définie ci-après, est elle aussi souvent utilisée pour son bon conditionnement numérique, comme l'ont montré Gevers et Li [31] (cf. page 124) :

[6] *L*inéaire à *P*aramètres *V*arants.

Définition 2.1 (Grammiens, Forme équilibrée)
On considère une réalisation sous forme espace d'état (A, B, C, D). Les grammiens de commandabilité et d'observabilité sont définis par

$$W_c \triangleq \sum_{k=0}^{\infty} A^k B B^\top \left(A^\top\right)^k \tag{2.6}$$

$$W_o \triangleq \sum_{k=0}^{\infty} \left(A^\top\right)^k C^\top C A^k \tag{2.7}$$

Une réalisation ***équilibrée*** *est une réalisation telle que $W_c = W_o$. Dans ce cas, W_c et W_o sont diagonaux. Pour un système asymptotiquement stable, il est toujours possible, par un changement de variable, d'écrire un système sous forme équilibrée.*

2.1.3 Retour d'état/observateur

On considère un système $\mathcal{P}$ régulé par un contrôleur $\mathcal{C}$, selon la figure 2.4. Soit (A_p, B_p, C_p) une réalisation sous forme d'état du système $\mathcal{P}$, que

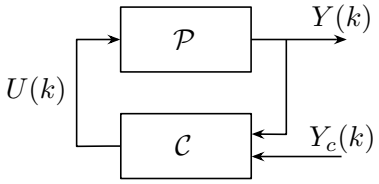


FIG. 2.4 – Le système $\mathcal{P}$ contrôlé par le régulateur $\mathcal{C}$

l'on suppose strictement propre[7] :

$$\mathcal{P} \begin{cases} X(k+1) &= A_p X(k) + B_p E(k) \\ U(k) &= C_p X(k) \end{cases} \tag{2.8}$$

Le régulateur $\mathcal{C}$ est sous forme retour d'état/observateur lorsqu'il est composé d'un observateur qui estime les états du système $\mathcal{P}$ et d'un retour d'état qui élabore la commande à partir de l'état estimé, comme décrit sur la figure 2.5.

L'observateur est décrit par le modèle de représentation (A_p, B_p, C_p) du système $\mathcal{P}$ et d'un gain d'observation K_o. L'équation générale de l'estimation de l'état est

$$\hat{X}(k+1) = A_p \hat{X}(k) + B_p U(k) + K_o \left(Y(k) - C_p \hat{X}(k)\right) \tag{2.9}$$

[7] On se restreint au cas d'un système strictement propre pour alléger l'écriture des équations des formes retours d'état observateur. Pour les équations avec $D_p \neq 0$, on se référera à [4].

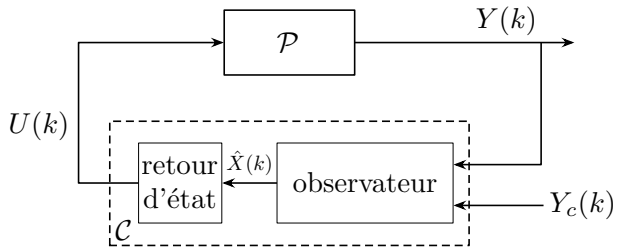


FIG. 2.5 – Un régulateur retour d'état/observateur

$\hat{X}(k)$ représente l'estimée du vecteur d'état du système $\mathcal{P}$. Lorsque l'estimation est parfaite, l'innovation $Y(k) - C_p\hat{X}(k)$ est nulle. La matrice K_o est choisie pour assurer la stabilité de la matrice $A_p - K_oC_p$.
Le retour d'état est défini par le gain de commande K_c. De plus, en cas de suivi de consigne de référence, les signaux $U_{ref}(k)$ et $\hat{X}_{ref}(k)$ sont introduits (pour simplifier on considérera uniquement le cas $U_{ref}(k) = M_1Y_c(k)$ et $\hat{X}_{ref}(k) = M_2Y_c(k)$ avec M_1 et M_2 constants). Le signal de commande est défini par

$$U(k) = K_c\left(\hat{X}(k) - \hat{X}_{ref}(k)\right) + U_{ref}(k) \tag{2.10}$$

La figure 2.6 représente ces équations.

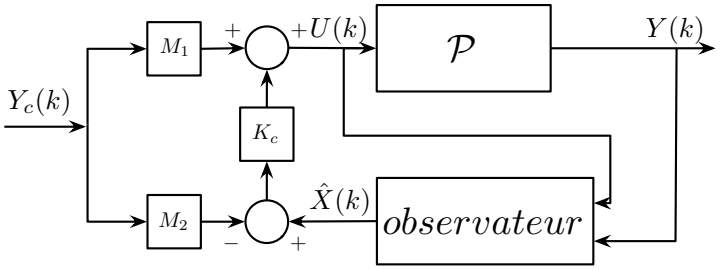


FIG. 2.6 – Retour d'état/observateur avec consignes

L'introduction de l'observateur permet d'estimer le vecteur d'état du système et donc de donner un sens physique à chaque état du régulateur : la compréhension des phénomènes physiques se trouve alors facilitée. Cette structure apporte aussi l'avantage d'autoriser un réglage post-synthèse.

Il est à noter aussi qu'il existe deux écritures de la forme retour d'état observateur en discret, la forme *prédicteur* (équations (2.9) et (2.10)), où, pour calculer et appliquer la commande à l'instant k, on ne connaît pas U_k et la forme *estimateur* (cf. [4]).

2.1.4 Formes Directes

Les formes directes sont des formes classiques d'implémentation des filtres parmi les plus utilisées dans le domaine du traitement du signal. Ces structures canoniques font directement intervenir les coefficients de la fonction de transfert. Nous les présenterons ici sous la forme SISO uniquement (mais elles peuvent être généralisées si nécessaire).
Leur principal avantage, qui justifie leur grande utilisation, est que les algorithmes découlant de ces formes possèdent un minimum de calculs. Malheureusement, comme nous le verrons plus tard, ces formes sont mal conditionnées numériquement (cf. chapitre 5).

2.1.4.1 Forme directe I

C'est la forme la plus naturelle car elle s'obtient en écrivant la relation entrée/sortie de la fonction de transfert exprimée avec l'opérateur retard.
Soit H une fonction de transfert correspondant à la relation entrée/sortie $H(z) = \frac{U(z)}{Y(z)}$ où $U(z)$ et $Y(z)$ sont les transformées en z de l'entrée et la sortie. En écrivant H sous la forme d'une fonction fractionelle en z^{-1} :

$$H(z) = \frac{\sum_{i=0}^{n} b_i z^{-i}}{\sum_{i=0}^{n} a_i z^{-i}} \tag{2.11}$$

il vient

$$Y(k) = \frac{1}{a_0}\left(\sum_{i=0}^{n} b_i U(k-i) - \sum_{i=1}^{n} a_i Y(k-i)\right) \qquad \forall k > n \tag{2.12}$$

On peut aussi décrire cette forme par le graphe 2.7.
Cette équation de récurrence est très souvent utilisée bien que son conditionnement numérique soit mauvais et qu'elle nécessite l'usage de $2n-1$ mémoires alors que le système est d'ordre n. Pour cette dernière raison, la forme directe II lui est préférée.
On remarquera qu'il n'y a pas unicité des coefficients $(a_i)_{0 \leqslant i \leqslant n}$ et $(b_i)_{0 \leqslant i \leqslant n}$ de l'équation (2.11), à moins de poser $a_0 = 1$, ce qui supprime la division par a_0 dans la réalisation décrite par l'équation (2.12) ou le graphe 2.7, mais peut poser des problèmes de représentation lorsque les coefficients ne sont pas représentés en virgule fixe mais seulement avec des entiers (cf. [35, 34]).
Notons que pour $a_0 \neq 1$, cette réalisation ne peut se mettre sous la forme de l'équation (2.3) (cf. section 3.2.2).

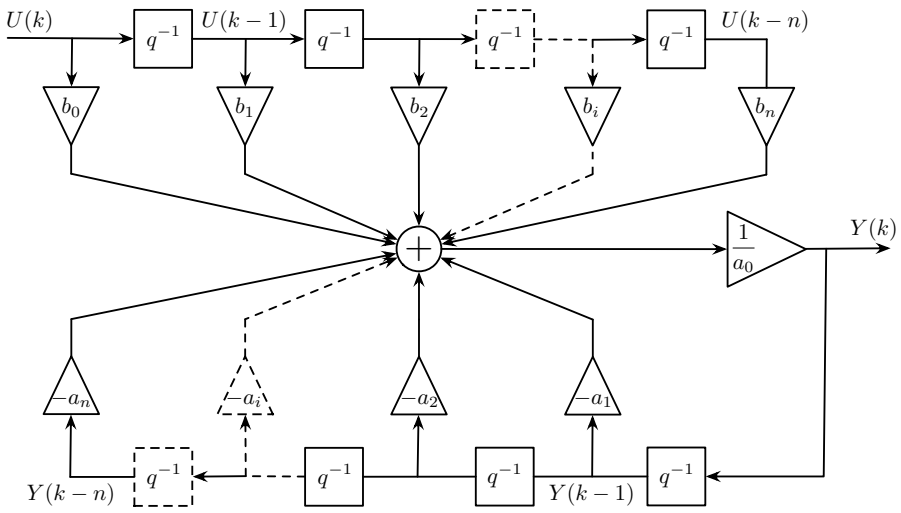


FIG. 2.7 – La forme directe I

2.1.4.2 Forme directe II

Il est possible de réécrire le système décrit par (2.11) sous une forme plus compacte que la forme directe I, en n'utilisant que n éléments retards (figure 2.8). Elle s'obtient en écrivant H sous la forme

$$H(z) = \frac{\sum_{i=0}^{n-1} \tilde{b}_i z^{-i}}{\sum_{i=0}^{n} a_i z^{-i}} + d \tag{2.13}$$

L'équation (2.13) diffère de (2.11) parce que l'on impose un degré du numérateur strictement inférieur au degré du dénominateur, cela étant possible grâce au terme direct d.
L'algorithme mis en oeuvre est alors

$$\left\{ \begin{array}{rcl} E(k) & = & \frac{1}{a_0}\left(U(k) - \sum_{i=1}^{n} a_i E(k-i)\right) \\ Y()k) & = & \sum_{i=0}^{n-1} \tilde{b}_i E(k-i) + dU(k) \end{array} \right. \tag{2.14}$$

où $E(k)$ correspond aux notions d'*état partiel* [55] et de sortie plate [68]. Cela peut s'écrire sous forme espace d'état, avec

$$X(k) = \begin{pmatrix} E(k-n) \\ E(k-n+1) \\ \vdots \\ E(k-1) \end{pmatrix} \tag{2.15}$$

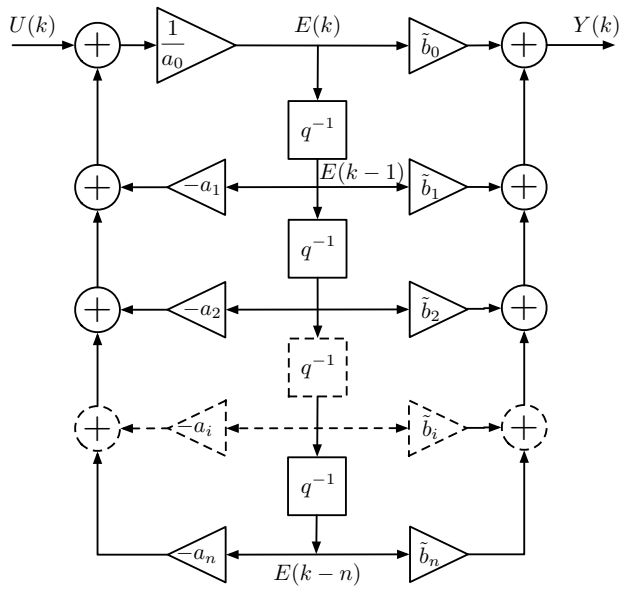


FIG. 2.8 – La forme directe II

Pour $a_0 = 1$, cette forme est aussi appelée forme **compagne** horizontale (ou forme *commandable*) et peut s'écrire sous forme espace d'état[8] :

$$\left\{\begin{array}{rcl} X(k+1) & = & \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \dots & \dots & 0 & 1 \\ -a_1 & -a_2 & \dots & \dots & -a_n \end{pmatrix} X(k) + \begin{pmatrix} 0 \\ \vdots \\ \vdots \\ 0 \\ 1 \end{pmatrix} U(k) \\ Y(k) & = & \begin{pmatrix} \tilde{b}_0 & \dots & \tilde{b}_{n-1} & 0 \end{pmatrix} X(k) + d\, U(k) \end{array}\right. \tag{2.16}$$

2.1.4.3 Formes transposées

Dans le cas mono-entrée/mono-sortie, le théorème de transposition de la théorie des graphes stipule qu'on ne change pas la fonction de transfert du système lorsque l'on construit le graphe transposé d'une structure de graphe. On transpose un graphe en changeant le sens de ses branches et en intervertissant entrée et sortie. On peut donc ainsi obtenir deux autres

[8] Si $a_0 \neq 1$, on ne peut l'écrire exactement sous forme espace d'état en préservant les paramètres $(a_i)_{0 \leqslant i \leqslant n}$, seulement avec des paramètres $\left(\frac{a_i}{a_0}\right)_{1 \leqslant i \leqslant n}$, ce qui est une forme bien entendu mathématiquement équivalente, mais qui devient non équivalente en précision finie.

formes directes, les formes transposées I et II, elles aussi fréquentes dans la littérature. La figure 2.9 montre cette réalisation.
La forme transposée II, lorsque $a_0 = 1$, correspond au système suivant (forme

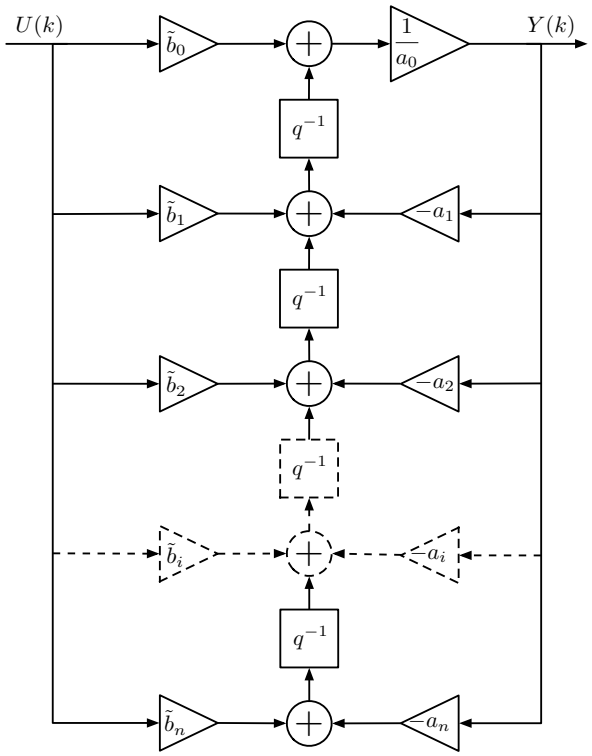


FIG. 2.9 – La forme transposée II

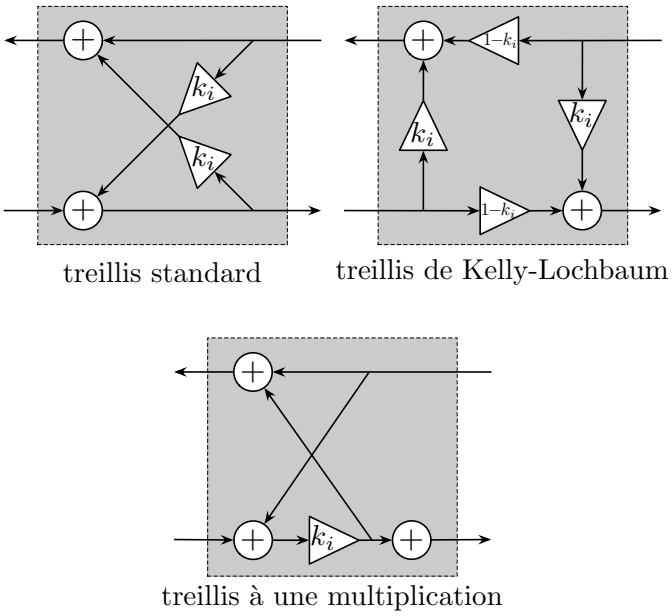
compagne verticale) :

$$\begin{cases} X(k+1) = \begin{pmatrix} -a_n & 1 & 0 & \dots & 0 \\ \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ -a_1 & 0 & \dots & 0 & 1 \\ -a_0 & 0 & \dots & \dots & 0 \end{pmatrix} X(k) + \begin{pmatrix} b_n \\ \vdots \\ \vdots \\ b_1 \\ b_0 \end{pmatrix} U(k) \\ Y(k) = \begin{pmatrix} 1 & 0 & \dots & 0 \end{pmatrix} X(k) + d \; U(k) \end{cases} \tag{2.17}$$

2.1.5 Filtres en treillis

La structure de filtres en treillis (*lattice filter*) est une forme de réalisation couramment utilisée dans les applications d'analyse et de synthèse de la

parole et, plus généralement, pour les systèmes de prédiction linéaire.
Cette forme consiste en l'utilisation de blocs de jonction dits "en treillis" représentés sur la figure 2.10. Il en existe de plusieurs sortes, tels que le treillis standard, le treillis de Kelly-Lochbaum ou encore le treillis à une seule multiplication[5].
En cascadant les blocs (cf. figure 2.11), on réalise simultanément deux filtres,



treillis standard — treillis de Kelly-Lochbaum

treillis à une multiplication

FIG. 2.10 – Différents blocs de jonction en treillis

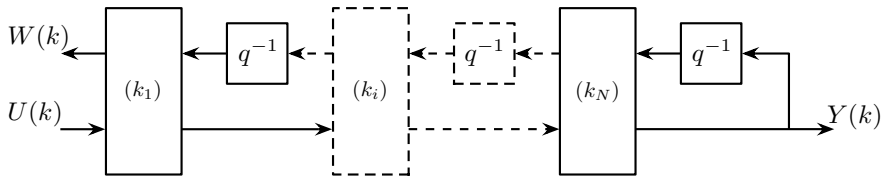


FIG. 2.11 – Filtre en treillis

d'entrée U et de sortie Y et W, qui vérifient :

$$Y(k) = U(k) + \sum_{i=0}^{N} a_i U(k-i) \tag{2.18}$$

$$W(k) = Y(k) + \sum_{i=0}^{N} a_{N-i} Y(k-i) \tag{2.19}$$

où les coefficients $(a_i)_{1\leqslant i\leqslant N}$ se calculent d'après les coefficients $(k_i)_{1\leqslant i\leqslant N}$. Il existe de nombreuses possibilités de filtres RIF et RII, suivant comment sont combinés les blocs en treillis. On peut aussi étendre le filtre en treillis pour réaliser n'importe quelle fonction de transfert, comme sur la figure 2.12. Les filtres en treillis sont principalement utilisés lorsque l'on veut réa-

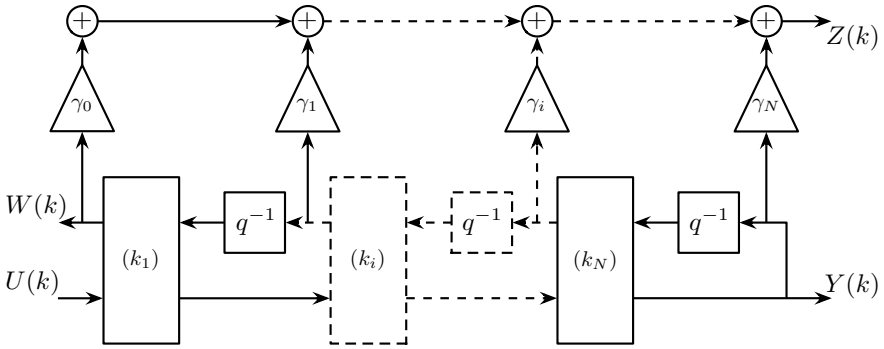


FIG. 2.12 – Utilisation de blocs treillis pour implémenter n'importe quelle fonction de transfert

liser simultanément deux filtres RIF dont les fonctions de transfert sont des polynômes images[9] (et sont donc réservées à des applications bien spécifiques). De plus, une propriété, intéressante en pratique, explique l'intérêt pour de telles structures : une condition nécessaire et suffisante pour qu'un filtre RII soit stable (tous ses pôles à l'intérieur du cercle unité) est que les coefficients $(k_i)_{1\leqslant i\leqslant N}$ soient de module inférieur à l'unité. Il en résulte un contrôle de stabilité très simple à réaliser, tout particulièrement pour les systèmes où les valeurs des coefficients évoluent en permanence, comme les filtres adaptatifs [11].

2.1.6 Opérateur δ

Toutes les réalisations vues précédemment utilisaient l'opérateur *avance* q (ou l'opérateur inverse *retard* q^{-1}) dans leur description et leurs calculs. Depuis Middleton et Goodwin [76], un autre opérateur, créé pour unifier les descriptions en temps continu et en temps discret, est utilisé pour ses bonnes propriétés numériques [117, 31]. Il s'agit de l'opérateur δ défini par

$$\delta \triangleq \frac{q-1}{\Delta} \tag{2.20}$$

[9] Deux polynômes P et Q de $\mathbb{R}[X]$ s'écrivant $P = \sum_{i=0}^{N} p_i X^i$ et $Q = \sum_{i=0}^{N} q_i X^i$ sont des polynômes images si et seulement si $p_i = q_{N-i}$, $\forall\ i$ tel que $0 \leqslant i \leqslant N$.

où Δ est une constante strictement positive (dans la définition de [76], Δ correspond à la période d'échantillonnage du système, mais Gevers et Li [31] montrent que l'on peut utiliser n'importe quelle valeur strictement positive. On notera toutefois que certaines propriétés des réalisations en δ, telles que le bon conditionnement numérique des réalisations, sont liées à la condition $\Delta < 1$).
L'opérateur δ a été mis en place pour rapprocher les opérateurs temps discret et temps continu : l'opérateur δ tend vers l'opérateur continu $\frac{d}{dt}$ lorsque Δ tend vers 0.
Il est aisé de passer d'une réalisation dans l'espace d'état avec l'opérateur q (équation (2.3)) à une réalisation décrite dans un espace d'état et utilisant l'opérateur δ définie par

$$\left\{ \begin{array}{rcl} \delta X(k) & = & A_\delta X(k) + B_\delta U(k) \\ Y(k) & = & C_\delta X(k) + D_\delta U(k) \end{array} \right. \tag{2.21}$$

par les relations

$$A_\delta = \frac{A - I}{\Delta}, \quad B_\delta = \frac{B}{\Delta}, \quad C_\delta = C, \quad D_\delta = D \tag{2.22}$$

L'algorithme associé à une telle réalisation serait alors le suivant :

1: $T \leftarrow A_\delta X(k) + B_\delta U(k)$
2: $X(k+1) \leftarrow X(k) + \Delta T$
3: $Y(k) \leftarrow C_\delta X(k) + D_\delta U(k)$

On calcule tout d'abord le terme correspondant à $\delta X(k)$. Puis $X(k+1)$ est calculé (car $\delta^{-1} = \frac{\Delta q^{-1}}{1-q^{-1}}$). Enfin, $Y(k)$ est calculé classiquement.
Une réalisation équivalente utilisant l'opérateur q conduit à :

1: $X(k+1) \leftarrow (A_\delta \Delta + I)\, X(k) + (B_\delta \Delta)\, U(k)$
2: $Y(k) \leftarrow C_\delta X(k) + D_\delta U(k)$

Ces deux algorithmes sont *mathématiquement* équivalents, mais ne le sont plus en précision finie car les coefficients impliqués ne sont pas les mêmes (A_δ, B_δ et Δ dans un cas, $(A_\delta \Delta + I)$ et $(B_\delta \Delta)$ dans l'autre).

Supposons que le régulateur ou filtre a pour fonction de transfert H_q, cette fonction de transfert peut s'écrire[10] avec la variable ρ, associée[11] à

[10] Par abus de langage, et pour ne pas alourdir le propos, nous noterons les fonctions $H_\delta : \mathbb{C} \to \mathbb{C}$, $\rho \mapsto H_\delta(\rho)$ et $H_q : \mathbb{C} \to \mathbb{C}$, $z \mapsto H_q(z)$ par la même fonction H, exprimée en ρ ou en z suivant le contexte.

[11] On peut souvent trouver, dans la littérature, l'utilisation de la variable "γ" associée à l'opérateur δ [76], mais nous réserverons la notation γ à l'opérateur γ présenté au pa-

l'opérateur δ comme suit :

$$H_\delta(\rho) = \frac{\sum_{i=0}^{n} d_i \rho^{-i}}{1 + \sum_{i=1}^{n} c_i \rho^{-i}} \tag{2.23}$$

Si cette même fonction de transfert s'écrit, en z

$$H_q(z) = \frac{\sum_{i=0}^{n} b_i z^{-i}}{1 + \sum_{i=1}^{n} a_i z^{-i}} \tag{2.24}$$

on aura $H_q(z) = H_\delta(\frac{z-1}{\Delta})\ \ \forall z \in \mathbb{C}$ si et seulement si les coefficients $(b_i)_{0 \leqslant i \leqslant n}$, $(d_i)_{0 \leqslant i \leqslant n}$, $(a_i)_{1 \leqslant i \leqslant n}$ et $(c_i)_{1 \leqslant i \leqslant n}$ sont reliés par

$$c_i = \Delta^{-i} \sum_{j=0}^{i} a_j C_{n-j}^{i-j} \qquad 1 \leqslant i \leqslant n \tag{2.25}$$

$$d_i = \Delta^{-i} \sum_{j=0}^{i} b_j C_{n-j}^{i-j} \qquad 0 \leqslant i \leqslant n \tag{2.26}$$

où C_n^p est le nombre de combinaisons possibles de p éléments parmi n (on trouvera la démonstration à l'annexe B).
Il est alors possible de reprendre les réalisations vues précédemment (notamment les formes directes II), en remplaçant l'opérateur retard q^{-1} par l'opérateur δ^{-1}, représenté par la figure 2.13, et en remplaçant les coefficients $(a_i)_{0 \leqslant i \leqslant n}$ et $(b_i)_{0 \leqslant i \leqslant n}$ par les coefficients $(c_i)_{0 \leqslant i \leqslant n}$ et $(d_i)_{0 \leqslant i \leqslant n}$.
On peut aussi voir l'intégrateur $\rho \mapsto \rho^{-1}$ comme une approximation discrète

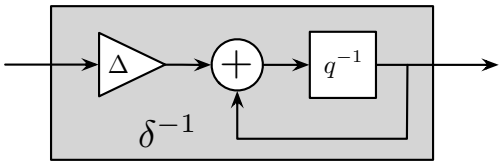


FIG. 2.13 – L'opérateur δ^{-1} réalisé avec l'opérateur retard q^{-1}

de l'intégrateur continue $s \mapsto \frac{1}{s}$ en utilisant la méthode des rectangles.

2.1.7 Autres opérateurs

En plus de l'opérateur δ introduit par Middleton et Goodwin, d'autres opérateurs, tels que l'opérateur γ ou l'opérateur δ-modifié, sont envisageables

ragraphe 2.1.7.1. Nous n'utiliserons pas non plus l'abus d'écriture consistant à confondre l'opérateur q (respectivement δ) et la variable z (resp. ρ) (voir voir les remarques à ce propos dans [70]).

pour mettre en œuvre numériquement les lois. Ils sont connus sous le nom d'*Opérateurs Alternatifs en Temps Discret*[12] [7]

2.1.7.1 Opérateur γ

L'opérateur γ, introduit par Gevers et Li [31], est défini par

$$\gamma \triangleq \frac{2}{\Delta}\frac{q-1}{q+1} \tag{2.27}$$

et correspond, si Δ représente la période d'échantillonnage, à l'approximation de l'intégrateur continu $s \mapsto \frac{1}{s}$ par l'intégrateur discret avec la méthode des trapèzes (approximation de Tustin).
Par contre, contrairement à ce qui se passe avec l'opérateur δ où l'on peut écrire explicitement les relations entrées/sorties, l'intégration est implicite : $Z(k) = \gamma^{-1}[V(k)]$ s'écrit

$$Z(k) = Z(k-1) + \frac{\Delta}{2}\big(V(k) + V(k-1)\big) \tag{2.28}$$

Ainsi, avec une relation du type

$$\gamma X(k) = f\left(X(k), U(k)\right) \tag{2.29}$$

$X(k+1)$ ne peut être calculé directement car l'évaluation de

$$\gamma^{-1}\left[f\left(X(k), U(k)\right)\right] \tag{2.30}$$

requiert la connaissance de $X(k+1)$. Et lever ce caractère implicite ne peut être obtenu sans changer la paramétrisation.
Toutefois, une technique itérative [91] peut permettre d'approximer le calcul de $X(k+1)$ au prix d'un coût de calcul important.

2.1.7.2 Autres opérateurs

De nombreux autres opérateurs peuvent ainsi être considérés pour l'implémentation. Ils sont décrits plus en détails dans [7].
On pourra citer :

– l'opérator δ-modifié :

$$\delta_m \triangleq \frac{q-c}{\Delta} \qquad \text{avec } 0 < c \leqslant 1 \tag{2.31}$$

où c est un paramètre ajustable. On montrera (cf. section 5.2.3) que ce paramètre supplémentaire peut être intéressant dans le cas où Δ n'est pas implémenté exactement car il permet de trouver une réalisation en δ-modifié moins sensible aux quantification de Δ ;

[12] *Alternative Discrete Time Operators.*

– l'opérateur ρ[13] :

$$\rho \triangleq \frac{q - (1 - c_1\Delta)}{c_2\Delta} \tag{2.32}$$

Cet opérateur, introduit dans le cadre de la commande robuste adaptative [25], est une généralisation des opérateurs q (pour $c_1\Delta = c_2\Delta = 1$), δ (pour $c_1 = 0$ et $c_2 = 1$) et δ_m ;

– l'opérateur π :

$$\pi \triangleq \frac{2}{\Delta}\frac{c_1 q - c_2}{c_3 q + c_4} \qquad \text{avec } c_1 c_4 \neq -c_2 c_3 \tag{2.33}$$

On retrouve ici les opérateurs q, δ, δ_m, ρ et γ (pour $c_1 = c_4 = c_2 = c_3 = 1$).

2.1.8 Décomposition de lois

En plus de toutes les réalisations que nous avons détaillées, il est possible de découper une loi de contrôle/commande en lois élémentaires d'ordre inférieur – qui peuvent être implémentées différemment – que l'on associe, en cascade ou en parallèle. Cela permet souvent, en découpant un système d'ordre élevé en sous-sytèmes d'ordre faible, de n'avoir que des systèmes simples, souvent du 1er ou 2nd ordre, à implémenter. On peut alors se concentrer sur la réalisation de manière optimisée (réalisation optimale et code optimisé) de lois génériques du 1er ou 2nd ordre.

2.1.8.1 Décomposition en cascade

Tout système qui est une interconnexion de sous-systèmes – au sens où la sortie de chaque sous-système est l'entrée d'un de ses voisins, à l'exception du dernier – est dit *système en cascade*, voir figure 2.14. Pour un régulateur

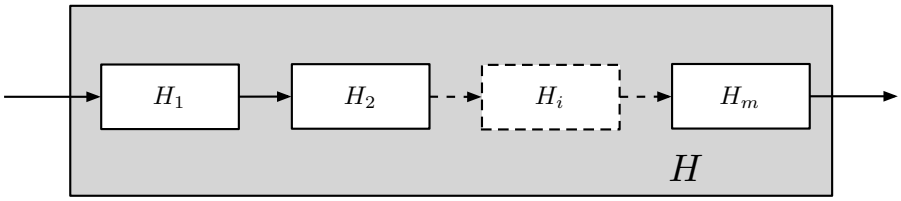


FIG. 2.14 – Structure en cascade

ou un filtre LTI, déterminer une structure en cascade revient à factoriser la

[13] À ne pas confondre avec la variable ρ associée à l'opérateur δ.

fonction de transfert H :

$$H(z) = \prod_{i=1}^{m} H_i(z) \qquad \forall z \in \mathbb{C} \tag{2.34}$$

chaque sous-système $H_i(z)$ étant réalisé indépendamment. La décomposition en sous-systèmes du 2nd ordre (et du 1er ordre si le système est d'ordre impair) est la plus naturelle. On suppose ici le système d'ordre n pair :

$$H(z) = K \prod_{i=1}^{\frac{n}{2}} \frac{1 + e_i z^{-1} + f_i z^{-2}}{1 + g_i z^{-1} + h_i z^{-2}} \tag{2.35}$$

Cette structure peut permettre une distribution optimale des zéros et des pôles pour les différents sous-systèmes, chacun pouvant être réalisé de manière optimale [36]. Cependant, cette structure possède quelques désavantages : pour les systèmes non strictement propres, ou un terme direct entre l'entrée et la sortie intervient, un délai de calcul est introduit, car la sortie n'est produite que lorsque le calcul de chacun des blocs cascadés est réalisé ; de plus cette structure ne peut s'appliquer au cas MIMO.

2.1.8.2 Décomposition en parallèle

De la même manière qu'il est possible de mettre un système sous forme cascade, on peut décrire un système sous une forme *parallèle*, où le système s'exprime comme la somme de sous-systèmes, cf. figure 2.15. Pour un système de fonction de transfert H, la décomposition en parallèle revient à décomposer H en somme de fractions rationnelles, souvent du 2nd ordre. Dans le cas de pôles simples[14], on peut écrire :

$$H(z) = d + \sum_{i=1}^{p} \frac{k_i}{1 + l_i z^{-1}} + \sum_{i=1}^{q} \frac{m_i + n_i z^{-1}}{1 + p_i z^{-1} + q_i z^{-2}} \tag{2.36}$$

Comme pour la structure en cascade, chaque sous-système peut être implémenté d'une manière quelconque, classiquement avec la forme directe II.

2.2 L'arithmétique en précision finie

Toutes les réalisations que l'on vient de voir dans la section précédente décrivent des algorithmes que l'on doit mettre en œuvre dans un calculateur cible. De par sa nature numérique (les calculs réalisés reposent sur une représentation électrique des éléments manipulés), certaines limitations vont

[14] Dans le cas contraire, on peut utiliser une forme de Jordan.

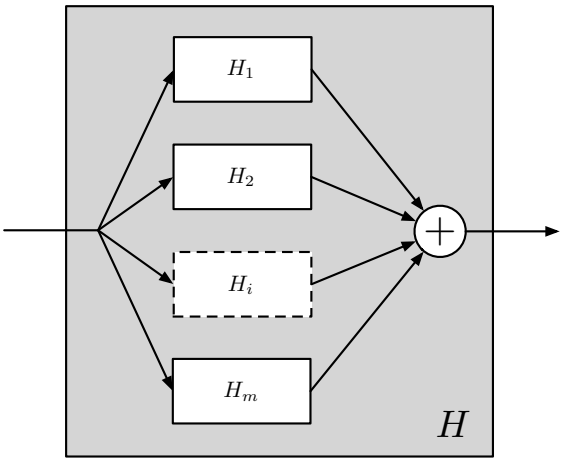


FIG. 2.15 – Structure parallèle

dégrader la réalisation des lois. Ces limites reposent sur les nombreuses possibilités pour représenter et manipuler des nombres - les deux plus utilisées étant la représentation en *virgule fixe* et la représentation en *virgule flottante*.
Cette section a pour but d'examiner les différentes représentations possibles des entiers et des réels, ainsi que leur impact, en terme de quantification des coefficients et de bruits numériques induits dans les calculs.

2.2.1 Représentation numérique

Tout d'abord, on peut commencer par s'interroger sur quels sont les objets mathématiques, tels que entiers, vecteurs, fonctions, etc. qu'il est possible de représenter de manière *binaire* dans un calculateur[15] actuel.

On comprend aisément qu'il est possible de manipuler des "objets" dont la représentation est finie (comme par exemple les entiers compris entre 0 et 255), mais qu'en est-il des ensembles infinis ? Par exemple, pouvons-nous représenter et manipuler tous les entiers ? Et tous les réels ?

Premièrement, on dispose d'un moyen de coder n'importe quel entier, aussi grand soit-il, car sa représentation en base 10 (et dans n'importe quelle base) est finie : pour un entier N, seuls $\lceil \log_{10}(N) \rceil$ chiffres suffisent ($\lceil . \rceil$ est l'opérateur d'arrondi par excès).

En revanche, si l'on considère le nombre π, sa transcendance[16] nous em-

[15] Le terme calculateur représente ici tout dispositif numérique permettant de réaliser un ensemble de calculs.

[16] Un nombre transcendant est un nombre réel ou complexe qui n'est racine d'aucune

pêche de le décrire exactement avec un nombre fini d'entiers, de fractions rationnelles et de leurs racines. Ainsi, puisque l'on ne peut disposer que des premières décimales de π, une description fondée sur une numérotation de position (c'est à dire une numération où la position d'un chiffre détermine son poids, en fonction de la base de numération) est impossible[17].

Nous allons donc essayer de voir quels ensembles peuvent être décrits de manière logicielle dans un calculateur.
Seules deux données peuvent être manipulées par un processeur : le **0** (potentiel nul, ou inférieur à une certaine valeur) et le **1** (potentiel haut, ou supérieur à un seuil). Dans ce qui suit, nous allons donc rappeler la définition de *langage* appliqué par un processeur pour définir quels ensembles il est possible de coder.

Définition 2.2 (alphabet, mot, langage)

*Soit Σ un ensemble d'éléments quelconques. Σ définit un **alphabet**.*
*On appelle **mot** une suite* finie *d'éléments d'un alphabet, ϵ étant le mot vide. Par exemple, pour l'alphabet $\Sigma = \{a, b\}$, aabbababbabb est un mot, tout comme aba ou bbbb.*
*Un **langage** défini sur un alphabet est un ensemble de mots construits avec cet alphabet.*
L'ensemble des mots formés sur l'alphabet Σ et de longueur k se note Σ^k.
On note aussi Σ^* l'ensemble de tous les mots formés sur Σ, y compris le mot vide ϵ (on a $\Sigma^* = \bigcup_{k\geq 0} \Sigma^k$), et Σ^+ l'ensemble de tous les mots formés sur Σ sans le mot vide.

Nous considérons donc ici l'alphabet $\Sigma = \{0, 1\}$, puisque seuls 0 et 1 sont utilisés *physiquement* sur les calculateurs étudiés[18].
Sur un processeur N bits, le langage utilisé par les circuits est le plus souvent Σ^N (lorsque les chemins de données ont tous la même largeur). Généralement N vaut 8,16,32 ou 64 suivant le modèle du processeur. Le tableau 2.1 donne l'exemple de quelques processeurs classiques

Cela permet, par exemple, de représenter des valeurs entières de 0 à $2^N - 1$, ou bien tout élément d'un ensemble comprenant moins de 2^N éléments.

Du point de vue logiciel, nous utilisons un autre langage : en effet, il nous est possible de combiner un nombre fini de mots sur l'alphabet Σ, comme c'est le cas lorsque l'on veut représenter un vecteur d'entiers ou bien

équation polynomiale à coefficients entiers.

[17] Le record actuel, de décembre 2002, est de 1 241 100 000 000 décimales de π calculés. Toutefois, il existe certains algorithmes permettant de calculer n'importe quelle décimale de π sans avoir à calculer les précédentes.

[18] Seuls les processeurs *quantiques* n'utilisent pas ce principe ; ils n'existent pour l'instant qu'à l'état expérimental [89].

Processeur	Nb de bits
Pentium IV	32
PowerPC G5	64
C167	16

TAB. 2.1 – Quelques processeurs classiques

lorsque l'on veut effectuer des calculs avec une plus grande *dynamique* que celle proposée par l'architecture du processeur (comme un calcul 32 bits sur un processeur 16 bits). Si nous ne nous soucions pas de la taille mémoire occupée, n'importe quelle suite finie de mots de Σ^N peut être utilisée.
Par exemple, en utilisant une représentation des lettres de l'alphabet (un entier par lettre), nous pouvons représenter n'importe quelle phrase, n'importe quel texte, aussi long soit-il, du moment qu'il reste de longueur finie, et *en supposant* – cette hypothèse est importante – que l'on dispose d'une mémoire suffisamment grande pour stocker ce texte.

Nous pouvons donc définir un alphabet *logiciel* $A = \Sigma^N$ (A est composé des 2^N mots de longueur N définis par le langage Σ^N). Le langage *logiciel* utilisé est alors A^+, c'est-à-dire que nous pouvons manipuler tous les mots de A de longueur finie, c'est-à-dire **toutes les suites finies d'éléments représentés sur N bits.**

Définition 2.3 (codage d'un ensemble par un alphabet)
Il est possible de ***coder*** *(*représenter*) un ensemble E par un alphabet L s'il existe une application $f : E \to L^+$ injective, c'est à dire*

$$\forall x \in E, \forall \tilde{x} \in E, \quad f(x) = f(\tilde{x}) \Rightarrow x = \tilde{x} \tag{2.37}$$

Par exemple, il est possible de représenter toutes les lettres de l'alphabet, les caractères de ponctuation, les chiffres et divers symboles, par des valeurs comprises entre 0 et 255 (8 bits), grâce à un tableau de correspondance normalisée, le codage ASCII[19]. Par exemple, le caractère `A` correspond à la valeur 65. Dans ce cas précis, l'application de codage, qui permet de passer de l'ensemble des caractères alphanumériques et des symboles associés, à un mot de Σ^N (un mot sur N bits), est une application bijective : à chaque valeur correspond un caractère, et à chaque caractère correspond une valeur.

Mais, pour un codage, seule l'injectivité est nécessaire. Elle permet de retrouver l'élément de E représenté par un mot de L^+ car celui-ci, s'il existe, est unique. Par exemple, pour représenter en binaire des entiers signés (positifs ou négatifs), de nombreux codages existent. On peut citer une méthode de représentation en signe et en valeur absolue (cf. 2.2.2.2). Elle consiste

[19] A*merican* S*tandard* C*ode for* I*nformation* I*nterchange*.

à utiliser un bit (de poids fort généralement) pour représenter le signe, et les autres bits pour représenter la valeur absolue de la valeur. Ce codage n'est pas bijectif, mais seulement injectif, car la valeur 0 est doublement représentée (par -0 et par $+0$).

En partant de l'hypothèse que l'ensemble des valeurs *logicielles* que l'on peut utiliser est A^+ (ce qui suppose que l'on dispose, *a priori*, d'une capacité de mémoire et de calcul suffisamment grande), nous allons pouvoir caractériser les ensembles codables, c'est à dire ceux que l'on peut représenter de manière logicielle.

Mais avant cela, rappelons deux définitions utiles :

Définition 2.4 (Ensemble dénombrable)
Un ensemble E est dénombrable si et seulement si il est équipotent à $\mathbb{N}$ (c'est à dire s'il est en bijection avec $\mathbb{N}$).

Par exemple, $\mathbb{Q}$ et $\mathbb{Z}$ sont dénombrables, alors que $\mathbb{R}$ ou $\mathbb{Q}^{\mathbb{N}}$ (ensemble des suites de nombres rationnels) ne le sont pas.

Définition 2.5 (Cardinal d'un ensemble)
Le cardinal d'un ensemble fini E est le nombre d'éléments de cet ensemble. Il est noté `Card`(E).
Cette définition a été étendue par Cantor[15] de la manière suivante : puisque la relation

$$E \text{ est équipotent à } F \tag{2.38}$$

est une relation d'équivalence, on peut définir le cardinal d'un ensemble comme une classe d'équivalence pour l'équipotence.
Ainsi $Card(E) = Card(F)$ est équivalent à l'équation (2.38).
De même, on peut ajouter une relation d'ordre. Les propositions suivantes sont équivalentes (conséquences du Théorème de Cantor-Bernstein) :

- $\mathrm{Card}(E) \leq \mathrm{Card}(F)$;
- *E est équipotent à une partie de F ;*
- *E est subpotent à F ;*
- *il existe une injection de E à F.*

On peut compléter cette définition par la définition de $\aleph_0$, la classe cardinale du plus petit ensemble infini[20]. $\mathbb{N}$ est de cardinalité $\aleph_0$, ainsi que tous les ensembles dénombrables[21].

Le résultat de cette dernière définition nous permet d'énoncer la proposition suivante :

[20] $\aleph$ est la 1ère lettre de l'alphabet hébreu et se prononce "aleph". Cantor a choisi cette notation.

[21] Le cardinal de $\mathbb{R}$ est noté $\aleph_1$, et on a : $\mathrm{Card}(\mathbb{R}) = \aleph_1 = 2^{\aleph_0}$.

Proposition 2.1 (Ensembles codables)
Un ensemble J est codable (de manière logicielle, et suivant l'hypothèse du langage A^+) si et seulement si J est dénombrable ou fini.

Démonstration :
Pour la démonstration, nous avons besoin du lemme suivant

Lemme 2.2
Si un alphabet Σ est fini ou dénombrable, alors Σ^*, l'ensemble des mots formés sur l'alphabet Σ, est dénombrable.

En effet, on ne peut coder de manière logicielle un ensemble J que s'il existe $f : J \to A^+$ injective.
Or ceci peut se traduire, en terme de classes cardinales par :

$$\text{card}(J) \leq \text{card}(A^+) \tag{2.39}$$

c'est à dire

$$\text{Card}(J) \leq \aleph_0 \tag{2.40}$$

Ainsi, on a :
- **ou bien** $\text{Card}(J) < \aleph_0$, et donc $\text{Card}(J)$ est fini, d'où J est fini (car $\aleph_0$ est le plus petit cardinal transfini) ;
- **ou bien** $\text{Card}(J) = \aleph_0$, et donc J est dénombrable.

La réciproque se montre de la même manière :
- Si J est dénombrable, J et $\mathbb{N}$ sont équipotents, et J et A^+ le sont. Il existe donc une fonction bijective (et donc injective) de $J \to A^+$, et J est codable.
- Si J est fini, de cardinal n, alors il existe une bijection $J \to [\![1, n]\!]$, et donc une fonction injective $J \to \mathbb{N}$. A^+ et N étant équipotents, il existe une injection $J \to A^+$ et J est codable.

■

Cette proposition nous permet ainsi d'affirmer que des ensembles tels que $\mathbb{N}$, $\mathbb{Z}^3$ ou $\mathbb{Q}$ sont représentables en utilisant le langage A^+, donc en utilisant une suite finie d'éléments sur N bits, mais qu'il est impossible de représenter en totalité des ensembles non dénombrables, tels que $\mathbb{R}$.

Ce résultat important nous amène à conclure qu'il est impossible de calculer de manière *exacte*, *absolue* sur un ensemble non dénombrable (impossible de calculer exactement dans $\mathbb{R}$ ou dans n'importe quel intervalle $[a, b]$). Ainsi, **on ne peut numériquement concevoir une arithmétique informatique qui soit *exacte* sur $\mathbb{R}$ ou sur $[a, b]$**. Tout au plus, il est possible d'imaginer une arithmétique en *précision arbitraire*[75, 33].

De plus, on voit bien aussi l'importance du choix de la représentation, de la fonction de codage, même si, en pratique, pour l'implémentation de lois embarquées, le choix est souvent restreint par l'architecture du calculateur :

une représentation en accord avec le processeur (virgule flottante simple ou double précision, virgule fixe en complément à 2, ...) sera presque toujours utilisée. Le choix d'une autre représentation implique inévitablement une surcharge de calculs : les langages de calculs formels, tels que Maple, Mathematica, etc..., disposent d'une couche logicielle supplémentaire (l'utilisation d'un langage sur Σ^+) qui permet une arithmétique rationnelle à précision arbitraire ; les calculs pouvant s'effectuer sur un nombre de décimales choisi par l'utilisateur.

2.2.2 Représentation des entiers

On considère $(b_i)_{0\leqslant i\leqslant N-1} \in \mathbb{B}^N$ N bits permettant de représenter des entiers. b_{N-1} correspond au bit de poids fort et b_0 au bit de poids faible. Il existe de multiples manières de représenter des entiers x avec ces N bits et on pourra retrouver ces représentations plus en détails dans [12, 111].

2.2.2.1 Codage binaire naturel

Le codage binaire naturel est l'écriture binaire la plus naturelle[22]. Elle consiste à représenter les entiers de $[\![0, 2^N - 1]\!]$ par leur écriture en base 2 (voir figure 2.16). x est donné par :

$$x = \sum_{i=0}^{N-1} 2^i b_i \tag{2.41}$$

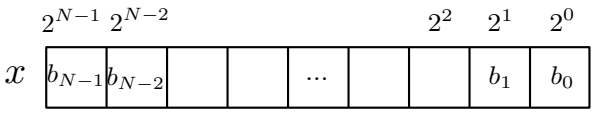


FIG. 2.16 – Représentation d'un entier avec le codage binaire naturel

[22] [48] retrace très précisément l'histoire des nombres dyadiques et de leur représentation :
- 1600 : Thomas Harist dresse une table de décomposition des entiers suivant les puissances de 2 ;
- 1679 : Leibnitz rédige le 1er manuscrit sur l'écriture binaire ;
- 1701 Thomas Fantet de Lagny montre les avantages du calcul binaire ;
- puis ensuite Euler, Brunetti, Legendre, Pierce (1876 : notation avec des 0 et des 1), etc.

2.2.2.2 Codage en valeur absolue signée

Pour pouvoir représenter des entiers relatifs, une première solution consiste à donner à un bit (généralement le bit de poids fort) la signification du signe (0 pour les entiers positifs, 1 pour les entiers négatifs). Les autres bits donnent la valeur absolue du nombre, en codage binaire (voir figure 2.17). C'est le codage en valeur absolue signée

$$x = (-1)^{b_{N-1}} \sum_{i=0}^{N-2} 2^i b_i \tag{2.42}$$

x appartient à l'intervalle symétrique $[\![-(2^{N-1}-1), 2^{N-1}-1]\!]$, et son opposé s'obtient en changeant son bit de signe.
Bien qu'intuitive et simple, cette approche comporte quelques inconvénients. Tout d'abord, le zéro est représenté de deux façons ($+0$ et -0), et, de plus, les opérations de comparaison et d'addition se trouvent compliquées par le fait qu'il faille distinguer différents cas suivant les signes des opérandes.

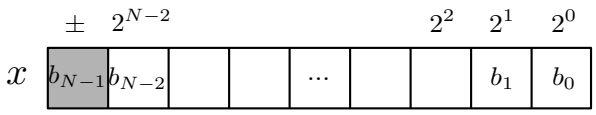


FIG. 2.17 – Représentation d'un entier relatif avec le codage valeur absolue signée

2.2.2.3 Complément à 1

Le complément à 1 est utilisé pour représenter les nombres positifs ou négatifs de $[\![-(2^{N-1}-1), 2^{N-1}-1]\!]$. Les nombres positifs sont représentés en code binaire naturel et les nombres négatifs en écrivant l'inverse du codage binaire de la valeur absolue du nombre. x s'écrit :

$$x = -b_{N-1}(2^{N-1} - 1) + \sum_{i=0}^{N-2} 2^i b_i \tag{2.43}$$

Cette représentation facilite la soustraction car l'opposé d'un nombre s'obtient en inversant tous les bits du nombre. Par contre, il existe toujours deux représentations pour le zéro, et l'addition doit se faire en distinguant les cas suivant les différents signes.
La table 2.2 donne les représentations binaires sur 4 bits des nombres entre -7 et 7.

Représentation binaire	Valeur	Représentation binaire	Valeur
0000	+0	1111	-0
0001	1	1110	-1
0010	2	1101	-2
0011	3	1100	-3
0100	4	1011	-4
0101	5	1010	-5
0110	6	1001	-6
0111	7	1000	-7

TAB. 2.2 – Représentation en complément à 1 sur 4 bits

2.2.2.4 Complément à 2

Cette représentation est une représentation fréquente pour les entiers relatifs. Elle permet de représenter les entiers de $[\![-2^{N-1}, 2^{N-1} - 1]\!]$ (intervalle non-symétrique) et ne possède qu'une seule représentation pour le zéro. L'addition est simple, quel que soit le signe (elle est identique à l'addition des nombres en codage binaire naturel). On obtient la représentation en complément à 2 d'un nombre en ajoutant 1 au complément à 1 pour les nombres négatifs.

$$x = -b_{N-1}2^{N-1} + \sum_{i=0}^{N-2} 2^i b_i \tag{2.44}$$

La table 2.3 donne les représentations binaires sur 4 bits des nombres entre -8 et 7.

Cette représentation est presque toujours utilisée car elle possède des pro-

Représentation binaire	Valeur	Représentation binaire	Valeur
0000	+0	1111	-1
0001	1	1110	-2
0010	2	1101	-3
0011	3	1100	-4
0100	4	1011	-5
0101	5	1010	-6
0110	6	1001	-7
0111	7	1000	-8

TAB. 2.3 – Représentation en complément à 2 sur 4 bits

priétés intéressantes pour l'addition et la soustraction : en effet, même si les résultats intermédiaires d'une série d'addition sont en dehors du domaine

de définition du codage (overflow), le résultat final sera correct, du moment que celui-ci appartient au domaine.
Par exemple, si on considère l'opération $4 + 5 - 6$ sur 4 bits, celle-ci produit un résultat (3) qui reste dans le domaine $[\![-8; 7]\!]$, mais un résultat intermédiaire $(4 + 5 = 9)$ en sort (overflow) :

$$\begin{array}{rll} & 0100 & (4) \\ + & 0101 & (5) \\ \hline & 1001 & \end{array} \qquad\qquad \begin{array}{rll} & 1001 & \\ + & 1010 & (-6) \\ \hline & 0011 & (3) \end{array} \tag{2.45}$$

Ainsi, avec le complément à 2, le résultat d'une opération reste correcte s'il est dans le domaine, même si un overflow intervient dans un calcul intermédiaire.
De plus, toutes les opérations sont compatibles avec les opérations définies avec le codage binaire et s'effectuent sans modifications, en utilisant les unités de calcul (UAL[23]) du calculateur.

2.2.2.5 Résumé des principales représentations

La figure 2.18 (tirée de [12]) résume les caractéristiques des diverses représentations binaires vues précédemment.

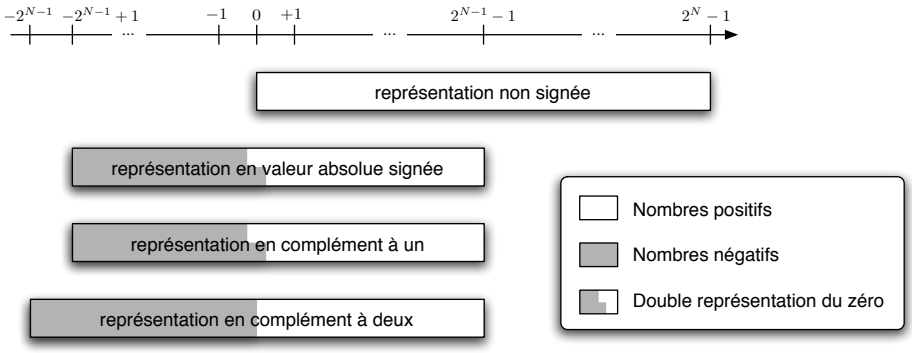


FIG. 2.18 – Résumé des diverses représentations entières sur N bits

2.2.2.6 Autres représentations

On peut aussi compléter cette liste de représentations en considérant les biais et les facteurs d'échelle que l'on peut ajouter, afin de pouvoir représenter sur N bits, le domaine que l'on cherche.

[23] *U*nité *A*rithmétique et *L*ogique.

2.2.2.6.1 Arithmétique biaisée Il s'agit ici tout simplement de décaler une représentation d'une certaine valeur *fixée*. Le biais n'étant pas codé, celui-ci peut être quelconque.
Le biais introduit permet de décaler le domaine de représentation et donc de coder un ensemble de valeur qui ne serait pas centré autour de 0 (arithmétique signée) ou de 2^{N-2} (arithmétique non signée).
Par exemple, les représentations en compléments (à 1 ou à 2) sont des représentations biaisées (à l'exception près que le biais n'est appliqué que pour les valeurs négatives).

2.2.2.6.2 Facteur d'échelle Jusqu'ici, nous n'avons utilisé que des arithmétiques entières, alors que nous pouvons avoir besoin de décrire un domaine plus petit, mais comportant des valeurs non nécessairement entières. Il faut alors introduire un facteur d'échelle K quelconque dans la représentation (comme le biais, celui-ci n'est pas codé).
Cela permet d'étaler, ou de rétrécir, la plage de valeurs admissibles pour une représentation (le pas de quantification – écart entre deux valeurs successives – est modifié en conséquence).

2.2.2.7 Passage d'une représentation à une autre

On peut être amené, dans certains cas, à utiliser plusieurs représentations en même temps. Si on veut effectuer un calcul faisant intervenir deux nombres exprimés dans deux représentations différentes, il faut alors d'abord convertir les deux opérandes dans une représentation unique avant de faire le calcul, en s'assurant qu'il est possible de passer d'une représentation à une autre (tâche qui incombe au concepteur).
Dans les langages informatiques de haut niveau (`C`, `Java`, ...), les entiers non signés sont représentés avec le codage binaire naturel, et les entiers signés sont représentés avec le complément à 2 – le compilateur gérant automatiquement toutes les opérations.
Pour faciliter les calculs lors du passage d'une représentation à une autre, il est préférable que le biais et le facteur d'échelle soient simples :

- entier pour le biais (le passage d'une représentation à une autre nécessite alors une addition) ;
- sous la forme d'une puissance de 2 pour le facteur d'échelle : $K = 2^p$ avec $p \in \mathbb{Z}$. Ainsi une nouvelle représentation peut être calculée grâce à un décalage de bits.

2.2.2.8 Représentation logarithmique

En plus des représentations linéaires, basées sur le codage binaire, il existe des représentations non-linéaires où le pas de quantification n'est pas constant. La représentation logarithmique [62, 111] consiste à choisir une base logarithmique $D \in]0,1[$ et à écrire x sous la forme

$$x = (-1)^{2^{N-1}} D^w \tag{2.46}$$

où $w \in [\![0, 2^N - 1]\!]$ s'écrit sur $N-1$ bits en codage binaire naturel

$$w = \sum_{i=0}^{N-2} 2^i b_i$$

Ainsi, $|x| \in \left\{1, D, D^2, D^3, \ldots, D^{2^N-1}\right\}$.
Cette représentation a pour avantage la simplicité des opérations de multiplication et de division, qui sont remplacées par des additions et soustractions. Par contre, il n'y a pas de représentation du zéro, et les additions et soustractions sont plus compliquées et font appel à des tables. On trouvera plus de détails dans [62].

2.2.3 Virgule Fixe

La représentation en virgule fixe, qui permet de représenter et manipuler facilement des nombres *décimaux* (non nécessairement entiers) dans un calculateur ne disposant que d'unité arithmétique entière, est la représentation la plus utilisée. Nous détaillerons la représentation, les différents calculs ainsi que les aspects liés à la quantification d'un réel et aux bruits de calcul.

2.2.3.1 Représentation

La représentation en virgule fixe consiste simplement à utiliser une représentation en complément à 2 avec un facteur d'échelle sous forme d'une puissance de 2, noté ici 2^{β_f}.

$$x = \left(-b_{N-1}2^{N-1} + \sum_{i=0}^{N-2} 2^i b_i\right).2^{-\beta_f} \tag{2.47}$$

$$= -b_{N-1}2^{N-1-\beta_f} + \sum_{i=0}^{N-2} 2^{i-\beta_f} b_i \tag{2.48}$$

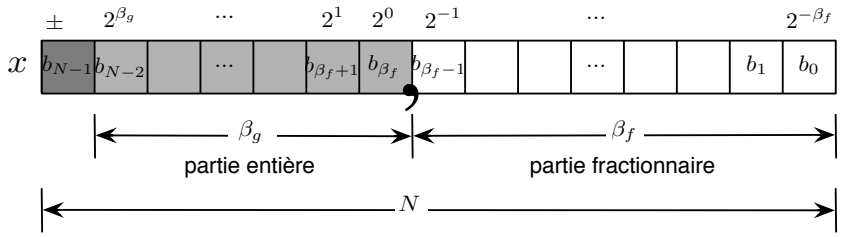


FIG. 2.19 – Représentation des données en virgule fixe

La figure 2.19 présente une donnée en virgule fixe : les β_f bits de poids faibles représentent la partie fractionnaire, et les β_g autres bits la partie entière (le dernier bit donne le signe). β_g vérifie

$$\beta_g + \beta_f + 1 = N \tag{2.49}$$

Le facteur d'échelle associé à chaque représentation en virgule fixe est constant et implicite (c'est à dire non représenté dans les bits).
On notera que β_f peut être plus grand que N : dans ce cas, le nombre ne possède pas de partie entière, mais seulement une partie fractionnaire dont seuls $\beta_f - N$ bits sont donnés. Dans ce cas, il n'y a aucun bit représentant la partie entière, et $N-1$ bits pour la partie fractionnaire (β_f ne représente plus le nombre de bits de la partie fractionnaire, mais seulement la position de la virgule). Il est aussi possible d'avoir β_f négatif.
Cette représentation permet donc de coder 2^N valeurs dans

$$\left[-2^{\beta_g}; 2^{\beta_g} - 2^{-\beta_f}\right]$$

avec un pas de quantification est $q = 2^{-\beta_g}$.
Un nombre représenté en virgule fixe est complètement déterminé par les N bits $(b_i)_{0 \leqslant i \leqslant N-1}$ et par la position de la virgule (la valeur de β_f, non codée).

2.2.3.2 Loi de quantification

On peut toutefois rattacher l'écriture en virgule fixe à celle d'un nombre $x \in \mathbb{R}$ en base 2 : celui-ci s'écrit de manière unique avec une infinité de bits :

$$x = \pm \sum_{i=-\infty}^{M} x_i 2^i \tag{2.50}$$

(avec $x_M = 1$ pour l'unicité de l'écriture et $\forall i \leqslant M, x_i \in \mathbb{B}$).
On peut d'ailleurs réécrire l'équation (2.50) sous la forme d'un complément

à 2 pour intégrer l'expression du signe

$$x = \tilde{x}_{M+1} 2^{M+1} + \sum_{i=-\infty}^{M} \tilde{x}_i 2^i \tag{2.51}$$

Les $(\tilde{x}_i)_{i \leqslant M+1}$ définissent entièrement x en base 2 :

$$\tilde{x}_{M+1}\tilde{x}_M\tilde{x}_{M-1}\ldots\tilde{x}_1\tilde{x}_{0\triangle}\tilde{x}_{-1}\tilde{x}_{-2}\ldots\ldots$$

où $\triangle$ représente la position de la virgule. Les $M+2$ bits donnent ainsi la partie entière signée de x.

On peut alors voir la représentation en virgule fixe comme une représentation binaire (en complément à 2) de x où seuls N bits de poids fort sont présents. La valeur représentée est alors une approximation.

Par exemple, les 1^{ers} bits de π s'écrivent

$$011_{\triangle}00100100001111110110101010001000\ldots\ldots$$

Pour représenter, de manière approchée, π sur 8 bits, on ne gardera que les 8 bits les plus importants, c'est à dire $011_{\triangle}00101$ ou $011_{\triangle}00100$ suivant la loi de quantification.

Définition 2.6 (loi de quantification)
Une loi de quantification est une fonction $\mathbb{R} \to \mathbb{R}$ permettant de transformer un réel x en un réel x^* représentable par une représentation donnée.

Dans le cas de la virgule fixe, deux lois de quantifications sont possibles : la quantification par *arrondi* et la quantification par *troncature* :

– La loi de quantification par troncature consiste à choisir, pour représenter x, la valeur x^* obtenue en ne gardant que les N bits de poids fort de la représentation binaire exacte de x (équation (2.51)). Cette loi est couramment employée car elle consiste à tronquer les bits de poids faible superflus (la perte de précision se fait par une perte de bits). Cette loi de quantification, noté $Q\lfloor.\rfloor$, qui transforme x en x^* est représentée en fonction de x à la figure 2.20, ainsi que l'erreur de quantification $x - x^*$.
x^* s'obtient par

$$x^* = 2^{-\beta_f} \left\lfloor x 2^{\beta_f} \right\rfloor \tag{2.52}$$

(où $\lfloor.\rfloor$ est l'opérateur d'arrondi à l'entier inférieur) et

$$|x - x^*| \leqslant 2^{-\beta_f} \tag{2.53}$$

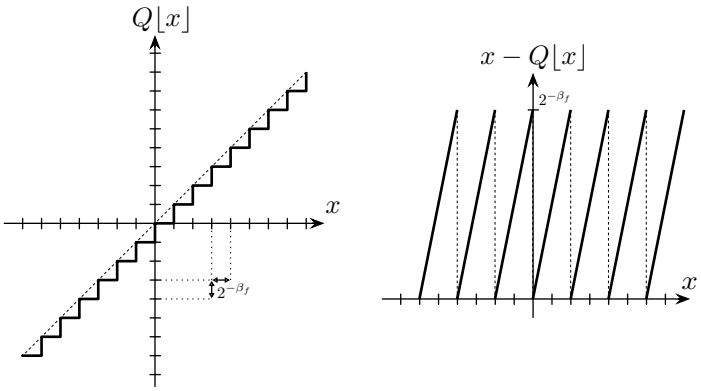


FIG. 2.20 – Loi de quantification par troncature et l'erreur de quantification

– La loi de quantification par arrondi consiste, elle, à choisir, pour représenter x, la valeur x^* la plus proche de x, représentable en virgule fixe sur N bits avec β_p bits significatifs de partie fractionnaire. Il s'agit d'un arrondi à la valeur la plus proche, et non à la valeur immédiatement inférieure (que l'on obtient en tronquant directement les bits superflus). On aura donc

$$x^* = 2^{-\beta_f} \left\lfloor 2^{\beta_f} x \right\rceil \tag{2.54}$$

(où $\lfloor . \rceil$ est l'opérateur d'arrondi à l'entier le plus proche) avec

$$|x - x^*| \leqslant 2^{-(\beta_f+1)} \tag{2.55}$$

Les caractéristiques de cette loi de quantification, notée $Q\lfloor . \rceil$, sont données à la figure 2.21.

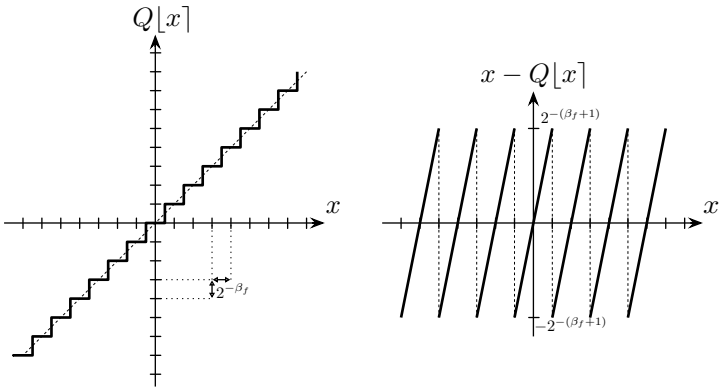


FIG. 2.21 – Loi de quantification par arrondi et l'erreur de quantification

2.2.3.3 Règles de l'arithmétique en virgule fixe

Les opérations entre deux opérandes en virgule fixe nécessitent de trouver une représentation commune aux deux opérandes, c'est à dire de trouver le paramètre β_f (la position de la virgule). Ensuite, les virgules sont alignées (par un décalage de bits, en étendant le bit de signe si nécessaire), ce qui peut entraîner une troncature ; et l'addition est réalisée [72]. Selon le processeur, l'arrondi peut s'effectuer avant ou après l'addition. La figure 2.22 montre un exemple d'addition de deux opérandes `a` et `b` ayant deux représentations différentes. Le code `C` permettant d'additionner `a` et `b` est

```
c = (a >> sa)+b ;
```

où `>>` est l'opérateur de décalage binaire.

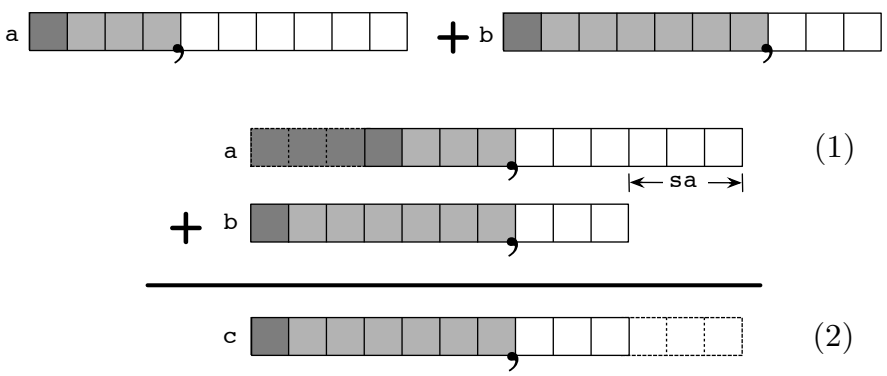


FIG. 2.22 – Addition `c=a+b` en virgule fixe ; (1) entraîne une extension de signe et (2) une troncature

Ces opérations sont, bien entendu, à faire réaliser par le compilateur qui génère le code (toutefois, il n'existe pas de type *virgule fixe* dans le `C ANSI`, il faut donc faire appel à une bibliothèque particulière, telles celles existant avec les outils FRIDGE ou AUTOSCALER [58, 56] ; de même, sous Simulink, il faut utiliser Fixed Point Blockset [1]).

La multiplication se fait plus simplement : les deux opérandes sont multipliées comme s'il s'agissait de nombres entiers en complément à 2 (si le processeur ne possède pas d'unité *hardware* de multiplication, il faut alors utiliser l'algorithme de *Booth* pour effectuer le calcul [111, 13]) et la position de la virgule s'obtient à partir des deux positions des opérandes (on notera que le bit de signe est doublé). La figure 2.23 illustre cette multiplication.

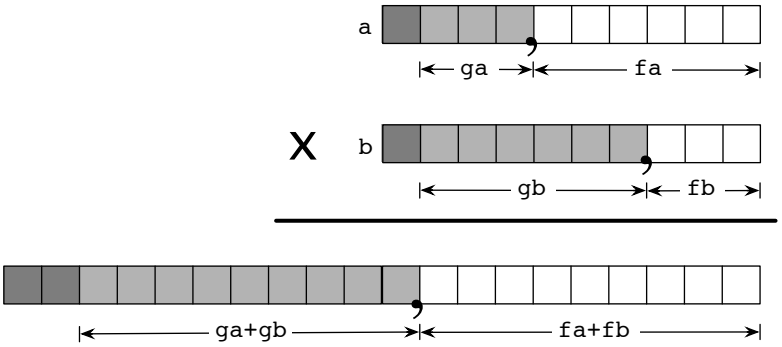


FIG. 2.23 – Multiplication en virgule fixe

Quant à la division, de nombreux algorithmes permettant d'obtenir le quotient et le reste existent, mais aucun ne permet une division aussi rapide qu'une multiplication : ainsi, pour mettre en place une loi de contrôle/commande, il faudra éviter le plus possible les divisions et les convertir en multiplications (en multipliant par l'inverse).

2.2.3.4 Quantification

Nous avons vu que, du fait de la représentation en précision finie, tous les réels ne sont pas représentables logiciellement. Une quantification apparaît lorsque l'on code des coefficients constants et lorsque l'on calcule toute opération.

La *quantification de représentation* intervient lorsque l'on désire représenter sur N bits virgule fixe un réel x constant : il faut alors trouver une position de la virgule (le facteur d'échelle 2^{β_f}) ainsi que les N bits correspondants qui forment x^* le quantifié de x.
Pour que la partie entière de x puisse être représentée sur les $N-1-p$ bits qui lui sont affectés, β_f doit vérifier

$$\beta_f \leqslant N - 1 - \lceil \log_2 |x| \rceil \tag{2.56}$$

(où $\lceil . \rceil$ représente l'arrondi à l'entier supérieur). β_f pourra être choisi le plus grand possible (donc $= N - 1 - \lceil \log_2 |x| \rceil$), pour garder le plus de bits significatifs et dégrader le moins possible x, ou bien pourra être choisi en commun pour un ensemble de coefficients, afin d'avoir une représentation commune et de ne pas devoir recourir à des changements de représentations lors des calculs, pour en diminuer le coût (cf. section 3.4 pour la définition de blocs de calculs).

De plus, pour représenter au mieux ces coefficients, la quantification par arrondi sera toujours utilisée (elle permet, par son arrondi au plus juste, de minimiser l'erreur de représentation).
La quantification de représentation est la cause d'une modification d'une loi de contrôle/commande lors de l'implémentation, par l'introduction d'erreurs paramétriques.

En plus de cette perturbation paramétrique, chaque affectation, chaque opération arithmétique peut provoquer un arrondi de calcul (comme vu au chapitre précédent, il faut souvent opérer un changement de représentation après une opération, et donc arrondir une valeur) : c'est la *quantification de calcul*.
Mais, contrairement au cas des coefficients constants que l'on quantifiait pour les coder dans une représentation donnée, la quantification sera effectuée pendant les calculs, et on utilisera donc la quantification par troncature, car celle-ci se réalise très facilement en tronquant les bits superflus (décalage de bits).
Remarque : la loi de quantification par arrondi est aussi possible en logiciel, en rajoutant une étape de calcul supplémentaire avant le décalage de bits : il s'agit de prendre en compte le bit de poids le plus fort parmi les bits de poids faible que l'on supprime. La valeur de ce bit nous indique si la valeur représentée est plus proche de la valeur représentable immédiatement supérieure ou immédiatement inférieure. Il suffit donc de rajouter la valeur de ce bit au nombre tronqué.
Par exemple, si l'on reprend le nombre π

$$011_{\vartriangle}00100100001111110110101010001000\ldots\ldots$$

et que l'on ne garde que 8 bits, $011_{\vartriangle}00100$ est obtenu après troncature et les bits et les bits $100001111\ldots\ldots$ sont perdus : le plus fort d'entre eux vaut 1, ce qui signifie que la valeur considérée est plus proche de $011_{\vartriangle}00101$ que de $011_{\vartriangle}00100$. La quantification par arrondi donnera donc la valeur $011_{\vartriangle}00101$, obtenue en rajoutant la valeur du bit de poids le plus fort des bits tronqués à la valeur.

L'étude de la propagation des erreurs d'arrondis dans les calculs est, depuis de nombreuses années, une thématique de recherche bien développée ([20, 71]). Il existe quatre grandes approches au problème d'estimation des erreurs de *quantification de calcul* :

- l'approche régressive, ou inverse ;
- l'approche directe ;
- l'approche déterministe (arithmétique par intervalle) ;
- l'approche statistique (analyse des bruits d'arrondi).

Les deux premières sont basées sur une analyse mathématique *a priori* des erreurs d'arrondi, alors que les deux dernières utilisent une évaluation *a posteriori* de la précision pour l'estimer ou la majorer. Seule l'approche

statistique permet vraiment d'évaluer la qualité numérique d'un résultat. On distingue aussi les méthodes basées sur la simulation (Cestac [101] par exemple) et les méthodes analytiques ([71]).

L'approche statistique est basée sur les travaux de Widrow ([108, 109]) et ceux de Sripad et Snyder ([98]) qui nous permettent, sous certaines conditions généralement réunies d'excitabilité suffisante de x (condition sur la fonction caractéristique[24] de x), de modéliser le processus de quantification (par arrondi ou troncature) par un système linéaire où le signal quantifié est égal à la somme du signal d'origine et d'un bruit dit *de quantification* (voir [71] et [111] pour plus de détails). Le bruit de quantification e est uniformément distribué et décorrelé de x (cf. conditions [98]). Il est caractérisé par ses moments d'ordre 1 (noté ici μ_e) et d'ordre 2 (σ_e), que l'on retrouve dans le tableau 2.4

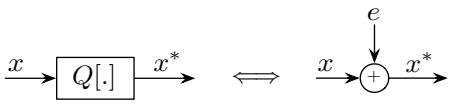


FIG. 2.24 – Modélisation du processus de quantification

	Arrondi	Troncature
e	$-2^{-(\beta_f+1)} \leqslant e \leqslant 2^{-(\beta_f+1)}$	$0 \leqslant e \leqslant 2^{-\beta_f}$
μ_e	0	$2^{-(\beta_f+1)}$
σ_e	$\frac{2^{-\beta_f}}{12}$	$\frac{2^{-\beta_f}}{12}$

TAB. 2.4 – Caractéristique du bruit de quantification

Dans le cas de la troncature, l'erreur est un bruit blanc non centré.

2.2.4 Virgule flottante

2.2.4.1 Description de la représentation

L'arithmétique flottante, née dans les années 70 est normalisée depuis 1985 sous les normes IEEE754 puis IEEE854.
Les représentations en virgule flottante sont réalisées en base θ (supposé pair) et avec une précision β_m, et utilisent la forme *mantisse/exposant* :

$$(-1)^s \cdot m \cdot \theta^e$$

[24] La fonction caractéristique d'une variable aléatoire est égale à la transformée inverse de sa densité de probabilité.

où la mantisse m s'écrit avec β_m digits, avec $0 \leqslant m < \beta$

$$m = \sum_{i=0}^{\beta_m - 1} m_i \theta^{-i}$$

et où l'exposant e s'écrit avec β_e digits, en complément à 2 :

$$e \in [\![-\theta^{\beta_e - 1} - 1, \theta^{\beta_e - 1} - 2]\!]$$

car deux valeurs de l'exposant, les valeurs maximales et minimales, ont une signification particulière.
On peut normaliser l'écriture en imposant $\frac{\theta}{2} \leqslant m < \theta$, ce qui permet de gagner un bit dans l'écriture de m (ce bit devient implicite).
De la même manière que pour la virgule fixe, un réel x est représenté en virgule flottante par

$$x^* = \begin{cases} 2^{e-\beta_m} \lfloor 2^{\beta_m - e} x \rceil & \text{quantification par arrondi} \\ 2^{e-\beta_m} \lfloor 2^{\beta_m - e} x \rfloor & \text{quantification par troncature} \end{cases} \tag{2.57}$$

Le standard IEEE754 impose $\theta = 2$, et le standard IEEE854 $\theta = 2$ ou $\theta = 10$. $\theta = 2$ sera pratiquement toujours utilisé. Ils offrent quatre précisions différentes de calcul : *simple*, *simple étendue*, *double* et *double étendue*

Paramètres	*simple*	*simple étendue*	*double*	*double étendue*
β_m	23	31	52	63
β_e	8	$\leqslant$11	11	15

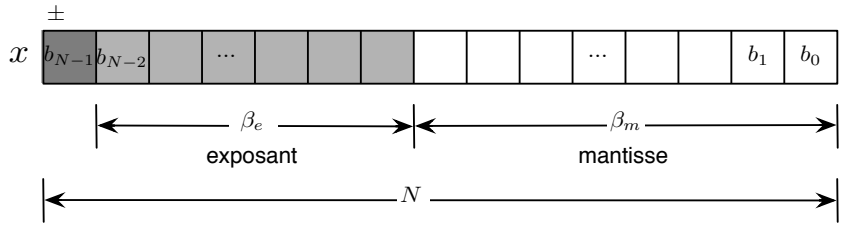


FIG. 2.25 – Représentation des données en virgule flottante

Le standard rajoute quelques valeurs particulières, notamment pour pouvoir exprimer $\pm\infty$ et pour les calculs donnant un résultat incorrect (la division par zéro renvoie *NaN*[25]).

En intégrant un exposant (dont la valeur peut varier) dans sa représentation, la forme virgule flottante permet de coder aussi bien les très faibles valeurs que les plus grandes. La mantisse permet d'exprimer la *précision* d'une

[25] N*ot* a N*umber*.

valeur, tandis que l'exposant exprime le facteur d'échelle. La représentation en virgule flottante présente donc une grande dynamique de représentation (bien plus que la virgule fixe, dont la dynamique est figée).

Les calculs en virgule flottante sont gérés par les unités matérielles de calcul flottant, ou bien simulés, si ces unités n'existent pas, avec un surcoût de calcul important.

On se référera par exemple à [32, 26, 63] pour plus de détails sur la virgule flottante.

2.2.4.2 Précision numérique

Bien que présentant une dynamique de représentation des nombres bien plus importante que pour la virgule fixe, les représentations et les calculs peuvent eux aussi être *non exacts* en virgule flottante. On attribue trop souvent aux calculs en virgule flottante des bonnes propriétés numériques, à savoir

- si le résultat d'un calcul ne peut être exact, il est certainement suffisamment proche du résultat exact (à la précision de la machine près) ;
- quelques opérations en virgule flottante ne peuvent induire qu'une légère imprécision sur le résultat.

L'exemple de J-M. Muller [78, 75] est un flagrant contre-exemple à ces idées. Si on considère la suite $(a_n)_{n\in\mathbb{N}}$ définie par

$$a_0 = \frac{11}{2}, a_1 = \frac{61}{11}, a_{n+1} = 111 - \frac{1130 - \frac{3000}{a_{n-1}}}{a_n} \tag{2.58}$$

On montre facilement que

$$a_n = \frac{6^{n+1} + 5^{n+1}}{6^n + 5^n} \tag{2.59}$$

et que $\lim_{n\to\infty} a_n = 6$

Or, si l'on calcule les termes (a_n) en utilisant la virgule flottante (simple ou double précision), cette suite converge numériquement très rapidement (dès le 10 ou 20$^{\text{e}}$ terme) vers 100.

Même avec peu d'opérations, les calculs en virgule flottante peuvent être tout à fait faux (l'ordre de grandeur n'étant pas respecté). En réalité, ce phénomène s'explique par la présence de trois points fixes 5, 6 et 100 à la fonction

$$x \mapsto 111 - \frac{1130 - \frac{3000}{x}}{x} \tag{2.60}$$

On peut donc conclure que les calculs en virgule flottante ne sont pas parfaits ou presque, comme on a trop tendance à le penser... Leur représentation est la plus adaptée pour manipuler des nombres de magnitudes

diverses, mais leurs calculs sont aussi entachés d'erreurs, souvent faibles, mais dont il faut avoir conscience (on se référera à [10], [32] et [75] pour plus de détails).
On notera tout de même que la propriété d'*arrondi correct* requise par la norme IEEE impose que les opérations flottantes $+$, $-$, $/$, $\times$ et $\sqrt{}$ rendent comme résultat l'arrondi du résultat exact, suivant un mode d'arrondi prédéfini [26].

De plus, contrairement à la virgule fixe où la quantification transforme x en $x + \delta$ avec $|\delta| \leqslant 2^{-(\beta_m+1)}$ (quantification par arrondi) ou $|\delta| \leqslant 2^{-\beta_m}$ (quantification par troncature), la quantification en virgule flottante transforme x en $x(1 + \delta)$ avec $|\delta| \leqslant 2^{-\beta_m}$ (ou $|\delta| \leqslant 2^{-(\beta_m+1)}$).

2.2.4.3 Comparaison *Virgule fixe* et *Virgule flottante*

De nombreuses architectures numériques sont utilisées pour mettre en œuvre les systèmes d'asservissement, de régulation ou de traitement du signal (micro-contrôleurs, microprocesseurs, processeurs généralistes, processeurs de traitement du signal[26], etc...). Mais les calculs qui y sont réalisés ne le sont que de deux manières : les calculs en virgule fixe (lorsque les grandeurs sont représentées en virgule fixe) et les calculs en virgule flottante.
On a vu précédemment que la représentation en virgule flottante présentait bien des avantages :

- sa dynamique est bien plus importante que celle de la virgule fixe (pour le même nombre de bits, virgule fixe et virgule flottante codent le même nombre de réels, mais la représentation en virgule flottante permet de représenter aussi bien des grandes valeurs que de très petites valeurs, ce que ne permet pas la virgule fixe, car son pas de quantification est fixe) ;
- les calculs en virgule flottante ne nécessitent pas de travaux préalables, comme la position de la virgule ou les gestions des *overflow* qui sont nécessaires en virgule fixe.

On peut donc se demander l'intérêt que peut encore présenter la virgule fixe. Pourtant cette représentation possède des atouts indéniables qui font qu'elle est toujours très utilisée [1, 86, 23] :

- **Généricité :** les unités de calcul en entier (et donc en virgule fixe) sont présentes sur chaque processeur, tandis que les unités d'arithmétique flottante ne le sont pas partout (de nombreux DSP n'ont pas d'unités de calcul flottant) ;
- **Taille et consommation énergétique :** Les circuits logiques virgule fixe sont beaucoup moins compliqués que ceux en virgule flottante : les processeurs en virgule fixe sont donc plus petits et moins gourmands

[26] D*igital* S*ignal* P*rocessors*.

énergétiquement. Ce critère est très important dans les applications embarquées où l'autonomie est essentielle. Par exemple [30] montre un rapport 4 entre les consommations énergétiques d'une même transformée inverse en cosinus discrète en virgule fixe et en virgule flottante ;
- **Rapidité des calculs :** la simplicité des circuits pour les différentes opérations amène à une exécution des calculs bien plus rapide en virgule fixe qu'en virgule flottante [86] ;
- **Coût :** produits en grande série, les processeurs sans arithmétique flottante sont moins chers qu'avec unité de calculs flottants[27].

2.2.4.4 Problématique du passage de virgule flottante à virgule fixe

La méthodologie de conversion d'algorithmes (principalement traitement du signal, d'images, ou de commande) s'exécutant en virgule flottante vers des algorithmes utilisant des calculs virgule fixe est une problématique intéressante qui a fait l'objet de nombreuses recherches. En effet, les algorithmes sont souvent développés et testés sur des architectures matérielles puissantes (ordinateur de bureau) disposant d'unités de calcul flottant, puis souvent transférés sur des calculateurs ne possédant que des unités de calcul entier (donc virgule fixe).
Une première problématique vise à définir, lors de la conversion, le format de donnée adapté pour chaque variable intervenant dans le calcul ([23, 22]) : cela consiste principalement à trouver automatiquement la position de la virgule (facteur d'échelle) de chaque variable pour assurer la meilleure précision possible en évitant les overflows et underflows ; cela est généralement obtenu par exécution de l'algorithme sur un ensemble de jeu d'essais représentatifs permettant de déterminer la dynamique des variables (l'utilisation d'une arithmétique par intervalle peut, dans certains cas, ne pas suffire [88], d'où l'utilisation de jeux d'essais pour s'approcher d'une étude statistique). On se référera aux projets FRIDGE[28] [56] et Autoscaler [58, 61].
De plus, la problématique d'évaluation automatique de la précision d'un algorithme écrit en virgule flottante (évaluation du rapport Signal à Bruit de Quantification) est abordée dans [71, 73, 74].

[27] Ce facteur est bien entendu déterminant pour bien des calculateurs embarqués dans des voitures PSA car chaque véhicule est produit en grande série : l'économie réalisée avec des calculateurs en virgule fixe peut donc être substantielle.
[28] F*ixed-point* p*R*o*grammIng* D*esi*G*n* E*nvironment*

2.3 Autres solutions *logicielles*

En plus des différentes paramétrisations possibles pour réaliser une loi, dont découle l'équation la décrivant, et en plus du choix du format de représentation, qui impacte sur la précision des coefficients et des calculs, il existe un troisième degré de liberté dans la mise en œuvre *logicielle* d'une loi.
En effet, les possibilités offertes par le logiciel pour effectuer les calculs nécessaires avec le format de données choisi sont aussi multiples.

On notera tout d'abord qu'il est possible d'effectuer certains calculs intermédiaires avec une précision plus grande que celle fournie avec le format de représentation des coefficients choisi, en simulant, par exemple, des calculs sur un nombre de bits supérieur à celui du processeur utilisé. Cette technique est par exemple utilisée pour mettre en œuvre les formes directes car elle permet d'augmenter la dynamique des variables intermédiaires et d'éviter les *overflow*.

Il est aussi possible d'améliorer la précision de certains calculs par quelques artifices : une perte de précision survient lorsqu'un résultat est quantifié avant d'être stocké (quantification de calcul).
Une 1re technique, assez peu coûteuse, consiste alors à ne pas utiliser une quantification par troncature (qui est réalisée nativement par le processeur) mais à pratiquer en *logiciel* une quantification par arrondi[29] (cf. 2.2.3.4).
Enfin, une 2de technique repose sur le fait que les équations des réalisations sont des équations de récurrence utilisant les résultats précédemment calculés, quantifiés puis stockés. Cette quantification entraîne donc une perte de précision qu'il est pourtant possible de récupérer en la stockant aussi et en la réinjectant dans l'équation de calcul. Il est ainsi possible d'améliorer la précision du résultat.
Un exemple simple est donné par la réalisation d'un filtre du 2nd ordre avec une forme directe I et des coefficients supposés entiers :

$$Y(k) = \frac{1}{a_0}\left(b_0U(k) + b_1U(k-1) + b_2U(k-2) - a_1Y(k-1) - a_2Y(k-2)\right) \tag{2.61}$$

La division par a_0, et la quantification qui s'en suit, entraîne une perte de précision (on devrait d'ailleurs écrire $Y(k) = Q\left[\frac{T}{a_0}\right]$ pour bien voir la quantification).
Si on note $r(k)$ le reste de cette division par a_0 (obtenue par un modulo, ou en remultipliant le résultat $Y(k)$ par a_0, ou bien par l'unité de calcul du processeur qui peut donner le reste d'une division en même temps que le quotient), $r(k)$ représente l'*information* perdue par la quantification d'après division.

[29] On notera que certains calculateurs disposent en *matériel* de cette possibilité.

Comme il est possible de stocker cette valeur, nous pouvons la réinjecter dans l'équation de récurrence, pour augmenter la précision de représentation de $Y(k-1)$ et $Y(k-2)$ qui interviennent dans ce calcul. On écrit donc l'algorithme suivant :

1: $T = b_0U(k) + b_1U(k-1) + b_2U(k-2) - a_1Y(k-1) - a_2Y(k-2) - \frac{1}{a_0}(a_1r(k-1) + a_2r(k-2))$
2: $r(k) = T \bmod a_0$
3: $Y(k)) = Q\left[\frac{T}{a_0}\right]$

On retrouvera plus de détails sur cette technique de *réinjection du reste* dans [122] (dans ce cas précis de la forme directe I avec coefficients en entier, cette technique a fait l'objet d'un brevet pour une utilisation de filtrage dynamique [123]). D. Williamson a généralisé cette méthode à une réalisation d'état, sous le terme de *Residue Feedback* [110, 113, 111].

On notera enfin que de nombreuses techniques logicielles existent pour, non pas augmenter la précision de calcul, mais diminuer le coût du calcul associé à un ensemble d'opérations, sans en affecter le résultat[30] : pour écrire des divisions sous forme d'additions et de décalages [67], pour la multiplication matricielle [14], la division [80], etc.

2.4 Conclusion

Parcourant la littérature et les pratiques, nous avons vu, dans la 1re partie de ce chapitre, qu'il existait de nombreuses façons d'écrire les équations qui régissent un filtre ou un régulateur LTI. Ces réalisations, mathématiquement équivalentes, ne le sont plus dès lors que les calculs sont effectués en précision finie, et il sera intéressant, aux chapitres suivants, d'étudier le comportement en précision finie de chacune d'entre elles.
En plus de ce large choix de réalisations possibles, les modes de représentation des nombres (principalement virgule fixe et virgule flottante), ainsi que les techniques logicielles pour mettre en œuvre les équations issues de l'automatique, ont un rôle important (même si, loin d'avoir été exhaustifs, nous avons laissé de côté nombre de solutions logicielles imaginables). Pour étudier le vaste panel d'implémentations possibles, nous proposerons, au chapitre 3, un formalisme unificateur englobant les différentes réalisations vues ici, et bien d'autres encore.

[30] Mais au prix d'une possible augmentation de la complexité apparente, au détriment donc de la lisibilité ou la portabilité.

Chapitre

3 La forme implicite pour formalisme unificateur

Résumé :
Le but de ce chapitre est de proposer un formalisme unificateur – la forme implicite spécialisée – qui permet de décrire et de caractériser l'ensemble des réalisations possibles vues au chapitre précédent. Tout en restant *macroscopique* et suffisamment général, ce formalisme doit être davantage représentatif des implémentations existantes et offre une plus grande latitude dans l'expression des possibilités d'implémentation.
Nous verrons quelques exemples de réalisations reformulés à l'aide de ce formalisme et nous caractériserons l'ensemble des réalisations équivalentes.

Sommaire

3.1 Forme implicite

3.1.1 Les besoins d'un formalisme unificateur

Les diverses réalisations que nous avons détaillées au chapitre 2 (formes directes, espace d'état, opérateur δ, etc...) associées aux nombreuses possibilités logicielles de mise en œuvre montrent l'étendue des algorithmes possibles pour implémenter une loi de contrôle/commande dans un calculateur. Ces réalisations, toutes mathématiquement équivalentes, ne le sont plus en précision finie et amènent, par la voie des erreurs paramétriques et des bruits numériques induits, une détérioration de la loi. Ces réalisations ne sont pas non plus équivalentes quant à la détérioration amenée, certaines étant plus *sensibles* que d'autres à la transformation en précision finie, d'autres plus *robustes*.

Il est alors important de pouvoir décrire chacune de ces réalisations dans un même formalisme afin de pouvoir toutes les caractériser et les comparer, suivant des critères qu'il nous faudra établir (cf. chapitre 4). Ce formalisme devra permettre tout d'abord d'expliciter la paramétrisation choisie, c'est à dire l'ensemble des paramètres utilisés lors du calcul et qui caractérise la loi (les paramètres réellement utilisés dans le code doivent clairement apparaître car ce sont eux qui seront potentiellement tronqués). De plus, ce formalisme doit donner une relation directe avec le code produit afin de pouvoir évaluer la dégradation engendrée pour chaque calcul et donc de permettre la mesure précise de l'impact du passage en précision finie, et aider à élaborer une synthèse de réalisations numériques optimales vis à vis du passage en précision finie.

Mais ce formalisme doit aussi rester *macroscopique* : il doit être suffisamment global pour s'exprimer de manière simple et mathématique. Par exemple, un graphe de fluence détaillant chaque opération ou même un algorithme (indépendamment du langage dans lequel on l'écrit) serait trop microscopique pour que l'on puisse étudier l'impact de la quantification.
L'idée est donc d'avoir une forme mathématique, à l'image de la forme d'état, permettant de décrire macroscopiquement mais fidèlement le plus grand nombre de réalisations [44].

3.1.2 Expression de la forme implicite spécialisée

L'idée retenue pour décrire macroscopiquement une réalisation est de détailler les calculs, tout en restant sous forme matricielle. Pour cela, nous considérons, en plus du vecteur d'état $X(k)$, le vecteur de variables intermédiaires $T(k)$.

On se propose alors de décrire un algorithme de commande ou de filtrage

de la manière suivante (dans l'ordre) :

① Calcul des variables intermédiaires $T(k+1)$ de l'instant k[1], à partir de l'état $X(k)$ et des entrées $U(k)$

$$JT(k+1) = MX(k) + NU(k) \tag{3.1}$$

② Calcul de l'état de l'instant prochain $X(k+1)$, à partir de l'entrée $U(k)$, de l'état présent $X(k)$ et des variables intermédiaires que l'on vient de calculer à l'étape [1]

$$X(k+1) = KT(k+1) + PX(k) + QU(k) \tag{3.2}$$

③ Calcul de la sortie $Y(k)$ à partir de l'état $X(k)$, de l'entrée $U(k)$ et des variables intermédiaires calculées à l'étape [1]

$$Y(k) = LT(k+1) + RX(k) + SU(k) \tag{3.3}$$

L'ordre des étapes ② et ③ est interchangeable[2]. Ces trois étapes définissent la forme implicite spécialisée et peuvent s'écrire sous une forme d'état implicite matricielle :

Définition 3.1 (Forme Implicite Spécialisée)
La forme implicite spécialisée est une réalisation décrite sous la forme :

$$\begin{pmatrix} J & 0 & 0 \\ -K & I_n & 0 \\ -L & 0 & I_p \end{pmatrix} \begin{pmatrix} T(k+1) \\ X(k+1) \\ Y(k) \end{pmatrix} = \begin{pmatrix} 0 & M & N \\ 0 & P & Q \\ 0 & R & S \end{pmatrix} \begin{pmatrix} T(k) \\ X(k) \\ U(k) \end{pmatrix} \tag{3.4}$$

où

- $T(k) \in \mathbb{R}^l$ *est le vecteur de variables intermédiaires. Celles-ci ne sont pas stockées (ce qui est caractérisé par la colonne de* 0 *dans la matrice*

$$\begin{pmatrix} 0 & M & N \\ 0 & P & Q \\ 0 & R & S \end{pmatrix}$$

  de l'équation (3.4)) mais réutilisées (à travers les matrices K *et* L*) dans le même pas d'itération ;*
- $X(k) \in \mathbb{R}^n$ *est le vecteur d'état : il est calculé à l'avance à chaque itération et stocké pour être utilisé à l'itération suivante ;*
- $U(k) \in \mathbb{R}^m$ *est le vecteur d'entrées ;*

[1] Remarque : les variables intermédiaires de l'instant k sont notées $T(k+1)$ car elles se calculent dans le même pas de calcul que l'état $X(k+1)$; cela permet de les rassembler et d'avoir une écriture plus compacte : nous détaillerons ce point plus tard.

[2] On pourra même, dans la pratique, exécuter l'étape ③ avant l'étape ② afin de réduire le plus possible le temps entre l'acquisition des entrées et la production des sorties.

– $Y(k) \in \mathbb{R}^p$ *est le vecteur de sorties ;*
– *et* $J \in \mathbb{R}^{l\times l}$, $K \in \mathbb{R}^{n\times l}$, $L \in \mathbb{R}^{p\times l}$, $M \in \mathbb{R}^{l\times n}$, $N \in \mathbb{R}^{l\times m}$, $P \in \mathbb{R}^{n\times n}$, $Q \in \mathbb{R}^{n\times m}$, $R \in \mathbb{R}^{p\times n}$ *et* $S \in \mathbb{R}^{p\times m}$ *sont les matrices définissant la réalisation.*

De plus, J *est une matrice triangulaire inférieure avec des* 1 *sur la diagonale :*

$$J = \begin{pmatrix} 1 & 0 & \dots & \dots & 0 \\ \star & \ddots & 0 & & \vdots \\ \vdots & \star & 1 & 0 & \vdots \\ \vdots & & \star & \ddots & 0 \\ \star & \dots & \dots & \star & 1 \end{pmatrix} \tag{3.5}$$

Par convention, cette forme correspond à l'algorithme (ordonné)

1: $JT(k+1) \leftarrow MX(k) + NU(k)$
2: $X(k+1) \leftarrow KT(k+1) + PX(k) + QU(k)$
3: $Y(k) \leftarrow LT(k+1) + RX(k) + SU(k)$

Remarque : on notera qu'il est possible de considérer une réalisation sans aucune variable intermédiaire. On a alors $l = 0$, et les matrices J,K, L, M et N sont des matrices vides. En pratique, Matlab, par exemple, autorise de telles matrices, rendant ainsi valables toutes les expressions que l'on verra par la suite. Les réalisations exprimées sous forme d'état classique sont donc un sous-cas ($l = 0$) de forme implicite spécialisée.

Cette forme (définie par l'équation 3.4) est une forme implicite[3] car l'état et la sortie peuvent être calculés à partir de variables intermédiaires (calculées à la même itération mais non stockées).

De plus, lors du calcul des variables intermédiaires (équation (3.1)), la matrice J permet à une variable intermédiaire d'être calculée à partir d'une autre variable intermédiaire, qui vient d'être calculée au même pas. Sa forme triangulaire inférieure imposée autorise à obtenir un élément $T_i(k+1)$ à partir des éléments $(T_j(k+1))_{j<i}$.
Ainsi dans l'étape ①, le calcul est **ordonné** (on calcule le premier élément

[3] Pour rappel, une réalisation linéaire implicite[6] est de la forme

$$\bar{E}\begin{pmatrix} Z(k+1) \\ Y_k \end{pmatrix} = \begin{pmatrix} \bar{A} & \bar{B} \\ \bar{C} & \bar{D} \end{pmatrix}\begin{pmatrix} Z(k) \\ U(k) \end{pmatrix} \tag{3.6}$$

Et dans le cas où $\bar{E} = \begin{pmatrix} \bar{E}_{11} & 0 \\ \bar{E}_{22} & I \end{pmatrix}$, le système est dit singulier (et ne peut donc s'implémenter aisément) si et seulement si $\bar{E}_{11}$ est singulière [24].

du vecteur colonne $T(k)$, puis le second, etc...), et le calcul se réalise sans inverser J selon l'algorithme suivant :

1: **Pour** i **de** 1 **à** l **faire**
2: $\quad T_i(k+1) \leftarrow M_{i,\bullet}X(k) + N_{i,\bullet}U(k) - \sum_{j=1}^{j<i} J_{i,j}T_j(k+1)$
3: **FinPour**

Sur un exemple plus simple, avec $J = \begin{pmatrix} 1 & 0 \\ \alpha & 1 \end{pmatrix}$, et $M = \begin{pmatrix} M_1 \\ M_2 \end{pmatrix}$, $N = \begin{pmatrix} N_1 \\ N_2 \end{pmatrix}$, l'algorithme de calcul de l'étape [1] s'écrit :

1: $T_1(k+1) \leftarrow M_1X(k) + N_1U(k)$
2: $T_2(k+1) \leftarrow M_2X(k) + N_2U(k) - \alpha.T_1(k+1)$

La matrice J permet donc la décomposition du calcul en plusieurs étapes qui s'enchaînent et la réutilisation de résultats précédemment calculés. Les exemples des paragraphes 3.2.4 et 3.2.3.2 montrent l'utilisation en pratique de la matrice J pour décrire un enchaînement de calculs dans un algorithme de contrôle/commande. On remarquera que l'inverse de J n'a pas à être calculée et que les coefficients utilisés dans le calcul sont ceux de J (au signe près).

De plus, il est important de noter que, bien qu'ils soient décrits sous une forme matricielle, les calculs ne sont pas forcément codés en logiciel sous forme matricielle : certaines matrices peuvent êtres nulles ou égales à l'identité (une opération matricielle serait alors inutile) et, plus généralement, la réalisation informatique peut procéder à un calcul équivalent (avec les mêmes paramètres), mais ne nécessitant pas les itérations nécessaires à une multiplication matricielle.

Notons enfin qu'il est possible de rendre explicite cette forme implicite spécialisée, qui n'est, par construction, pas singulière :

Proposition 3.1 (Forme explicite équivalente)
La forme implicite spécialisée (3.4) peut s'expliciter

$$\left(\begin{array}{c} T(k+1) \\ X(k+1) \\ \hline Y(k) \end{array}\right) = \left(\begin{array}{cc|c} 0 & J^{-1}M & J^{-1}N \\ 0 & A & B \\ \hline 0 & C & D \end{array}\right) \left(\begin{array}{c} T(k) \\ X(k) \\ \hline U(k) \end{array}\right) \tag{3.7}$$

avec $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$ *et* $D \in \mathbb{R}^{p\times m}$ *définis par*

$$A = KJ^{-1}M + P \tag{3.8}$$
$$B = KJ^{-1}N + Q \tag{3.9}$$
$$C = LJ^{-1}M + R \tag{3.10}$$
$$D = LJ^{-1}N + S \tag{3.11}$$

La fonction de transfert H *du système considéré est alors définie par*

$$H(z) = C\left(zI_n - A\right)^{-1}B + D \qquad \forall z \in \mathbb{C} \tag{3.12}$$

On notera que la paramétrisation est changée dans la forme explicitée : les deux réalisations (3.4) et (3.7) sont mathématiquement équivalentes, mais ne le sont plus en précision finie.

On remarquera qu'il aurait pu être intéressant, pour augmenter la représentativité de la forme implicite (3.4), de considérer, de la même manière que nous l'avons considéré pour les variables intermédiaires, la possibilité d'écrire le calcul de $X(k+1)$ sous une forme implicite

$$EX(k+1) = KT(k+1) + PX(k) + QU(k) \tag{3.13}$$

avec E triangulaire inférieur (avec une diagonale unitaire).
Mais on peut facilement se ramener à la forme proposée en réalisant le calcul (3.13) dans une seconde variable intermédiaire T_2 (le calcul de $X(k+1)$ se ramène alors à $X(k+1) \leftarrow T_2(k+1)$) :

$$\left(\begin{array}{ccc:c} J & 0 & 0 & 0 \\ -K & E & 0 & 0 \\ \hdashline 0 & -I_n & I_n & 0 \\ \hdashline -L & 0 & 0 & I_p \end{array}\right) \begin{pmatrix} T_1(k+1) \\ T_2(k+1) \\ X(k+1) \\ Y(k) \end{pmatrix} = \left(\begin{array}{cc:cc} 0 & 0 & M & N \\ 0 & 0 & P & Q \\ \hdashline 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & R & S \end{array}\right) \begin{pmatrix} T_1(k) \\ T_2(k) \\ X(k) \\ U(k) \end{pmatrix} \tag{3.14}$$

Cela augmente, de manière artificielle, la taille du vecteur de variables intermédiaires, mais n'introduit pas de paramètres additionnels (on verra à la section 3.4.3 que les matrices nulles ou identités ne seront pas considérées comme des paramètres signifiants subissant une quantification : ce sont des paramètres *transparents*). L'intérêt de cette écriture réside dans la simplicité de formulation des développements ultérieurs. Le paragraphe 5.3.3 montrera une réalisation avec une matrice E, différente de l'identité, intéressante.

3.1.3 Définitions

Bien que M. Gevers et G. Li [31] ne fassent pas de différence entre les termes *réalisations*, *paramétrisations* et *représentations*, il peut être utile ici de les distinguer et de les préciser, afin de caractériser un ensemble de réalisations possibles pour une même loi LTI [43].

Définition 3.2 (Réalisation)
*Une **réalisation** $\mathcal{R}$ est définie par les valeurs spécifiques des matrices J, K, L, M, N, P, Q, R et S utilisées pour la description interne de la forme implicite (équation (3.4))*

$$\mathcal{R} :\triangleq (J, K, L, M, N, P, Q, R, S) \tag{3.15}$$

Une réalisation $\mathcal{R}$ est une réalisation de la fonction de transfert H si H et $\mathcal{R}$ ont la même relation entrées/sorties (en partant de conditions initiales nulles).

On remarquera aussi que l'on peut rassembler ces matrices en une seule matrice $Z \in \mathbb{R}^{(l+n+p)\times(l+n+m)}$ définie par

$$Z \triangleq \begin{pmatrix} -J & M & N \\ K & P & Q \\ L & R & S \end{pmatrix} \tag{3.16}$$

L'utilisation de cette matrice permettra, dans les chapitres suivants, de simplifier grandement l'écriture des mesures de certaines équations. On notera le signe "$-$" avant le J, qui sera justifié par la suite (proposition 4.6).
Ainsi, une réalisation $\mathcal{R}$ peut aussi être définie par la seule donnée de la matrice Z et des dimensions l, m, n et p (respectivement taille du vecteur de variables intermédiaires, nombre d'entrées, taille du vecteur d'état et nombre de sorties)

$$\mathcal{R} := (Z, l, m, n, p) \tag{3.17}$$

par la suite, nous utiliserons indifféremment les matrices J, K, L, M, N, P, Q, R, S ou la matrice Z, suivant les cas, avec une préférence pour cette dernière car elle permet de représenter, de manière compacte, une réalisation.

On notera *dimension d'une réalisation* la valeur des dimensions l, m, n et p de cette réalisations.

Définition 3.3 (Ensemble de réalisations équivalentes)
*On note $\mathscr{R}_H$ l'ensemble des réalisations non singulières de la fonction de transfert H. Ces réalisations sont dites **équivalentes***

De plus, on peut aussi considérer un ensemble de réalisations où certaines matrices sont contraintes à prendre certaines valeurs. Les réalisations sont alors dites structurées.

Définition 3.4 (Structuration)
*Une **structuration** $\mathscr{S}$ est un ensemble de réalisations avec une certaine structure : certains coefficients ou certaines dimensions des matrices J, K, L, M, N, P, Q, R et S sont alors fixés* a priori.

Par exemple, les réalisations sous forme d'état classique (explicites car n'utilisant pas de variables intermédiaires) sont les réalisations pour lesquelles $K = 0_{n\times l}$ et $L = 0_{p\times l}$.

$$\begin{pmatrix} J & 0 & 0 \\ 0 & I_n & 0 \\ 0 & 0 & I_p \end{pmatrix} \begin{pmatrix} T(k+1) \\ X(k+1) \\ Y(k) \end{pmatrix} = \begin{pmatrix} 0 & M & N \\ 0 & A_q & B_q \\ 0 & C_q & D_q \end{pmatrix} \begin{pmatrix} T(k) \\ X(k) \\ U(k) \end{pmatrix} \tag{3.18}$$

Bien entendu, les états $T(k)$ étant ici inutiles, on se ramènera toujours au cas $l = 0$ (les matrices J, K, L, M et N étant vides).
Ainsi, la q-structuration, l'ensemble des réalisations sous forme d'état (avec l'opérateur avance q) n'utilisant pas de variables intermédiaires ($l = 0$), est définie par

$$\mathscr{S}_q \triangleq \left\{ \mathcal{R} \setminus \mathcal{R} = (.,.,.,.,.,A_q,B_q,C_q,D_q)\,, \quad \forall (A_q,B_q,C_q,D_q) \right\} \tag{3.19}$$

De même, les réalisations écrites sous forme d'état *mais* avec l'opérateur δ décrites par l'équation (2.21) correspondent à une structuration bien particulière. Ces réalisations, conduisant à l'algorithme suivant (voir le paragraphe 2.1.6) :

1: $T \leftarrow A_\delta X(k) + B_\delta U(k)$
2: $X(k+1) \leftarrow X(k) + \Delta T$
3: $Y(k) \leftarrow C_\delta X(k) + D_\delta U(k)$

sont réalisées avec la forme implicite spécialisée :

$$\begin{pmatrix} I_n & 0 & 0 \\ -\Delta I_n & I_n & 0 \\ 0 & 0 & I_p \end{pmatrix} \begin{pmatrix} T(k+1) \\ X(k+1) \\ Y(k) \end{pmatrix} = \begin{pmatrix} 0 & A_\delta & B_\delta \\ 0 & I_n & 0 \\ 0 & C_\delta & D_\delta \end{pmatrix} \begin{pmatrix} T(k) \\ X(k) \\ U(k) \end{pmatrix} \tag{3.20}$$

Donc la structuration en δ, l'ensemble des réalisations en δ explicites, originellement décrites à l'aide de l'opérateur δ, est définie par

$$\mathscr{S}_\delta \triangleq \left\{ \mathcal{R} \setminus \mathcal{R} = (I_n,\Delta I_n,0,A_\delta,B_\delta,I_n,0,C_\delta,D_\delta)\,, \quad \forall (A_\delta,B_\delta,C_\delta,D_\delta,\Delta) \right\} \tag{3.21}$$

Ainsi, on pourra remarquer qu'une réalisation en δ (une réalisation exprimée avec l'opérateur δ) peut s'écrire, sans changer la paramétrisation, comme une réalisation *non minimale* utilisant l'opérateur q, écrite sous forme implicite

Définition 3.5 (Réalisation structurée)
On notera $\mathscr{R}_H^{\mathscr{S}}$ l'ensemble des réalisations d'une fonction de transfert H selon une structuration $\mathscr{S}$

$$\mathscr{R}_H^{\mathscr{S}} \triangleq \mathscr{R}_H \cap \mathscr{S} \tag{3.22}$$

Une réalisation de $\mathscr{R}_H^{\mathscr{S}}$ sera appelée ***réalisation structurée de*** *H, ou encore réalisation de H avec une structuration en $\mathscr{S}$.*

Le paragraphe 3.2 explicite la structuration de différentes structures génériques importantes, au moyen de la forme implicite spécialisée.

Définition 3.6 (Paramétrisation)
On nommera ***paramétrisation*** *d'une réalisation $\mathcal{R}$ l'ensemble des paramètres (coefficients, éléments) de J, K, L, M, N, P, Q, R et S qui sont des paramètres signifiants pour la réalisation.*

Par exemple, pour une réalisation structurée en q (équation (3.19)), la paramétrisation est définie par les données des matrices A_q, B_q, C_q et D_q, tandis que pour une réalisation structurée en δ, la paramétrisation est définie par les matrices A_δ, B_δ, C_δ et D_δ et du paramètre Δ (cf. équation (3.20) par exemple).

3.2 Exemples de structuration

Dans cette section, nous reprenons les différents exemples de réalisations possibles exhibés au paragraphe 2.1 (et en introduisons d'autres), pour les réécrire avec la forme implicite spécialisée et montrer que celle-ci permet de les représenter de manière réaliste et unifiée, en exhibant la paramétrisation utilisée.

3.2.1 Forme d'état

La forme d'état classique (cf. description section 2.1.2) peut, bien entendu, s'écrire avec la forme implicite spécialisée. La réalisation

$$\left\{\begin{array}{rcl} X(k+1) & = & A_qX(k)+B_qU(k) \\ Y(k) & = & C_qX(k)+D_qU(k) \end{array}\right. \tag{3.23}$$

correspond à une forme implicite où $l=0$

$$\begin{pmatrix} . & . & . \\ . & I_n & 0 \\ . & 0 & I_m \end{pmatrix} \begin{pmatrix} T(k+1) \\ X(k+1) \\ Y(k) \end{pmatrix} = \begin{pmatrix} . & . & . \\ . & A_q & B_q \\ . & C_q & D_q \end{pmatrix} \begin{pmatrix} T(k) \\ X(k) \\ U(k) \end{pmatrix} \tag{3.24}$$

ou encore

$$Z = \begin{pmatrix} . & . & . \\ . & A_q & B_q \\ . & C_q & D_q \end{pmatrix} \tag{3.25}$$

3.2.2 Forme directe I

La forme directe I (cf. section 2.1.4.1) est décrite par l'équation

$$Y(k) = \frac{1}{a_0}\left(\sum_{i=0}^{n} b_i U(k-i) - \sum_{i=1}^{n} a_i Y(k-i)\right) \forall k > n \tag{3.26}$$

Les $2n$ valeurs $(U(k-i))_{1\leqslant i\leqslant n}$ et $(Y(k-i))_{1\leqslant i\leqslant n}$ composent l'état (l'ordre n'a pas d'importance) :

$$X(k) = \begin{pmatrix} U(k-1) \\ \vdots \\ U(k-n) \\ \hline Y(k-1) \\ \vdots \\ Y(k-n) \end{pmatrix} \tag{3.27}$$

De plus, la valeur de $Y(k)$ calculée d'après l'équation (3.26) doit aussi être stockée pour le pas suivant (dans $X(k+1)$). Dans la réalité, cette valeur n'étant pas calculée deux fois (une fois pour l'état $X(k+1)$ et une fois pour $Y(k)$), elle est calculée au préalable dans une variable intermédiaire[4] T, puis utilisée pour l'expression de $Y(k)$ et pour $X(k+1)$:

$$1:\quad T \leftarrow \frac{1}{a_0}\left(\sum_{i=1}^{n} b_i X_i(k) - \sum_{i=1}^{n} a_i X_{n+i}(k) + b_0 U(k)\right)$$

ou encore

$$1:\quad T \leftarrow \frac{1}{a_0}\left(\Gamma_1 X(k) + b_0 U(k)\right)$$

avec $\Gamma_1 = \left(\begin{array}{cccc|cccc} b_1 & \cdots & \cdots & b_n & -a_1 & \cdots & \cdots & -a_n \end{array}\right)$.
Le calcul de l'état $k+1$ sert juste à exprimer la progression du temps : l'entrée courante est stockée à la place de $U(k-1)$; $U(k-1)$ prend la place de $U(k-2)$, etc... ; il en est de même avec les $(Y(k-i))_{1\leqslant i\leqslant n-1}$:

[4] On supposera qu'il est équivalent, en précision finie (comme en précision infinie) de calculer T avec une soustraction faisant intervenir les coefficients $(-a_i)_{1\leqslant i\leqslant n}$ ou bien avec une addition et les coefficients $(a_i)_{1\leqslant i\leqslant n}$ (on assimilera donc l'opposé de la quantification d'un réel à la quantification de l'opposé de ce réel, ce qui est faux dans le cas général). Ceci revient à dire qu' l'on peut écrire ce calcul de la variable intermédiaire comme suit :

$$T \leftarrow \frac{1}{a_0}\left(\sum_{i=1}^{n} b_i X_i(k) + \sum_{i=1}^{n} -a_i X_{n+i}(k) + b_0 U(k)\right) \tag{3.28}$$

1: $X(k+1) \leftarrow \Gamma_2 X(k) + \Gamma_3 U(k) + \Gamma_4 T$

avec

$$\Gamma_2 = \left(\begin{array}{ccccc|ccccc} 0 & & & & & & & & & \\ 1 & \ddots & & & & & & & & \\ & \ddots & \ddots & & & & & & & \\ & & 1 & 0 & & & & & & \\ \hline & & & & & 0 & & & & \\ & & & & & 1 & \ddots & & & \\ & & & & & & \ddots & \ddots & & \\ & & & & & & & 1 & 0 \end{array}\right) \tag{3.29}$$

$$\Gamma_3 = \left(\begin{array}{cccc|cccc} 1 & 0 & \dots & 0 & 0 & \dots & \dots & 0 \end{array}\right)^\top \tag{3.30}$$

$$\Gamma_4 = \left(\begin{array}{cccc|cccc} 0 & \dots & \dots & 0 & 1 & 0 & \dots & 0 \end{array}\right)^\top \tag{3.31}$$

Cet algorithme s'écrit

$$\begin{pmatrix} a_0 & 0 & 0 \\ -\Gamma_4 & I_n & 0 \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} T(k+1) \\ X(k+1) \\ Y(k) \end{pmatrix} = \begin{pmatrix} 0 & \Gamma_1 & b_0 \\ 0 & \Gamma_2 & \Gamma_3 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} T(k) \\ X(k) \\ U(k) \end{pmatrix} \tag{3.32}$$

ou encore

$$Z = \left(\begin{array}{c:ccccccccc:c} -a_0 & b_1 & \cdots & \cdots & b_n & -a_1 & \cdots & \cdots & -a_n & b_0 \\ \hdashline 0 & 0 & & & & & & & & 1 \\ \vdots & 1 & \ddots & & & & & & & 0 \\ \vdots & & \ddots & \ddots & & & & & & \vdots \\ 0 & & & 1 & 0 & & & & & \vdots \\ 1 & & & & 0 & 0 & & & & \vdots \\ 0 & & & & & 1 & \ddots & & & \vdots \\ \vdots & & & & & & \ddots & \ddots & & \vdots \\ 0 & & & & & & & 1 & 0 & 0 \\ \hdashline 1 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 0 \end{array}\right) \tag{3.33}$$

Remarque : on a estimé ici que le calcul de $Y(k)$ selon l'équation (3.26) se faisait avec une division par a_0 ($J = a_0$). On peut aussi considérer cette équation avec une multiplication par $\alpha = \frac{1}{a_0}$ au lieu de la division. L'algorithme s'écrit alors

1: $T_1 \leftarrow \sum_{i=1}^{n} b_i X_i(k) + \sum_{i=1}^{n} -a_i X_{n+i}(k) + b_0 U(k)$
2: $T_2 \leftarrow \alpha T_1$

d'où

$$Z = \left(\begin{array}{cc:ccccccc:c} -1 & 0 & b_1 & \cdots & \cdots & b_n & -a_1 & \cdots & -a_n & b_0 \\ -\alpha & -1 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 0 \\ \hdashline 0 & 0 & 0 & & & & & & & 1 \\ \vdots & \vdots & 1 & \ddots & & & & & & 0 \\ \vdots & \vdots & & \ddots & \ddots & & & & & \vdots \\ \vdots & 0 & & & 1 & 0 & & & & \vdots \\ \vdots & 1 & & & & 0 & 0 & & & \vdots \\ \vdots & 0 & & & & & 1 & \ddots & & \vdots \\ \vdots & \vdots & & & & & & \ddots & \ddots & \vdots \\ 0 & 0 & & & & & & & 1 & 0 \\ \hdashline 0 & 1 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 0 \end{array}\right) \tag{3.34}$$

(une variable supplémentaire est rajoutée).

3.2.3 Réalisation en δ

3.2.3.1 Forme générale

La structuration en δ (cf. section 2.1.6), correspondant à la réalisation

$$\left\{\begin{array}{rcl} \delta[X(k)] & = & A_\delta X(k) + B_\delta U(k) \\ Y(k) & = & C_\delta X(k) + D_\delta U(k) \end{array}\right. \tag{3.35}$$

a été donnée par les équations (3.20) et (3.21). Elle correspond à

$$\begin{pmatrix} I_n & 0 & 0 \\ -\Delta I_n & I_n & 0 \\ 0 & 0 & I_p \end{pmatrix} \begin{pmatrix} T(k+1) \\ X(k+1) \\ Y(k) \end{pmatrix} = \begin{pmatrix} 0 & A_\delta & B_\delta \\ 0 & I_n & 0 \\ 0 & C_\delta & D_\delta \end{pmatrix} \begin{pmatrix} T(k) \\ X(k) \\ U(k) \end{pmatrix} \tag{3.36}$$

Le vecteur de variables intermédiaires et le vecteur d'état ont la même taille.

3.2.3.2 Forme directe I

On peut aussi, par exemple, exprimer une forme directe I avec l'opérateur δ (la forme directe II ne représente aucune difficulté car il s'agit de la forme classique (3.36) où A_δ est mise sous forme compagne) : cette forme est représentée par la figure 3.1 (voir paragraphe 2.1.6).

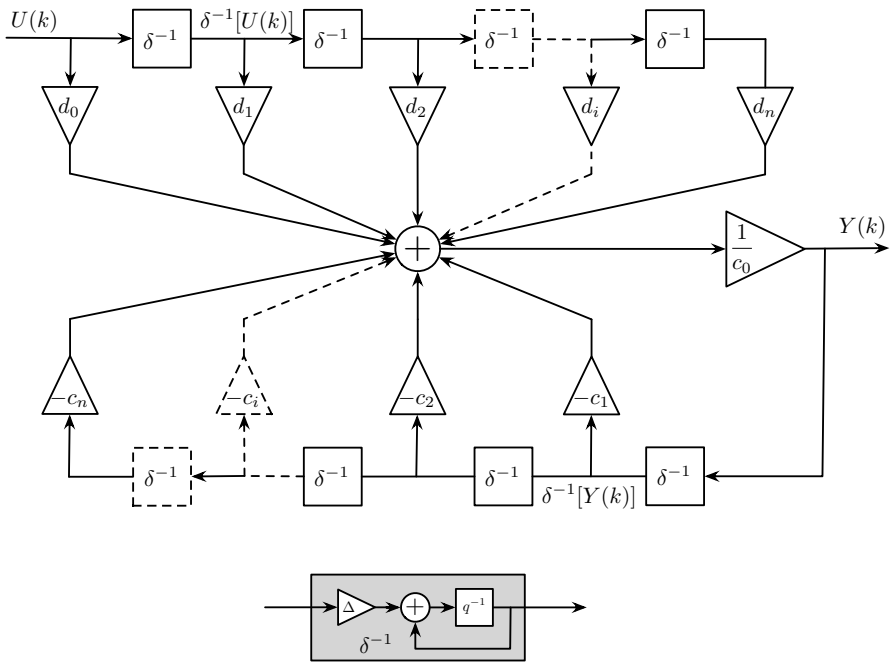


FIG. 3.1 – La forme directe I avec l'opérateur δ

Si on considère la fonction de transfert écrite avec la variable complexe ρ (associée à l'opérateur δ, cf. paragraphe 2.1.6)

$$H(\rho) = \frac{\sum_{i=0}^{n} d_i \rho^{-i}}{\sum_{i=0}^{n} c_i \rho^{-i}} \tag{3.37}$$

la forme directe I en δ correspond à

$$Y(k) = \frac{1}{c_0}\left(\sum_{i=0}^{n} d_i \delta^{-i}[U(k)] - \sum_{i=1}^{n} c_i \delta^{-i}[Y(k)]\right) \quad \forall k > n \tag{3.38}$$

(la forme directe I en q s'écrit avec la même équation (3.38), mais avec q^{-i} au lieu de δ^{-i}. On retrouve donc l'équation (2.12) avec $q^{-i}[U(k)] = U(k-i)$). On choisit l'espace d'état suivant (par analogie avec (3.27))

$$X(k) = \begin{pmatrix} \delta^{-1}[U(k)] \\ \vdots \\ \delta^{-n}[U(k)] \\ \hline \delta^{-1}[Y(k)] \\ \vdots \\ \delta^{-n}[Y(k)] \end{pmatrix} \tag{3.39}$$

La transition sur $X(k)$ s'écrit alors

$$\delta[X(k)] = \Gamma_2 X(k) + \Gamma_3 U(k) + \Gamma_4 Y(k) \tag{3.40}$$

avec Γ_2, Γ_3 et Γ_4 définis aux équations (3.29), (3.30) et (3.31), ou encore

$$X(k+1) = X(k) + \Delta\Gamma_2 X(k) + \Delta\Gamma_3 U(k) + \Delta\Gamma_4 Y(k) \tag{3.41}$$

c'est à dire

1: $X(k+1) \leftarrow \Gamma_{2\delta} X(k) + \Gamma_{3\delta} U(k) + \Gamma_{4\delta} Y(k)$

avec

$$\Gamma_{2\delta} = \left(\begin{array}{ccccc|ccccc} 1 & & & & & & & & & \\ \Delta & \ddots & & & & & & & & \\ & \ddots & \ddots & & & & & & & \\ & & \Delta & 1 & & & & & & \\ \hline & & & & & 1 & & & & \\ & & & & & \Delta & \ddots & & & \\ & & & & & & \ddots & \ddots & & \\ & & & & & & & \Delta & 1 \end{array}\right) \tag{3.42}$$

$$\Gamma_{3\delta} = \left(\begin{array}{cccc|cccc} \Delta & 0 & \dots & 0 & 0 & \dots & \dots & 0 \end{array}\right)^\top \tag{3.43}$$

$$\Gamma_{4\delta} = \left(\begin{array}{cccc|cccc} 0 & \dots & \dots & 0 & \Delta & 0 & \dots & 0 \end{array}\right)^\top \tag{3.44}$$

De la même manière que pour la forme directe I en q, le calcul de $Y(k)$ (équation (3.38)) s'effectue d'abord dans une variable intermédiaire (car on en a besoin aussi pour le calcul de $X(k+1)$) :

1: $T_1 \leftarrow \frac{1}{c_0}\left(\sum_{i=1}^{n} d_i X_i(k) + \sum_{i=1}^{n} -c_i X_{n+i}(k) + d_0 U(k)\right)$

ou encore

1: $T_1 \leftarrow \frac{1}{c_0}\left(\Gamma_{1\delta} X(k) + d_0 U(k)\right)$

avec

$$\Gamma_1 = \left(\begin{array}{cccc|cccc} d_1 & \cdots & \cdots & d_n & -c_1 & \cdots & \cdots & -c_n \end{array}\right) \tag{3.45}$$

Tout cela s'écrit

$$Z = \left(\begin{array}{c:c:c} -c_0 & \Gamma_{1\delta} & d_0 \\ \hdashline \Gamma_{4\delta} & \Gamma_{2\delta} & \Gamma_{3\delta} \\ \hdashline 1 & 0 & 0 \end{array}\right) \tag{3.46}$$

3.2.4 Découpage en cascade

On a vu, au paragraphe (2.1.8.1), qu'un système de contrôle/commande peut être décomposé en sous-systèmes que l'on cascade.
On considère deux systèmes $\mathcal{S}_1$ et $\mathcal{S}_2$ dont on connaît la réalisation sous forme implicite spécialisée

$$\begin{pmatrix} J_1 & 0 & 0 \\ -K_1 & I & 0 \\ -L_1 & 0 & I \end{pmatrix} \begin{pmatrix} T_1(k+1) \\ X_1(k+1) \\ Y_1(k) \end{pmatrix} = \begin{pmatrix} 0 & M_1 & N_1 \\ 0 & P_1 & Q_1 \\ 0 & R_1 & S_1 \end{pmatrix} \begin{pmatrix} T_1(k) \\ X_1(k) \\ U_1(k) \end{pmatrix} \tag{3.47}$$

$$\begin{pmatrix} J_2 & 0 & 0 \\ -K_2 & I & 0 \\ -L_2 & 0 & I \end{pmatrix} \begin{pmatrix} T_2(k+1) \\ X_2(k+1) \\ Y_2(k) \end{pmatrix} = \begin{pmatrix} 0 & M_2 & N_2 \\ 0 & P_2 & Q_2 \\ 0 & R_2 & S_2 \end{pmatrix} \begin{pmatrix} T_2(k) \\ X_2(k) \\ U_2(k) \end{pmatrix} \tag{3.48}$$

On cascade les deux systèmes (on supposera que la taille du vecteur de sortie du premier correspond à la taille du vecteur d'entrée du second), comme sur la figure 3.2.
En introduisant la variable intermédiaire T, égale à la sortie du premier

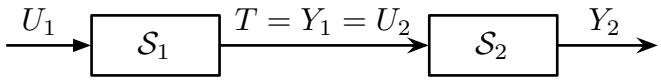


FIG. 3.2 – Mise en cascade de deux sous-systèmes

système et avec les calculs du second système utilisant ce vecteur comme entrée, le système résultant a pour réalisation :

$$\left(\begin{array}{ccc:cc:c} J_1 & 0 & 0 & 0 & 0 & 0 \\ -L_1 & I & 0 & 0 & 0 & 0 \\ 0 & -N_2 & J_2 & 0 & 0 & 0 \\ \hdashline -K_1 & 0 & 0 & I & 0 & 0 \\ 0 & -Q_2 & -K_2 & 0 & I & 0 \\ \hdashline 0 & -S_2 & -L_2 & 0 & 0 & I \end{array}\right) \begin{pmatrix} T_1(k+1) \\ T(k+1) \\ T_2(k+1) \\ X_1(k+1) \\ X_2(k+1) \\ Y_2(k) \end{pmatrix} = \left(\begin{array}{ccc:cc:c} 0 & 0 & 0 & M_1 & 0 & N_1 \\ 0 & 0 & 0 & R_1 & 0 & S_1 \\ 0 & 0 & 0 & 0 & M_2 & 0 \\ \hdashline 0 & 0 & 0 & P_1 & 0 & Q_1 \\ 0 & 0 & 0 & 0 & P_2 & 0 \\ \hdashline 0 & 0 & 0 & 0 & R_2 & 0 \end{array}\right) \begin{pmatrix} T_1(k) \\ T(k) \\ T_2(k) \\ X_1(k) \\ X_2(k) \\ U_1(k) \end{pmatrix}$$

ou encore

$$Z = \left(\begin{array}{ccc:cc:c} -J_1 & 0 & 0 & M_1 & 0 & N_1 \\ L_1 & -I & 0 & R_1 & 0 & S_1 \\ 0 & N_2 & -J_2 & 0 & M_2 & 0 \\ \hdashline K_1 & 0 & 0 & P_1 & 0 & Q_1 \\ 0 & Q_2 & K_2 & 0 & P_2 & 0 \\ \hdashline 0 & S_2 & L_2 & 0 & R_2 & 0 \end{array}\right) \tag{3.49}$$

Il est important de noter que cette écriture permet de conserver intacts la paramétrisation, les calculs et l'ordre des calculs. Cela est nécessaire pour étudier l'impact possible de la quantification et la propagation des bruits de calculs.

Dans le cas particulier où les deux systèmes à cascader sont sous forme d'état classique ($\mathcal{S}_1$ décrit par (A_1, B_1, C_1, D_1) et $\mathcal{S}_2$ par (A_2, B_2, C_2, D_2)), on pourra écrire

$$\begin{pmatrix} I & 0 & 0 \\ \begin{pmatrix} 0 \\ -B_2 \end{pmatrix} & I & 0 \\ -D_2 & 0 & I \end{pmatrix} \begin{pmatrix} T(k+1) \\ \begin{pmatrix} X_1(k+1) \\ X_2(k+1) \end{pmatrix} \\ Y_2(k) \end{pmatrix} = \begin{pmatrix} 0 & \begin{pmatrix} C_1 & 0 \end{pmatrix} & D_1 \\ 0 & \begin{pmatrix} A_1 & 0 \\ 0 & A_2 \end{pmatrix} & \begin{pmatrix} B_1 \\ 0 \end{pmatrix} \\ 0 & \begin{pmatrix} 0 & C_2 \end{pmatrix} & 0 \end{pmatrix} \begin{pmatrix} T(k) \\ \begin{pmatrix} X_1(k) \\ X_2(k) \end{pmatrix} \\ U_1(k) \end{pmatrix} \tag{3.50}$$

Notons que la forme implicite est ici une nouvelle fois nécessaire pour rendre compte de la paramétrisation. En effet, si la mise en cascade de $\mathcal{S}_1$ et $\mathcal{S}_2$ admet la réalisation classique (A, B, C, D) donnée par

$$\begin{array}{cc} A = \begin{pmatrix} A_1 & 0 \\ B_2C_1 & A_2 \end{pmatrix} & B = \begin{pmatrix} B_1 \\ B_2D_1 \end{pmatrix} \\ C = \begin{pmatrix} D_2C_1 & C_2 \end{pmatrix} & D = D_2D_1 \end{array} \tag{3.51}$$

cette dernière ne correspond plus à la paramétrisation souhaitée.
Mais, en pratique, le calcul n'est pas réalisé avec les matrices A, B, C et D mais (et c'est important pour la quantification) avec les matrices A_1, B_1, C_1, D_1, A_2, B_2, C_2 et D_2 selon l'algorithme décrit par l'équation (3.50).

3.2.5 Retour d'état/observateur

On considère ici une forme retour d'état-observateur s'écrivant

$$\left\{ \begin{array}{rcl} \hat{X}(k+1) & = & A_p\hat{X}(k) + B_pU(k) + K_o\left(Y(k) - C_p\hat{X}(k)\right) \\ U(k) & = & -K_c\hat{X}(k) + Q\left(Y(k) - C_p\hat{X}(k)\right) \end{array} \right.$$

avec $A_p \in \mathbb{R}^{n\times n}$, $B_p \in \mathbb{R}^{n\times m}$ et $C_p \in \mathbb{R}^{p\times n}$, $Q \in \mathbb{R}^{m\times p}$ (paramètre de Youla), $K_o \in \mathbb{R}^{n\times p}$ et $K_c \in \mathbb{R}^{m\times n}$ (par rapport à la forme exprimée au paragraphe 2.1.3, U_{ref} et $\hat{X}_{ref}$ ont été supposés nuls).

Suivant la façon de regrouper les différents termes, et d'organiser les calculs, il existe différentes implémentations possibles, et donc des paramétrisations différentes :

– Une première façon d'organiser les calculs est d'agréger le maximum de termes

$$\left\{ \begin{array}{rcl} \hat{X}(k+1) & = & \left(A_p - K_fC_p\right)\hat{X}(k) + B_pU(k) + K_fY(k) \\ U(k) & = & -\left(QC_p + K_c\right)\hat{X}(k) + QY(k) \end{array} \right.$$

Il s'agit donc de calculer d'abord $U(k)$ puis ensuite $\hat{X}(k+1)$, selon l'algorithme

1: $T \leftarrow -\left(QC_p + K_c\right)\hat{X}(k) + QY(k)$
2: $\hat{X}(k+1) \leftarrow \left(A_p - K_f C_p\right)\hat{X}(k) + B_p U(k) + K_f Y(k)$
3: $U(k) \leftarrow T$

ce qui est décrit par la forme implicite

$$Z = \left(\begin{array}{c|c|c} -I_p & -\left(QC_p + K_c\right) & Q \\ \hline B_p & A_p - K_f C_p & K_f \\ \hline I_p & 0 & 0 \end{array}\right) \tag{3.52}$$

– Une autre solution, faisant apparaître explicitement cette fois tous les paramètres K_c, K_f, Q, A_p, B_p et C_p, consiste à calculer d'abord le terme $Y(k) - C_p\hat{X}(k)$ qui apparaît dans le calcul de $\hat{X}(k+1)$ et de $U(k)$

1: $T_1 \leftarrow Y(k) - C_p\hat{X}(k)$
2: $T_2 \leftarrow QT_1 - K_c\hat{X}(k)$
3: $\hat{X}(k+1) \leftarrow A_p\hat{X}(k) + B_pT_2 + K_fT_1$
4: $U(k) \leftarrow T_2$

et

$$Z = \left(\begin{array}{cc|c|c} -I_p & 0 & -C_p & I_p \\ Q & -I_p & -K_c & 0 \\ \hline K_f & B_p & A_p & 0 \\ \hline 0 & I_m & 0 & 0 \end{array}\right) \tag{3.53}$$

Le choix d'une solution impacte la paramétrisation et par suite le volume de calcul, la lisibilité de l'implémentation et son comportement après implémentation.

3.3 Classes d'équivalence

Dans l'objectif de recherche d'une réalisation performante en précision finie, il peut être intéressant de caractériser mathématiquement l'ensemble des réalisations équivalentes à une réalisation donnée, ainsi que des sous-ensembles de réalisations de dimensions fixées (non nécessairement minimales) ou de structuration fixée.
Il nous faut alors décrire les transformations nécessaires pour passer d'une réalisation à l'autre, tout en préservant l'équivalence (mathématique). Cette caractérisation s'appuiera sur le *Principe d'Inclusion*, que nous adapterons à la forme implicite spécialisée.

3.3.1 Principe d'Inclusion

Le *Principe d'Inclusion*, introduit par Šiljak et Ikeda ([49, 50, 102, 8, 9, 99]) dans le cadre de la commande décentralisée avec recouvrement, permet d'énoncer mathématiquement les relations d'*équivalence*, d'*inclusion*, de *restriction*, d'*agrégation* entre deux systèmes linéaires. Cela peut concerner deux systèmes aux comportements équivalents, mais de dimensions différentes : par exemple, le "grand" système peut être construit à partir du "petit" via un processus d'*expansion*, dans le sens où le "grand" système contient l'information relative au comportement du "petit", information pouvant être extraite du "grand" par un processus de *contraction*.
Le principe d'inclusion est principalement énoncé en temps continu dans la littérature, mais peut tout autant s'appliquer en temps discret, quasiment sans changements (les démonstrations, que l'on trouve dans [49, 102] seront réécrites en discret si nécessaire).
On considère deux systèmes linéaires $\mathcal{S}$ et $\tilde{\mathcal{S}}$ de dimension n et $\tilde{n}$, avec les mêmes m entrées et p sorties (le principe d'inclusion s'écrit aussi, de manière plus compliquée, avec une transformation sur des entrées et sorties, qui ne nous intéresse pas ici) :

$$\mathcal{S}\begin{cases} X(k+1) &= AX(k)+BU(k) \\ Y(k) &= CX(k) \end{cases} \tag{3.54}$$

$$\tilde{\mathcal{S}}\begin{cases} \tilde{X}(k+1) &= \tilde{A}\tilde{X}(k)+\tilde{B}U(k) \\ \tilde{Y}(k) &= \tilde{C}\tilde{X}(k) \end{cases} \tag{3.55}$$

avec $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$, $\tilde{A} \in \mathbb{R}^{\tilde{n}\times\tilde{n}}$, $\tilde{B} \in \mathbb{R}^{\tilde{n}\times m}$ et $\tilde{C} \in \mathbb{R}^{p\times\tilde{n}}$.
On supposera par la suite que la dimension de $\mathcal{S}$ est inférieure (ou égale) à la dimension de $\tilde{\mathcal{S}}$: $n \leqslant \tilde{n}$.

Définition 3.7 (Inclusion d'un système)

Le système $\tilde{\mathcal{S}}$ inclut le système $\mathcal{S}$ (ou, de manière équivalente, le système $\mathcal{S}$ est inclus dans le système $\tilde{\mathcal{S}}$) s'il existe une paire de matrices $(\mathcal{U},\mathcal{V}) \in \mathbb{R}^{n\times\tilde{n}} \times \mathbb{R}^{\tilde{n}\times n}$ telles que $\mathcal{U}\mathcal{V} = I_n$ et, pour tout état initial $X(0) = X_0$ de $\mathcal{S}$ et toute séquence d'entrée $(U(k)_{k\geqslant 0})$, le choix de l'état initial $\tilde{X}(0) = \mathcal{V}X_0$ de $\tilde{\mathcal{S}}$ implique

$$\begin{cases} X(k) &= \mathcal{U}\tilde{X}(k) \\ Y(k) &= \tilde{Y}(k) \end{cases} \qquad \forall k \geqslant 0 \tag{3.56}$$

On notera alors $\tilde{\mathcal{S}} \supset \mathcal{S}$.

Remarque : la condition (3.56) implique que le système $\tilde{\mathcal{S}}$ contient toute l'information nécessaire pour connaître le comportement et la sortie de $\mathcal{S}$ (qu'il est donc possible de déduire toute trajectoire de $\mathcal{S}$ par simulation de $\tilde{\mathcal{S}}$).

Si $\tilde{\mathcal{S}}$ inclut le système $\mathcal{S}$, alors $\tilde{\mathcal{S}}$ est une *expansion* de $\mathcal{S}$, et $\mathcal{S}$ est une *contraction* de $\tilde{\mathcal{S}}$.

Théorème 3.2 (1re caractérisation du Principe d'Inclusion)

Le système $\mathcal{S}$, défini par (3.54), est inclus dans le système $\tilde{\mathcal{S}}$, défini par (3.55) si et seulement si

$$\begin{cases} \mathcal{U}\tilde{A}^i\mathcal{V} &= A^i \\ \mathcal{U}\tilde{A}^i\tilde{B} &= A^iB \\ \tilde{C}\tilde{A}^i\mathcal{V} &= CA^i \\ \tilde{C}\tilde{A}^i\tilde{B} &= CA^iB \end{cases} \qquad \forall i \geqslant 0 \tag{3.57}$$

Démonstration :
La démonstration en discret s'inspire de la démonstration en continu que l'on trouve dans [49].
Pour tout état initial X_0 de $\mathcal{S}$, tout état initial $\mathcal{V}X_0$ de $\tilde{\mathcal{S}}$ et toute séquence d'entrée $(U(k)_{k\geqslant 0})$, on a :

$$\begin{aligned} X(k) &= A^kX_0 + \sum_{i=0}^{k-1} A^{k-i}BU(i) \\ \tilde{X}(k) &= \tilde{A}^k\mathcal{V}X_0 + \sum_{i=0}^{k-1} \tilde{A}^{k-i}\tilde{B}U(i) \\ Y(k) &= CA^kX_0 + \sum_{i=0}^{k-1} CA^{k-i}BU(i) \\ \tilde{Y}(k) &= \tilde{C}\tilde{A}^k\mathcal{V}X_0 + \sum_{i=0}^{k-1} \tilde{C}\tilde{A}^{k-i}\tilde{B}U(i) \end{aligned}$$

Tirant parti de ces relations, on montre sans difficulté que (3.56) implique (3.57) et réciproquement, et donc que la CNS est vérifiée. ■

De plus, une seconde caractérisation de l'inclusion est donnée dans le théorème suivant ([49, 50]) :

Théorème 3.3 (2de caractérisation du Principe d'Inclusion)

On choisit une paire de matrices $(\mathcal{U},\mathcal{V}) \in \mathbb{R}^{n\times\tilde{n}} \times \mathbb{R}^{\tilde{n}\times n}$ telles que $\mathcal{U}\mathcal{V} = I_n$.
Le système $\tilde{\mathcal{S}}$, défini par (3.55), est inclus dans le système $\mathcal{S}$, défini par

(3.54) si et seulement si

$$\begin{array}{rl} \mathcal{U}\left(\mathcal{M}_{\tilde{A}}\right)^i\mathcal{V}=0 & \forall i\geqslant 1\\ \mathcal{U}\left(\mathcal{M}_{\tilde{A}}\right)^i\mathcal{M}_{\tilde{B}}=0 & \forall i\geqslant 0\\ \mathcal{M}_{\tilde{C}}\left(\mathcal{M}_{\tilde{A}}\right)^i\mathcal{V}=0 & \forall i\geqslant 0\\ \mathcal{M}_{\tilde{C}}\left(\mathcal{M}_{\tilde{A}}\right)^i\mathcal{M}_{\tilde{B}}=0 & \forall i\geqslant 0 \end{array} \tag{3.58}$$

où $\mathcal{M}_{\tilde{A}}\in\mathbb{R}^{\tilde{n}\times\tilde{n}}$, $\mathcal{M}_{\tilde{B}}\in\mathbb{R}^{\tilde{n}\times m}$ et $\mathcal{M}_{\tilde{C}}\in\mathbb{R}^{p\times\tilde{n}}$ sont les matrices complémentaires de $\tilde{A}$, $\tilde{B}$ et $\tilde{C}$ telles que

$$\begin{array}{rcl} \tilde{A} & = & \mathcal{V}A\mathcal{U}+\mathcal{M}_{\tilde{A}}\\ \tilde{B} & = & \mathcal{V}B+\mathcal{M}_{\tilde{B}}\\ \tilde{C} & = & C\mathcal{U}+\mathcal{M}_{\tilde{C}} \end{array} \tag{3.59}$$

(elles sont définies à partir de A, B, C, $\tilde{A}$, $\tilde{B}$, $\tilde{C}$, $\mathcal{U}$ et $\mathcal{V}$)

(Les matrices $\mathcal{M}_{\tilde{A}}$, $\mathcal{M}_{\tilde{B}}$ et $\mathcal{M}_{\tilde{C}}$ sont souvent notées M, N et L dans la littérature, mais nous avons changé la notation pour pouvoir étendre le principe d'inclusion à la forme implicite spécialisée, où les matrices M, N et L sont déjà utilisées et où de nouvelles matrices complémentaires, notées $\mathcal{M}_{\tilde{X}}$ avec $\tilde{X}$ de $\tilde{J}$ à $\tilde{S}$, seront introduites).

Cette dernière proposition nous permet de définir la classe d'équivalence des réalisations $\tilde{\mathcal{S}}$ incluant $\mathcal{S}$:

Proposition 3.4
Soit une réalisation $\mathcal{S}$ définie par (3.54). L'ensemble des réalisations $\tilde{\mathcal{S}}$ de dimension supérieure ou égale à celle de $\mathcal{S}$ et incluant $\mathcal{S}$ est défini par

$$\Omega_{\supset\mathcal{S}}=\left\{\begin{array}{lll} \tilde{\mathcal{S}}= & (\tilde{A},\tilde{B},\tilde{C}) & \left\backslash\begin{array}{l}\tilde{A}=\mathcal{V}A\mathcal{U}+\mathcal{M}_{\tilde{A}}\\ \tilde{B}=\mathcal{V}B+\mathcal{M}_{\tilde{B}}\\ \tilde{C}=C\mathcal{U}+\mathcal{M}_{\tilde{C}}\end{array}\right.\\ & \forall\tilde{n}\geqslant n & \\ & \forall\left(\mathcal{U},\mathcal{V}\right)\in\mathbb{R}^{n\times\tilde{n}}\times\mathbb{R}^{\tilde{n}\times n}\ \textit{tel que}\ \mathcal{U}\mathcal{V}=I_n & \\ & \forall\mathcal{M}_{\tilde{A}}\in\mathbb{R}^{\tilde{n}\times\tilde{n}},\mathcal{M}_{\tilde{B}}\in\mathbb{R}^{\tilde{n}\times m},\mathcal{M}_{\tilde{C}}\in\mathbb{R}^{\tilde{n}\times p} & \\ & \textit{tels que les équations (3.58) soient vérifiées} & \end{array}\right\} \tag{3.60}$$

Démonstration :
Cette proposition découle trivialement du théorème 3.3 ■

Il est enfin possible de relier la relation d'inclusion entre deux systèmes à leur équivalence :

Proposition 3.5
Si $\tilde{\mathcal{S}}\supset\mathcal{S}$, alors les deux systèmes $\mathcal{S}$ et $\tilde{\mathcal{S}}$ ont même fonction de transfert et sont équivalents.

Démonstration :
voir [50] ■

La figure 3.3 montre les relations entre trois réalisations équivalentes $(\mathcal{S}_0)$, $(\mathcal{S}_1)$ et $(\mathcal{S}_2)$, avec $(\mathcal{S}_0)$ minimale, et $(\mathcal{S}_1)$ et $(\mathcal{S}_2)$ de dimensions supérieures. Ainsi, parant d'une réalisation $\mathcal{S}_0$ minimale, on peut parcourir l'ensemble

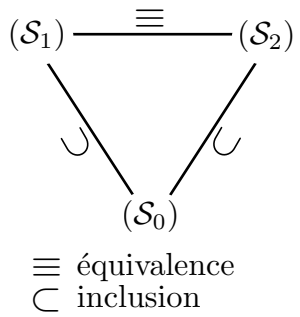


FIG. 3.3 – Relations entre réalisations équivalentes ($\mathcal{S}_0$ est supposée minimale)

des réalisations équivalentes de dimension $\tilde{n} \geqslant n$ arbitraire, en parcourant l'ensemble des matrices $(\mathcal{U}, \mathcal{V})$ telles que $\mathcal{U}\mathcal{V} = I_n$ et des matrices $\mathcal{M}_{\tilde{A}}$, $\mathcal{M}_{\tilde{B}}$ et $\mathcal{M}_{\tilde{C}}$ vérifiant (3.58).

3.3.2 Extension du Principe

Il peut être intéressant d'étendre la notion de principe d'inclusion à notre forme implicite spécialisée afin de caractériser mathématiquement l'ensemble des réalisations, sous forme implicite, qui sont équivalentes à une réalisation initiale.

Le résultat ci-dessous généralise le principe de l'inclusion au cas de la forme implicite spécialisée :

Théorème 3.6 (Principe d'inclusion pour la forme implicite)

Soit $\mathcal{R} := (J, K, L, M, N, P, Q, R, S)$ *une réalisation de dimension* l, m, n, p.
On choisit $\tilde{n} \geqslant n$ *et* $\tilde{l} \geqslant l$
On choisit $(\mathcal{U}, \mathcal{V}) \in \mathbb{R}^{n\times\tilde{n}} \times \mathbb{R}^{\tilde{n}\times n}$ *telles que* $\mathcal{U}\mathcal{V} = I_n$.
On choisit $(\mathcal{W}, \mathcal{T}) \in \mathbb{R}^{l\times\tilde{l}} \times \mathbb{R}^{\tilde{l}\times l}$ *telles que* $\mathcal{W}\mathcal{T} = I_l$.
On choisit $(\mathcal{X}, \mathcal{Y}) \in \mathbb{R}^{l\times\tilde{l}} \times \mathbb{R}^{\tilde{l}\times l}$ *telles que* $\mathcal{X}\mathcal{Y} = I_l$.
On choisit $\mathcal{M}_{\tilde{J}^{-1}} \in \mathbb{R}^{\tilde{l}\times\tilde{l}}$, $\mathcal{M}_{\tilde{K}} \in \mathbb{R}^{\tilde{n}\times\tilde{l}}$, $\mathcal{M}_{\tilde{L}} \in \mathbb{R}^{p\times\tilde{l}}$, $\mathcal{M}_{\tilde{M}} \in \mathbb{R}^{\tilde{l}\times\tilde{n}}$,

$\mathcal{M}_{\tilde{N}} \in \mathbb{R}^{\tilde{l}\times m}$, $\mathcal{M}_{\tilde{P}} \in \mathbb{R}^{\tilde{n}\times\tilde{n}}$, $\mathcal{M}_{\tilde{Q}} \in \mathbb{R}^{\tilde{n}\times m}$, $\mathcal{M}_{\tilde{R}} \in \mathbb{R}^{p\times\tilde{n}}$ et $\mathcal{M}_{\tilde{S}} \in \mathbb{R}^{p\times m}$ telles que les matrices définies par

$$\begin{aligned}
\mathcal{M}_{\tilde{A}} &= \left(\mathcal{V}K\mathcal{W} + \mathcal{M}_{\tilde{K}}\right)\left(\mathcal{T}J^{-1}\mathcal{X} + \mathcal{M}_{\tilde{J}^{-1}}\right)\left(\mathcal{Y}M\mathcal{U} + \mathcal{M}_{\tilde{M}}\right) + \mathcal{M}_{\tilde{P}} - KJ^{-1}M \\
\mathcal{M}_{\tilde{B}} &= \left(\mathcal{V}K\mathcal{W} + \mathcal{M}_{\tilde{K}}\right)\left(\mathcal{T}J^{-1}\mathcal{X} + \mathcal{M}_{\tilde{J}^{-1}}\right)\left(\mathcal{Y}N + \mathcal{M}_{\tilde{N}}\right) + \mathcal{M}_{\tilde{Q}} - KJ^{-1}N \\
\mathcal{M}_{\tilde{C}} &= \left(L\mathcal{W} + \mathcal{M}_{\tilde{L}}\right)\left(\mathcal{T}J^{-1}\mathcal{X} + \mathcal{M}_{\tilde{J}^{-1}}\right)\left(\mathcal{Y}M\mathcal{U} + \mathcal{M}_{\tilde{M}}\right) + \mathcal{M}_{\tilde{R}} - LJ^{-1}M \\
\mathcal{M}_{\tilde{D}} &= \left(L\mathcal{W} + \mathcal{M}_{\tilde{L}}\right)\left(\mathcal{T}J^{-1}\mathcal{X} + \mathcal{M}_{\tilde{J}^{-1}}\right)\left(\mathcal{Y}N + \mathcal{M}_{\tilde{N}}\right) + \mathcal{M}_{\tilde{S}} - LJ^{-1}N
\end{aligned}$$

vérifient

$$\begin{aligned}
\mathcal{U}\left(\mathcal{M}_{\tilde{A}}\right)^i\mathcal{V} &= 0 \qquad \forall i \geqslant 1 \\
\mathcal{U}\left(\mathcal{M}_{\tilde{A}}\right)^i\mathcal{M}_{\tilde{B}} &= 0 \qquad \forall i \geqslant 0 \\
\mathcal{M}_{\tilde{C}}\left(\mathcal{M}_{\tilde{A}}\right)^i\mathcal{V} &= 0 \qquad \forall i \geqslant 0 \\
\mathcal{M}_{\tilde{C}}\left(\mathcal{M}_{\tilde{A}}\right)^i\mathcal{M}_{\tilde{B}} &= 0 \qquad \forall i \geqslant 0 \\
\mathcal{M}_{\tilde{D}} &= 0
\end{aligned} \tag{3.61}$$

Alors, la réalisation $\tilde{\mathcal{R}} := (\tilde{J}, \tilde{K}, \tilde{L}, \tilde{M}, \tilde{N}, \tilde{P}, \tilde{Q}, \tilde{R}, \tilde{S})$ définie par les relations

$$\begin{aligned}
\tilde{J}^{-1} &= \mathcal{T}J^{-1}\mathcal{X} + \mathcal{M}_{\tilde{J}^{-1}} & & \\
\tilde{K} &= \mathcal{V}K\mathcal{W} + \mathcal{M}_{\tilde{K}} & \tilde{L} &= L\mathcal{W} + \mathcal{M}_{\tilde{L}} \\
\tilde{M} &= \mathcal{Y}M\mathcal{U} + \mathcal{M}_{\tilde{M}} & \tilde{N} &= \mathcal{Y}N + \mathcal{M}_{\tilde{N}} \\
\tilde{P} &= \mathcal{V}P\mathcal{U} + \mathcal{M}_{\tilde{P}} & \tilde{Q} &= \mathcal{V}Q + \mathcal{M}_{\tilde{Q}} \\
\tilde{R} &= R\mathcal{U} + \mathcal{M}_{\tilde{R}} & \tilde{S} &= S + \mathcal{M}_{\tilde{S}}
\end{aligned} \tag{3.62}$$

inclut la réalisation $\mathcal{R}$: $\tilde{\mathcal{R}} \supset \mathcal{R}$

Démonstration :
On vérifie aisément que les matrices $\mathcal{M}_{\tilde{A}}$, $\mathcal{M}_{\tilde{B}}$, $\mathcal{M}_{\tilde{C}}$ et $\mathcal{M}_{\tilde{D}}$ définies ci-dessus satisfont

$$\tilde{A} = \mathcal{V}A\mathcal{U} + \mathcal{M}_{\tilde{A}} \tag{3.63}$$
$$\tilde{B} = \mathcal{V}B + \mathcal{M}_{\tilde{B}} \tag{3.64}$$
$$\tilde{C} = C\mathcal{U} + \mathcal{M}_{\tilde{C}} \tag{3.65}$$
$$\tilde{D} = S + \mathcal{M}_{\tilde{D}} \tag{3.66}$$

où A, B, C, D et $\tilde{A}$, $\tilde{B}$, $\tilde{C}$, $\tilde{D}$ sont les matrices associées respectivement aux réalisations $\mathcal{R}$ et $\tilde{\mathcal{R}}$, selon les équations (3.8) à (3.9).

Dans ces conditions, on peut appliquer la proposition 3.4. ■

De plus, la réciproque est vraie : si une réalisation $\tilde{\mathcal{R}}$ vérifie $\tilde{\mathcal{R}} \supset \mathcal{R}$, alors il existe $(\mathcal{U}, \mathcal{V})$, $(\mathcal{W}, \mathcal{T})$, $(\mathcal{X}, \mathcal{Y})$ tels que $\mathcal{U}\mathcal{V} = I_n$, $\mathcal{W}\mathcal{T} = I_l$, $\mathcal{X}\mathcal{Y} = I_l$ et il existe $\mathcal{M}_{\tilde{J}^{-1}}$, $\mathcal{M}_{\tilde{K}}$, $\mathcal{M}_{\tilde{L}}$, $\mathcal{M}_{\tilde{M}}$, $\mathcal{M}_{\tilde{N}}$, $\mathcal{M}_{\tilde{P}}$, $\mathcal{M}_{\tilde{Q}}$, $\mathcal{M}_{\tilde{R}}$, $\mathcal{M}_{\tilde{S}}$ telles que les relations (3.61) sont vérifiées.

3.3.3 Sous-classes d'équivalence

La caractérisation de l'ensemble des réalisations implicites spécialisées équivalentes à une réalisation $\mathcal{R}$ donnée ne permet pas en pratique une recherche systématique de la *meilleure* réalisation. L'ensemble est trop vaste et les expressions de $\mathcal{M}_{\tilde{A}}$, $\mathcal{M}_{\tilde{B}}$, $\mathcal{M}_{\tilde{C}}$ et $\mathcal{M}_{\tilde{D}}$ en fonction de $\mathcal{U}$, $\mathcal{V}$, $\mathcal{W}$, $\mathcal{T}$, $\mathcal{X}$, $\mathcal{Y}$ et J, K, L, M, N, P, Q, R et S trop complexes (cf. conditions (3.61)) . On se restreindra donc, en pratique, à parcourir des sous-ensembles.

3.3.3.1 Cas particuliers

Par exemple, les conditions (3.61) peuvent être remplacées par des conditions plus contraignantes :

$$\mathcal{M}_{\tilde{A}} = 0, \quad \mathcal{M}_{\tilde{B}} = 0, \quad \mathcal{M}_{\tilde{C}} = 0 \quad \mathcal{M}_{\tilde{D}} = 0 \tag{3.67}$$

Un exemple est le passage d'une réalisation en q à une réalisation en δ. On considère donc la réalisation $\mathcal{R}$ suivante

$$\mathcal{R} := \left(\begin{array}{c:c:c} I_l & 0 & 0 \\ \hdashline 0 & A_q & B_q \\ \hdashline 0 & C_q & D_q \end{array}\right) \tag{3.68}$$

On choisit alors $\mathcal{U} = \mathcal{V} = I_n$ et $\mathcal{W}$, $\mathcal{T}$, $\mathcal{X}$, $\mathcal{Y}$ tel que $\mathcal{W}\mathcal{T} = I_n$ et $\mathcal{X} = \mathcal{W}$, $\mathcal{Y} = \mathcal{T}$
On choisit alors les matrices complémentaires suivantes

$$\begin{array}{rclrcl} \mathcal{M}_{\tilde{J}^{-1}} & = & 0 & \mathcal{M}_{\tilde{K}} & = & \Delta I_n \\ \mathcal{M}_{\tilde{L}} & = & 0 & \mathcal{M}_{\tilde{M}} & = & \frac{A_q - I_n}{\Delta} \\ \mathcal{M}_{\tilde{N}} & = & \frac{B_q}{\Delta} & \mathcal{M}_{\tilde{P}} & = & I_n - A_q \\ \mathcal{M}_{\tilde{Q}} & = & -B_q & \mathcal{M}_{\tilde{R}} & = & 0 \\ \mathcal{M}_{\tilde{S}} & = & 0 & & & \end{array} \tag{3.69}$$

On a alors

$$\begin{aligned} \mathcal{M}_{\tilde{A}} &= \mathcal{M}_{\tilde{K}}\mathcal{T}\mathcal{X}\mathcal{M}_{\tilde{M}} + \mathcal{M}_{\tilde{P}} \\ &= 0 && (3.70) \\ \mathcal{M}_{\tilde{B}} &= \mathcal{M}_{\tilde{K}}\mathcal{T}\mathcal{X}\mathcal{M}_{\tilde{N}} + \mathcal{M}_{\tilde{Q}} \\ &= 0 && (3.71) \\ \mathcal{M}_{\tilde{C}} &= \mathcal{M}_{\tilde{L}}\mathcal{T}\mathcal{X}\mathcal{M}_{\tilde{M}} + \mathcal{M}_{\tilde{R}} \\ &= 0 && (3.72) \\ \mathcal{M}_{\tilde{C}} &= \mathcal{M}_{\tilde{L}}\mathcal{T}\mathcal{X}\mathcal{M}_{\tilde{N}} + \mathcal{M}_{\tilde{S}} \\ &= 0 && (3.73) \end{aligned}$$

(ce choix de matrices complémentaires convient bien)
Alors la réalisation $\tilde{\mathcal{R}} = (\tilde{E}, \tilde{J}, \tilde{K}, \tilde{L}, \tilde{M}, \tilde{N}, \tilde{P}, \tilde{Q}, \tilde{R}, \tilde{S})$ définie par

$$\begin{array}{rclcrcl}
\tilde{J} &=& \left(\mathcal{T}J^{-1}\mathcal{X} + \mathcal{M}_{\tilde{J}^{-1}}\right)^{-1} & & \tilde{K} &=& \mathcal{V}K\mathcal{W} + \mathcal{M}_{\tilde{K}} \\
&=& I_n & & &=& \Delta I_n \\
\tilde{L} &=& L\mathcal{W} + \mathcal{M}_{\tilde{L}} & & \tilde{M} &=& \mathcal{Y}M\mathcal{U} + \mathcal{M}_{\tilde{M}} \\
&=& 0 & & &=& \frac{A_q - I_n}{\Delta} \\
\tilde{N} &=& \mathcal{Y}N + \mathcal{M}_{\tilde{N}} & & \tilde{P} &=& \mathcal{V}P\mathcal{U} + \mathcal{M}_{\tilde{P}} \\
&=& \frac{B_q}{\Delta} & & &=& I_n \\
\tilde{Q} &=& \mathcal{V}Q + \mathcal{M}_{\tilde{Q}} & & \tilde{R} &=& R\mathcal{U} + \mathcal{M}_{\tilde{R}} \\
&=& B_q - B_q = 0 & & &=& C_q \\
\tilde{S} &=& S + \mathcal{M}_{\tilde{S}} & & & & \\
&=& D_q & & & &
\end{array}$$

inclut la réalisation $\mathcal{R}$. $\tilde{\mathcal{R}}$ s'écrit aussi

$$\tilde{Z} = \begin{pmatrix} I_n & \frac{A_q - I_n}{\Delta} & \frac{B_q}{\Delta} \\ -\Delta I_n & I_n & 0 \\ 0 & C_q & D_q \end{pmatrix} \tag{3.74}$$

et n'est autre que la réalisation $\mathcal{R}$ réécrite avec une structuration en δ (cf. section 2.1.6).
Ici, $\mathcal{U} = \mathcal{V} = I_n$ impose qu'il n'y ait pas de transformation entre A et $\tilde{A}$ et $\mathcal{M}_{\tilde{A}} = 0$ amène $A = \tilde{A}$. La seule différence entre $\mathcal{R}$ et $\tilde{\mathcal{R}}$ est que, dans le cas de $\tilde{\mathcal{R}}$, la valeur de $\tilde{A} = A_q$ est distribuée entre les termes $\tilde{K}\tilde{J}^{-1}\tilde{M}$ et $\tilde{P}$ ($\tilde{K}\tilde{J}^{-1}\tilde{M} = A_q - I_n$ et $\tilde{P} = I_n$), alors que, dans le cas de $\mathcal{R}$, $KJ^{-1}M = 0$ et $P = A_q$.

3.3.3.2 Changement de base

Pour simplifier encore, on peut aussi se suffire des changements de base (du vecteur de variables intermédiaires et du vecteur d'états) qu'offre le choix des matrices $\mathcal{U}$, $\mathcal{V}$, $\mathcal{W}$, $\mathcal{W}$, $\mathcal{T}$, $\mathcal{X}$, $\mathcal{Y}$. On impose alors aux matrices complémentaires $\mathcal{M}_{\tilde{J}^{-1}}$, $\mathcal{M}_{\tilde{K}}$, $\mathcal{M}_{\tilde{L}}$, $\mathcal{M}_{\tilde{M}}$, $\mathcal{M}_{\tilde{N}}$, $\mathcal{M}_{\tilde{P}}$, $\mathcal{M}_{\tilde{Q}}$, $\mathcal{M}_{\tilde{R}}$ et $\mathcal{M}_{\tilde{S}}$ d'être nulles, ce qui implique que les conditions (3.61) sont satisfaites.
Le sous-ensemble de réalisations équivalentes se construit autour des matrices $\mathcal{U}$, $\mathcal{V}$, $\mathcal{W}$, $\mathcal{T}$, $\mathcal{X}$, $\mathcal{Y}$: il s'agit de l'ensemble des réalisations $\tilde{\mathcal{R}} = (\tilde{J}, \tilde{K}, \tilde{L}, \tilde{M}, \tilde{N}, \tilde{P}, \tilde{Q}, \tilde{R}, \tilde{S})$ telles que

$$\begin{array}{rclcrcl}
\tilde{J}^{-1} &=& \mathcal{T}J^{-1}\mathcal{X} & & & & \\
\tilde{K} &=& \mathcal{V}K\mathcal{W} & & \tilde{L} &=& L\mathcal{W} \\
\tilde{M} &=& \mathcal{Y}M\mathcal{U} & & \tilde{N} &=& \mathcal{Y}N \\
\tilde{P} &=& \mathcal{V}P\mathcal{U} & & \tilde{Q} &=& \mathcal{V}Q \\
\tilde{R} &=& R\mathcal{U} & & \tilde{S} &=& S
\end{array} \tag{3.75}$$

3.3.3.3 Réalisations de mêmes dimensions

Une dernière simplification possible est, en plus des matrices complémentaires nulles, de ne caractériser que les réalisations de mêmes dimensions $n = \tilde{n}$ et $l = \tilde{l}$, ce qui implique que les matrices $\mathcal{U}$, $\mathcal{V}$, $\mathcal{W}$, $\mathcal{T}$, $\mathcal{X}$, $\mathcal{Y}$ doivent être carrées et inversibles avec

$$\mathcal{V} = \mathcal{U}^{-1} \quad \mathcal{T} = \mathcal{W}^{-1} \quad \mathcal{Y} = \mathcal{X}^{-1} \tag{3.76}$$

D'où la proposition suivante

Proposition 3.7

On considère une réalisation $\mathcal{R} = (J, K, L, M, N, P, Q, R, S)$, de fonction de transfert H, de taille l, m, n et p (l la dimension des variables intermédiaires, m est le nombre d'entrées, n la dimension de l'état et p le nombre de sorties).
On considère aussi une autre réalisation $\tilde{\mathcal{R}} = (\tilde{J}, \tilde{K}, \tilde{L}, \tilde{M}, \tilde{N}, \tilde{P}, \tilde{Q}, \tilde{R}, \tilde{S})$ de mêmes dimensions l, m, n et p et telle que

$$\begin{array}{lllllll} \tilde{J} & = & \mathcal{Y}J\mathcal{W} & & & & \\ \tilde{K} & = & \mathcal{U}^{-1}K\mathcal{W} & \tilde{L} & = & L\mathcal{W} & \\ \tilde{M} & = & \mathcal{Y}M\mathcal{U} & \tilde{N} & = & \mathcal{Y}N & \\ \tilde{P} & = & \mathcal{U}^{-1}P\mathcal{U} & \tilde{Q} & = & \mathcal{U}^{-1}Q & \\ \tilde{R} & = & R\mathcal{U} & \tilde{S} & = & S & \end{array} \tag{3.77}$$

avec $\mathcal{U} \in \mathbb{R}^{n\times n}$, $\mathcal{Y} \in \mathbb{R}^{l\times l}$ et $\mathcal{W} \in \mathbb{R}^{l\times l}$ des matrices non-singulières.
En utilisant les matrices Z et $\tilde{Z}$ associées (définies par l'équation (3.16)), les équations (3.77) peuvent se réécrire par les deux similarités suivantes :

$$\tilde{Z} = \begin{pmatrix} \mathcal{Y} & & \\ & \mathcal{U}^{-1} & \\ & & I_p \end{pmatrix} Z \begin{pmatrix} \mathcal{W} & & \\ & \mathcal{U} & \\ & & I_m \end{pmatrix} \tag{3.78}$$

Avec cette construction, ***les réalisations $\tilde{\mathcal{R}}$ et $\mathcal{R}$ sont équivalentes****.*

On remarquera que les similarités sur les matrices J, K, L, M, N, P, Q, R et S, ou sur la matrice Z permettent d'effectuer un changement de base sur les variables intermédiaires (matrices $\mathcal{Y}$ et $\mathcal{W}$) et des matrices d'état (matrice $\mathcal{U}$) de $\mathcal{R}$ pour construire $\tilde{\mathcal{R}}$.
Cela est à rapprocher de l'ensemble des réalisations équivalentes : si l'on se donne une réalisation initiale sous forme d'état (A_0,B_0,C_0,D_0), on sait générer l'ensemble des réalisations équivalentes de même dimension n par un changement de base :

$$\Psi_H = \left\{ \left(\mathcal{T}^{-1}A_0\mathcal{T}, \mathcal{T}^{-1}B_0, \mathcal{T}C_0, D_0 \right) \backslash \forall \mathcal{T} \in \mathbb{R}^{n\times n} \text{ inversible} \right\} \tag{3.79}$$

Les similarités de l'équation (3.77) ou (3.78) nous permettent donc de construire des sous-ensembles de $\mathscr{R}_H$, par similarité sur une réalisation initiale.

3.3.4 Exemples

De plus, il est possible d'exhiber des sous–ensembles de réalisations structurées équivalentes à une similarité près. Par exemple, si on considère la q-structuration et les réalisations $\mathcal{R}$ et $\tilde{\mathcal{R}}$ de la proposition 3.7 :
$\mathcal{R}$ et $\tilde{\mathcal{R}} \in \mathscr{S}_q$ (cf. équation (3.18)) implique $\tilde{J} = J = I_l$, $\tilde{K} = K = 0$, $\tilde{L} = L = 0$, $\tilde{M} = M = 0$ et $\tilde{N} = N = 0$, ce qui impose $\mathcal{Y}\mathcal{W} = I_l$ (on notera aussi que les similarités avec $\mathcal{Y}$ et $\mathcal{W}$ ne servent à rien car les matrices concernées (L, M et N) sont nulles. On peut donc choisir arbitrairement $\mathcal{Y} = \mathcal{W} = I_l$, quoique toute autre valeur soit possible, $\mathcal{Y}$ et $\mathcal{W}$ n'intervenant plus).
Ainsi, si l'on considère une réalisation initiale q-structurée $\mathcal{R}_0 := (Z_0,l,m,n,p)$, on peut exhiber un sous-ensemble Ω de $\mathscr{R}_H^{\mathscr{S}_q}$ par

$$\Omega_H^{\mathscr{S}_q} = \left\{ \mathcal{R} := (Z,l,m,n,p) \;\middle\backslash\; \begin{array}{c} Z = \begin{pmatrix} I_l & & \\ & \mathcal{U}^{-1} & \\ & & I_p \end{pmatrix} Z_0 \begin{pmatrix} I_l & & \\ & \mathcal{U} & \\ & & I_m \end{pmatrix} \\ \forall \mathcal{U} \in \mathbb{R}^{n\times n} \text{ inversible} \end{array} \right\}$$

ce qui permet de retrouver la définition classique de Ψ_H (équation (3.79)), en utilisant le formalisme de la forme implicite spécialisée.
Remarque : on n'a cherché ici que les réalisations q-structurées de même dimension que la réalisation $\mathcal{R}_0$, que l'on peut supposer minimale.

On peut, bien entendu, faire de même pour l'ensemble des réalisations structurées en δ :
$\mathcal{R}$ et $\tilde{\mathcal{R}} \in \mathscr{S}_\delta$ (cf. équation (3.21)) implique $\tilde{J} = J = I_n$, $\tilde{K} = K = -\Delta I_n$, $\tilde{L} = L = 0$, $\tilde{P} = P = I_n$ et $\tilde{Q} = Q = 0$, ce qui impose $\mathcal{U}^{-1}\mathcal{W} = I_n$ et $\mathcal{Y}\mathcal{W} = I_n$ (le même changement de base opère sur les variables d'état et les variables intermédiaires).
Ainsi, si l'on considère une réalisation initiale $\mathcal{R}_0 := (Z_0, l, m, n, p)$ structurée en δ, on peut exhiber un sous-ensemble de $\mathscr{R}_H^{\mathscr{S}_\delta}$ par

$$\Omega_H^{\mathscr{S}_\delta} = \left\{ \mathcal{R} := (Z,n,m,n,p) \;\middle\backslash\; \begin{array}{c} Z = \begin{pmatrix} \mathcal{U}^{-1} & & \\ & \mathcal{U}^{-1} & \\ & & I_p \end{pmatrix} Z_0 \begin{pmatrix} \mathcal{U} & & \\ & \mathcal{U} & \\ & & I_m \end{pmatrix} \\ \forall \mathcal{U} \in \mathbb{R}^{n\times n} \text{ inversible} \end{array} \right\} \tag{3.80}$$

On retiendra que la structuration fixe certaines matrices à des valeurs prédéfinies, ce qui contraint d'autant le changement de base général (3.77), et réduit le nombre de variables de décision et donc la complexité du problème d'optimisation permettant la recherche d'une *bonne* réalisation (cf. chapitre 5).

3.4 Quantification d'une réalisation

Nous avons vu (au chapitre 2) qu'il existait de nombreuses manières de représenter, en précision finie sur N bits[5], les coefficients d'une réalisation. Du choix de cette représentation dépend le coût de calcul mais aussi l'impact de cette quantification.

3.4.1 Représentations regroupées

Nous ne nous intéressons qu'aux représentations en virgule fixe et virgule flottante.
Nous considérons aussi la possibilité de spécifier pour chaque coefficient d'une représentation un format particulier :

- position de la virgule (valeur de β_f cf. 2.2.3) pour la virgule fixe ;
- valeur de l'exposant e pour la virgule flottante (cf. 2.2.4).

Nous considérons les possibilités de spécification suivantes :

- donner à chaque coefficient le format qui lui convient le mieux (qui minimise donc la quantification) ;
- fixer un format particulier pour un ensemble de valeurs : il est évoqué dans [53] la possibilité d'avoir le même format pour les coefficients d'une même matrice intervenant dans la réalisation considérée (un format pour les coefficients de J, un autre pour ceux de K, etc.). Regrouper des coefficients avec un même format permet de simplifier les calculs effectués car cela supprime les opérations nécessaires pour passer d'un format à un autre (un décalage, effectué en logiciel pour la virgule fixe ou en matériel pour la virgule flottante) au sein d'un même groupe de coefficients. Dans le cas de la virgule fixe, il est aussi possible d'envisager un format identique pour tous les coefficients[6].

Nous introduisons donc la notion de *bloc* qui regroupe des coefficients de même format. Différents découpages sont possibles :

- un bloc par coefficient si chaque coefficient est représenté au mieux ;
- un seul bloc si tous les coefficients partagent le même format ;
- neufs blocs associées aux neufs matrices J, K, L, M, N, P, Q, R, S ;
- ou bien encore des blocs quelconques.

Nous avons tout d'abord besoin de distinguer les coefficients de Z qui seront quantifiés de ceux, triviaux, qui ne seront pas *réellement* implémentés,

[5] Ce qui suit est facilement généralisable si on ne considère pas le même nombre de bits pour représenter chaque coefficient (pour coder, par exemple, un coefficient particulier qui requerrait une plus grande précision et un plus grand nombre de bits). Nous ne l'avons pas intégré ici pour ne pas surcharger l'écriture.

[6] On pourra noter que donner en virgule flottante le même format à tous les coefficients reviendrait à fixer le même exposant à toutes les valeurs : celui-ci ne serait plus utile et ce qui reviendrait à utiliser une représentation en virgule fixe. Pour cette raison, le cas "monobloc" ne sera pas considéré dans le cas de la virgule flottante.

tels que les coefficients 0, ±1 (mais aussi les puissances de 2). Pour tenir compte de cette différence, nous introduisons les matrices de pondération W_J, W_K, W_L, W_M, W_N, W_P, W_Q, W_R, W_S, associées aux matrices J, K, L, M, N, P, Q, R, S d'une réalisation soit donc une matrice W_Z associée à Z.

Définition 3.8 (matrice de pondération)
Soit $X \in \mathbb{R}^{a\times b}$. On associe à X une matrice W_X définie par

$$(W_X)_{i,j} \triangleq \begin{cases} 0 & \textit{si } X_{i,j} \textit{ est implémenté exactement} \\ 1 & \textit{sinon} \end{cases} \tag{3.81}$$

On considère, en première approche, que les valeurs 0, 1, -1 et les puissances de 2 seront implémentés exactement, c.-à-d. que la quantification n'aura pas d'effets sur eux.

Ces matrices de pondérations seront également très utiles au chapitre 4.

Le paramètre β_f ou e caractérisant le format d'un coefficient de Z dépend de la plus grande valeur absolue du bloc auquel il appartient ; ceci nous conduit à introduire[7] la matrice η_Z, de même dimension que Z, telle que

$$(\eta_Z)_{i,j} \triangleq \begin{cases} \text{la plus grande valeur absolue du bloc} \\ \text{auquel appartient } Z_{i,j} & \text{si } (W_Z)_{i,j} = 1 \\ 0 & \text{sinon} \end{cases}$$

Considérons les différentes configurations possibles :

– si le découpage par bloc correspond au découpage selon les matrices J, K, L, M, N, P, Q, R, S, alors

$$\eta_Z = \begin{pmatrix} \|J\|_{\max}\mathscr{E}_{l,l} & \|M\|_{\max}\mathscr{E}_{l,n} & \|N\|_{\max}\mathscr{E}_{l,m} \\ \|K\|_{\max}\mathscr{E}_{n,l} & \|P\|_{\max}\mathscr{E}_{n,n} & \|Q\|_{\max}\mathscr{E}_{n,m} \\ \|L\|_{\max}\mathscr{E}_{p,l} & \|R\|_{\max}\mathscr{E}_{p,n} & \|S\|_{\max}\mathscr{E}_{p,m} \end{pmatrix} \tag{3.82}$$

où $\mathscr{E}_{a,b}$ est la matrice de $\mathbb{R}^{a\times b}$ dont tous les éléments valent 1 et $\|.\|_{\max}$ est définie par

$$\|X\|_{\max} \triangleq \max_{i,j} |X_{i,j}| \tag{3.83}$$

– si on choisit le même format pour tous les coefficients :

$$(\eta_Z)_{i,j} = \|Z\|_{\max} \tag{3.84}$$

– si chaque coefficient possède son propre format :

$$(\eta_Z)_{i,j} = |Z_{i,j}| \tag{3.85}$$

[7] Il s'agit d'une généralisation des matrices η_F, η_G, η_J, η_M et η_H considérées dans [120].

On se propose donc de regrouper les cinq représentations suivantes :
- virgule fixe, avec un format adapté à chaque coefficient ;
- virgule fixe par bloc ;
- virgule fixe avec un format unique ;
- virgule flottante par bloc ;
- virgule flottante, avec un format adapté pour chaque coefficient.

sous le même formalisme[8].
Dans les propositions suivantes, le choix de la représentation, nous utiliserons donc η_Z pour indiquer les blocs et l'index α pour désigner le format, selon :

$$\alpha \triangleq \begin{cases} 1 & \text{pour une représentation virgule fixe} \\ 2 & \text{pour une représentation virgule flottante} \end{cases} \tag{3.86}$$

Pour chacune de ces représentations, une plage de bits exprime la *dynamique possible* de la représentation, et une autre la *précision* dans cette plage.
En suivant les notations de [116], on notera β_r le nombre de bits alloués pour la dynamique ($\beta_r = \beta_g$ pour la représentation en virgule fixe ou $\beta_r = \beta_e$ pour la représentation en virgule flottante) et β_p le nombre de bits représentant la précision de la représentation ($\beta_p = \beta_f$ ou $\beta_p = \beta_m$).

3.4.2 Expression du format

Dans le cas de la virgule fixe, on a vu (cf. équation (2.56)) que le paramètre β_f définissant ce format pour le coefficient x est tel que

$$\beta_f \leqslant N - 1 - \lceil \log_2 |x| \rceil \tag{3.87}$$

D'après la définition de η_Z, le paramètre $(\beta_f)_{i,j}$ définissant le format en virgule fixe pour le coefficient $Z_{i,j}$ est donné par

$$(\beta_f)_{i,j} = \begin{cases} N - 1 - \lceil \log_2(\eta_Z)_{i,j} \rceil & \text{si } (W_Z)_{i,j} = 1 \\ 0 & \text{sinon} \end{cases} \tag{3.88}$$

car on prend $(\beta_f)_{i,j}$ le plus grand possible pour représenter tous les éléments du bloc auquel appartient $Z_{i,j}$, (donc le plus grand élément, en valeur absolue, de ce bloc). Par ailleurs, $(\beta_f)_{i,j}$ n'est pas à définir lorsque $Z_{i,j}$ est implémenté exactement.

De la même manière, en virgule flottante par bloc, tous les coefficients s'écrivent sous la forme

$$(-1)^s \cdot m \cdot 2^e \tag{3.89}$$

[8] Dans [53, 116, 121, 120], seuls les formats virgule fixe (avec un même format pour tous les coefficients), virgule flottante (format adapté) et virgule flottante par bloc sont proposés.

où $m \in [0;2[$ [9] et e est fixé pour chaque bloc.
En notant $e_{i,j}$ le paramètre définissant le format virgule flottante pour le coefficient $Z_{i,j}$, on a

$$e_{i,j} = \begin{cases} \lceil \log_2(\eta_Z)_{i,j} \rceil & \text{si } (W_Z)_{i,j} = 1 \\ 0 & \text{sinon} \end{cases} \tag{3.90}$$

3.4.3 Quantification

En connaissant le format de représentation pour chaque élément de Z, il nous est possible de connaître la matrice Z^*, représentant Z après quantification (par arrondi, puisqu'il s'agit d'une *quantification de représentation*). La quantification d'une réalisation, en fonction du format de représentation, est donné par la proposition ci-dessous :

Proposition 3.8
On considère une réalisation $\mathcal{R} := (Z, l, m, n, p)$. Le quantifié sur N bits Z^* de Z est donné, en fonction du choix des blocs exprimé par η_Z et du format α, par

$$Z^*_{i,j} = \begin{cases} Z_{i,j} & si\ (W_Z)_{i,j} = 0 \\ 2^{-\phi_{i,j}} \left\lfloor 2^{\phi_{i,j}} Z_{i,j} \right\rceil & si\ \alpha = 1\ et\ (W_Z)_{i,j} = 1 \\ 2^{\beta_e - \phi_{i,j}} \left\lfloor 2^{\phi_{i,j} - \beta_e} Z_{i,j} \right\rceil & si\ \alpha = 2\ et\ (W_Z)_{i,j} = 1 \end{cases} \tag{3.91}$$

avec

$$\phi_{i,j} = N - 1 - \left\lceil \log_2 (\eta_Z)_{i,j} \right\rceil \tag{3.92}$$

Démonstration :
Dans le cas de la virgule fixe, la démonstration découle du choix de β_f donné à l'équation (3.88) et l'expression de la quantification par arrondi (équation (2.52)).
Pour la virgule flottante, la quantification de Z_{ij} s'écrit $2^{e-\beta_m} \left\lfloor 2^{\beta_m - e} Z_{ij} \right\rceil$ où e est donné par l'équation (3.90) et β_m vérifie l'égalité $1 + \beta_m + \beta_e = N$. ■

3.4.4 Erreur de quantification

Nous avons rappelé, au paragraphe 2.2.4.2, que le processus de quantification modifie un réel x en virgule fixe en $x + \delta$ avec $|\delta| \leqslant 2^{-(\beta_f+1)}$

[9] On remarquera que, pour l'écriture en virgule flottante par bloc, on ne peut imposer la normalisation $m \in [1;2[$. Ainsi, généraliser l'écriture de la virgule flottante *classique* en utilisant la notion de bloc entraîne l'impossibilité de cette normalisation. Ce choix assumé implique que l'on ne considère que des écritures non normalisées (c.-à-d. $m \in [0;2[$ au lieu de $m \in [1;2[$) ; l'écriture normalisée permettant de sauvegarder un bit de représentation.

(quantification par arrondi) et un réel y en virgule flottante en $y(1+\delta)$ avec $|\delta| \leqslant 2^{-(\beta_m+1)}$.
On peut le vérifier grâce au résultat de la proposition 3.8 qui indique

$$\epsilon_{i,j} = \left|Z^*_{i,j} - Z_{i,j}\right| \leqslant \begin{cases} 0 & \text{si } (W_Z)_{i,j} = 0 \\ 2^{-(\beta_f+1)} & \text{si } \alpha = 1 \text{ et } (W_Z)_{i,j} = 1 \\ 2.2^{e_{i,j}}.2^{-(\beta_m+1)} & \text{si } \alpha = 2 \text{ et } (W_Z)_{i,j} = 1 \end{cases} \tag{3.93}$$

Si on note $\Delta_{i,j}$ l'écart sur la précision engendrée par la quantification (donc relatif aux β_p bits), il vient

$$|\Delta_{i,j}| \leqslant 2^{-(\beta_{p_{i,j}}+1)} \tag{3.94}$$

β_p est identique pour tous les coefficients dans le cas de la virgule fixe et dépend de la plus grande valeur absolue du bloc auquel appartient $Z_{i,j}$ dans le cas de la virgule fixe (cf. équation (3.88)).
La proposition suivante permet d'exprimer l'erreur de quantification sur Z :

Proposition 3.9
Lors de la quantification, Z est modifié en $Z + r_Z \times \Delta$ où r_Z est donné par[10]

$$r_Z \triangleq \begin{cases} W_Z & si\ \alpha = 1 \\ 2\eta_Z \times W_Z & si\ \alpha = 2 \end{cases} \tag{3.95}$$

Δ l'écart sur la précision engendrée (et vérifie l'équation (3.94)) et $\times$ le produit de Schur *(produit direct terme à terme).*

Cette écriture nous permettra, au chapitre suivant, de comparer différents indicateurs avant et après la quantification (en Z et $Z + r_Z \times \Delta$) et d'introduire des critères de sensibilité pertinents en virgule fixe et aussi en virgule flottante. De plus, en majorant $\|\Delta\|_{\max}$ par

$$\|\Delta\|_{\max} \leqslant 2^{-(\beta_p+1)} \tag{3.96}$$

il sera possible d'estimer un nombre de bits de précision β_p minimum pour assurer que la valeur de certains critères soit inférieure à un seul donné.

3.4.5 Nombre de bits minimum

Pour assurer que tous les coefficients puissent être correctement représentés dans un format donné, c'est à dire qu'ils soient codés sans *overflow*

[10] On notera que, dans le cas de la virgule flottante *classique*, où chaque coefficient possède son propre exposant, on a $r_Z = Z \times W_Z$, alors que l'équation (3.95) donne $_Z = 2Z \times W_Z$. Ce facteur 2 provient du choix de l'écriture non normalisée choisie, cf. commentaire 9.

(dépassement de la valeur maximale) ni *underflow* (valeur non représentable car inférieure à la valeur minimum, la valeur est alors codée par 0), un nombre de bits minimum est nécessaire.
La proposition suivante permet d'évaluer ce nombre :

Proposition 3.10
On considère une réalisation $\mathcal{R} := (Z, l, m, n, p)$ *avec un codage sur* N *bits, caractérisée par le choix de* α *et* η_Z*.*
Pour éviter les overflow et underflow, N *doit vérifier*

$$N \geqslant \begin{cases} \beta_{\min} & si\ \alpha = 1 \\ \beta_{\min} + \beta_e + 1 & si\ \alpha = 2 \end{cases} \tag{3.97}$$

avec

$$\beta_{\min} = \max_{i,j\ tq\ \substack{(W_Z)_{i,j}=1 \\ Z_{i,j}\neq 0}} \left\lceil \log_2 (\eta_Z)_{i,j} \right\rceil - \lceil \log_2 |Z_{i,j}| \rceil + 1$$

$$\beta_e = \left\lceil \log_2 \left(\left\lfloor \log_2 \|\eta_Z \times W_Z\|_{\max} \right\rfloor - \left\lfloor \log_2 \|\eta_Z \times W_Z\|_{\min} \right\rfloor + 1 \right) \right\rceil$$

et où $\|.\|_{\min}$ *est défini par*

$$\|X\|_{\min} \triangleq \min_{i,j} \{|X_{i,j}|\ \ tq\ X_{i,j} \neq 0\} \tag{3.98}$$

Démonstration :

– Dans le cas de la virgule fixe, l'absence d'*overflow* est assurée par le choix de la position de la virgule. Dans le même temps, pour éviter un *underflow*, chaque coefficient $Z_{i,j}$ non nul et implémenté doit vérifier

$$|Z_{i,j}| \geqslant 2^{-(\beta_f+1)}$$

car $2^{-(\beta_f+1)}$ est la plus petite valeur représentable. D'où

$$N > \left\lceil \log_2 (\eta_Z)_{i,j} \right\rceil - \lceil \log_2 |Z_{i,j}| \rceil$$

– En virgule flottante, le nombre de bits β_e alloués à l'expression de l'exposant est, soit fixé par la norme (IEEE754), soit tel que tous les $e_{i,j}$ puissent s'écrire sur β_e bits. Il est tel que[11]

$$\left(\max_{i,j} e_{i,j} \right) - \left(\min_{i,j} e_{i,j} \right) \leqslant 2^{\beta_e} - 1$$

[11] On notera qu'en toute rigueur on devrait écrire $\left(\max_{i,j} e_{i,j} \right) - \left(\min_{i,j} e_{i,j} \right) \leqslant 2^{\beta_e} - 2$ car les valeurs significatives, *NaN* et $\pm\infty$ sont aussi codées sur l'exposant, et réduisent la plage de valeurs possibles pour représenter e.

D'où

$$\beta_e \geqslant \left\lceil \log_2 \left(\left\lfloor \log_2 \|\eta_Z \times W_Z\|_{\max} \right\rfloor - \left\lfloor \log_2 \|\eta_Z \times W_Z\|_{\min} \right\rfloor + 1 \right) \right\rceil$$

(cette expression est valide car on suppose qu'il existe au moins un coefficient quantifié non nul). On choisira donc la valeur de β_e donnée par l'équation (3.98) pour que toutes les valeurs puissent être représentées sans overflow.
De plus, pour éviter les underflow, la plus petite valeur de Z doit pouvoir être représentée. La mantisse d'un flottant est codée sur β_m bits, et est comprise entre $[0; 2 - 2^{-\beta_m+2}]$ avec un pas de quantification de $2^{-\beta_m+2}$. La plus petite valeur exprimable est donc $2^{-\beta_m+2-e_{i,j}}$. $Z_{i,j}$ doit donc vérifier

$$|Z_{i,j}| \geqslant 2^{-\beta_m+2}$$

d'où

$$\beta_m \geqslant \left\lceil \log_2 (\eta_Z)_{i,j} \right\rceil - \left\lceil \log_2 |Z_{i,j}| \right\rceil + 2 = \beta_{\min} + 1$$

De ces deux bornes sur le nombre de bits nécessaires pour exprimer la mantisse (β_m) et l'exposant (β_e), on obtient le nombre de bits nécessaires.

■

Remarque 1 : on notera que ce nombre de bits minimum n'est valable que si l'on respecte le format choisi pour chaque coefficient au paragraphe 3.4.2. En effet, on peut trouver une position de la virgule (virgule fixe) ou un nombre de bits d'exposant (virgule flottante) tel qu'une représentation avec le nombre de bits minimum énoncé dans cette proposition amène un overflow ou un underflow. Il s'agit ici de représenter les nombres au plus juste.
Remarque 2 : La proposition montre que la virgule fixe nécessite moins de bits que la virgule flottante pour représenter des coefficients : cela est du au fait que l'exposant de la virgule fixe est implicite et non stocké.
Remarque 3 : si l'on considère un format différent pour chaque coefficient, un bit, en virgule fixe, suffit (chaque coefficient est alors représenté par la puissance de 2 la plus proche, l'exposant étant implicite), tandis que cinq sont nécessaires en virgule flottante.
Remarque 4 : ce nombre de bits minimum ne présage pas de la complexité de représenter Z, mais uniquement de la complexité de représenter, selon un format défini, une approximation de Z en préservant les ordres de grandeur. On se référera à [60, 16] et [107] pour une évaluation de la complexité de représentation, au travers de la notion de complexité de *Kolmogorov-Chaitin*.

Cette proposition nous indique aussi, pour la virgule fixe, le nombre de bits de dynamique minimum : on peut rapprocher ce résultat de celui donné dans [116], par exemple.

3.4.6 Exemple

On considère l'exemple (fictif) suivant :

$$Z = \left(\begin{array}{cc:c} & & \\ \hdashline 0.12788 & -0.45227 & 1.21341 \\ 0.36411 & 0.87909 & 0.12399 \\ \hdashline 2.68100 & 6.18965 & 12.89766 \end{array}\right) \tag{3.99}$$

ainsi que les cinq formats :

- Z_1^* : virgule fixe, avec un format adapté à chaque coefficient ;
- Z_2^* : virgule fixe par bloc, découpage selon les matrices J, K, ... , S ;
- Z_3^* : virgule fixe avec un format unique ;
- Z_4^* : virgule flottante, avec un format adapté pour chaque coefficient ;
- Z_5^* : virgule flottante par bloc, découpage selon les matrices J, K, ..., S.

Le nombre de bits minimum pour représenter Z sans *overflow* ni *underflow* est donné par le tableau suivant :

Si on considère une représentation avec 9 bits, on a, d'après la proposition

format	nb de bits minimum
Z_1^*	1
Z_2^*	5
Z_3^*	8
Z_4^*	5
Z_5^*	9

TAB. 3.1 – Nombre de bits minimum pour représenter Z selon les formats

3.8 et l'expression du format du paragraphe 3.4.2 :

$$Z_1^* = \left(\begin{array}{cc:c} & & \\ \hdashline 0.12793 & -0.45312 & 1.21094 \\ 0.36328 & 0.87891 & 0.12402 \\ \hdashline 2.68750 & 6.18750 & 12.87500 \end{array}\right) \tag{3.100}$$

$$Z_2^* = \left(\begin{array}{cc:c} & & \\ \hdashline 0.12891 & -0.45312 & 1.21094 \\ 0.36328 & 0.87891 & 0.12500 \\ \hdashline 2.68750 & 6.18750 & 12.87500 \end{array}\right) \tag{3.101}$$

$$Z_3^* = \left(\begin{array}{cc:c} & & \\ \hdashline 0.12500 & -0.43750 & 1.18750 \\ 0.37500 & 0.87500 & 0.12500 \\ \hdashline 2.68750 & 6.18750 & 12.87500 \end{array}\right) \tag{3.102}$$

$$Z_4^* = \left(\begin{array}{cc:c} & & \\ \hdashline 0.12500 & -0.43750 & 1.25000 \\ 0.37500 & 0.87500 & 0.12500 \\ \hdashline 2.75000 & 6.00000 & 13.00000 \end{array}\right) \tag{3.103}$$

$$Z_5^* = \left(\begin{array}{c|cc|c} & & & \\ \hline & 0.12500 & -0.43750 & 1.18750 \\ & 0.37500 & 0.87500 & 0.12500 \\ \hline & 2.75000 & 6.25000 & 13.00000 \end{array}\right) \tag{3.104}$$

En ne considérant que la représentation virgule fixe, sur 5 bits, avec un format commun aux matrices qui composent Z, le quantifié s'écrit

$$Z_6^* = \left(\begin{array}{c|cc|c} & & & \\ \hline & 2 \cdot 2^{-4} & -7 \cdot 2^{-4} & 10 \cdot 2^{-3} \\ & 6 \cdot 2^{-4} & 14 \cdot 2^{-4} & 1 \cdot 2^{-3} \\ \hline & 5 \cdot 2^{-1} & 12 \cdot 2^{-1} & 13 \cdot 2^{0} \end{array}\right) \tag{3.105}$$

ou encore, en binaire (avec le bit de signe en gras)

$$Z_6^* = \left(\begin{array}{c|cc|c} & & & \\ \hline & \mathbf{0}_{\triangle}0010 & \mathbf{1}_{\triangle}1001 & \mathbf{0}1_{\triangle}010 \\ & \mathbf{0}_{\triangle}0110 & \mathbf{0}_{\triangle}1110 & \mathbf{0}0_{\triangle}001 \\ \hline & \mathbf{0}010_{\triangle}1 & \mathbf{0}110_{\triangle}0 & \mathbf{0}1101_{\triangle} \end{array}\right) \tag{3.106}$$

ou encore

Remarque 1 : la valeur des exposants (implicite) de la dernière formulation de Z_6^* nous permet d'écrire les majorations $\left|(Z - Z_6^*)_{i,j}\right| \leqslant \Delta_{i,j}$ avec

$$\Delta = \left(\begin{array}{c|cc|c} & & & \\ \hline & 2^{-5} & 2^{-5} & 2^{-4} \\ & 2^{-5} & 2^{-5} & 2^{-4} \\ \hline & 2^{-2} & 2^{-2} & 2^{-1} \end{array}\right) \tag{3.107}$$

Remarque 2 : On note bien qu'un bit de moins entraînerait un underflow pour le paramètre $1 \cdot 2^{-3}$, car la valeur absolue de l'autre coefficient du même bloc s'écrit déjà sur 4 bits ($10 \cdot 2^{-3}$).

3.5 Conclusion

Nous avons introduit dans ce chapitre, un formalisme nouveau de représentation des réalisations possibles pour une loi donnée : la forme implicite spécialisée proposée permet de décrire, de manière unifiée, les paramétrisations envisagées jusqu'à présent dans la littérature. Il s'agit là, nous semble-t-il, d'une contribution importante de la thèse qui est illustrée autour de nombreuses structurations données en exemple, comme les réalisations en q ou δ, mixtes q et δ, et d'autres possibilités encore.
Le principe d'inclusion, étendu ici aux cas des réalisations implicites spécialisées proposées, permet de décrire les classes d'équivalence qui seront utilisées au chapitre 5, afin de rechercher au sein de ces classes des réalisations optimales au sens de critères à définir. Enfin, l'opération de quantification des

coefficients d'une réalisation (regroupés dans les matrices J, K, L, M, N, P, Q, R, S ou la matrice unique Z) est formalisée de manière unifiée.
Il nous sera donc possible, au chapitre 4, de mettre en place différents critères d'évaluation de la dégradation induite par la quantification sans avoir à distinguer de cas particuliers.

Chapitre

4 Critères d'analyse de réalisations en précision finie

Résumé :

Nous allons, dans ce chapitre, associer à la forme implicite spécialisée présentée au chapitre 3 différents outils d'analyse permettant d'évaluer les coûts logiciels ou le degré de résilience d'une réalisation particulière donnée. Ces *mesures* développées et adaptées pour la forme implicite permettront d'évaluer et de comparer l'impact de l'opération de l'implémentation sur différentes réalisations équivalentes (en précision finie).

Sommaire

Nous avons vu, dans les paragraphes précédents, que l'implémentation en précision finie amenait une dégradation de la loi, et qu'une des sources de dégradation était la modification des paramètres due à la quantification. Nous avons vu également qu'il existait de nombreuses formes de réalisations mathématiquement équivalentes.
Les mesures proposées dans ce chapitre ont pour but d'étudier l'impact de cette dégradation sur une réalisation, quelle que soit sa structure. Il pourra s'agir d'évaluer la modification des caractéristiques intrinsèques d'une loi (comportement entrées/sorties, comportement global, etc.) ou bien , simplement, d'évaluer le coût logiciel (nombres d'opérations, ...) d'une réalisation. Ces mesures permettent de comparer plusieurs réalisations entre elles, et tout particulièrement comparer des réalisations mathématiquement équivalentes, qui ne le sont plus après quantification. Elles aident donc à la recherche de réalisations minimisant cette dégradation : la synthèse de réalisations, avec un critère lié à la quantification, sera abordée au chapitre 5.

4.1 Coûts logiciels

Les coûts logiciels liés à une réalisation sont difficiles à évaluer sans autres informations que celles données par les paramètres liés à la forme implicite (équation (3.4)). Ces coûts dépendent principalement du matériel (*Hardware*) utilisé pour les calculs et de la manière dont le code est écrit (*Software*).

Le calculateur peut disposer d'unités de calcul spécialisées (unité MAC pour accélerer le calcul de produits scalaires, avec possibilité de gérer au mieux les calculs en virgule fixe ; multiplication réalisée en hardware ou non (ce qui implique une multiplication logicielle) ; etc...) et chaque opération demande plus ou moins de temps. De plus, le logiciel peut mettre en œuvre, de manière plus ou moins efficace, les calculs nécessaires : de nombreux degrés d'optimisation sont possibles (au détriment, éventuellement, de la portabilité et de la lisibilité du code), les calculs peuvent être écrits sous forme matricielle ou bien développés sous forme scalaire, le matériel plus ou moins bien exploité (nécessité d'écrire des portions de code en assembleur, ou à défaut de maîtriser le code assembleur produit par le compilateur pour utiliser les particularités du matériel).
Le logiciel peut aussi dépendre du processus de génération de code : codage automatique (depuis des planches Matlab/Simulink, planches Scade, Ascet, ...), semi-automatique ou entièrement manuel. La qualité du codage n'est alors pas la même : le code qui est produit automatiquement (en temps très court) peut présenter un degré d'optimisation beaucoup plus faible qu'un codage manuel. En contrepartie, le code produit automatiquement est conforme aux spécifications initiales, et peut même, selon les cas, être

certifié à moindre coût.

Face à cette difficulté, deux indicateurs nous permettent, à défaut d'évaluer réellement le coût logiciel, de rendre compte de la *complexité* d'une réalisation :

- la quantité de mémoire utilisée ;
- le nombre d'opérations (le nombre d'additions et de multiplications, ou encore le nombre de "multiplication/accumulation", puisque toutes les opérations peuvent s'écrire comme des produits scalaires).

Ceux-ci évoluent bien entendu avec l'ordre de la loi, mais aussi avec la structure choisie pour la réalisation.

4.1.1 Coût mémoire

La taille mémoire utilisée dépend de la taille du vecteur d'état de la réalisation implicite et du nombre d'octets requis pour représenter un élément de l'état. De plus, à chaque itération de calcul, les variables intermédiaires sont utilisées : même si leurs valeurs ne sont plus reprises d'une itération sur l'autre, ces variables participent à la quantité de mémoire requise pour le calcul[1].
En supposant que chaque élément du vecteur d'état et du vecteur des variables intermédiaires prend la même place en mémoire (même nombre d'octets), le coût de mémoire est évalué par

$$n + l \tag{4.1}$$

Ce critère sera donc mis en avant si l'on s'intéresse à la taille mémoire requise par la réalisation numérique. Considérons également un autre critère souvent plus critique, dans les applications automobiles : celui du coût de calcul, que l'on peut rapprocher du temps de calcul.

4.1.2 Coût de calcul

Afin évaluer le coût de calcul d'une réalisation (en nombre d'additions et de multiplications), introduisons la proposition suivante :

Proposition 4.1
Soit $Y \in \mathbb{R}^{a\times b}$ une constante, et $V \in \mathbb{R}^{b\times 1}$ une variable.
Le calcul YV requiert $a(b-1) - n_Y^0$ additions et $ab - n_Y^1$ multiplications, où n_Y^0 est le nombre d'éléments nuls de Y et n_Y^1 est le nombre d'éléments triviaux (0,1,-1) de Y (ces éléments n'entraînent pas de multiplications)

[1] Dans le cas où ce nombre de variables intermédiaires est faible, les valeurs intermédiaires du calcul qui représentent ces variables seront uniquement stockées dans les registres généraux du processeur, et non réellement en mémoire.

Démonstration :
Pour le calcul de YV, il y ab multiplications, mais celles par une constante nulle ou ± 1 n'entrainent pas réellement de multiplication.
Il y a, de plus, $a(b-1)$ additions pour YV, mais on ne compte pas les additions où une opérande est nulle[2]. ■

Remarque : On pourrait encore affiner cette évaluation en considérant les éléments de Y qui sont des puissances de 2 : lorsque les nombres sont représentés avec un format binaire classique (codage binaire, complément à 2, virgule fixe, ...), la multiplication par une puissance de 2 n'a pas le même coût qu'une multiplication quelconque puisqu'elle consiste en un décalage de bits (qui s'effectue généralement en un cycle de calcul, plus rapidement qu'une multiplication).
On pourrait aussi prendre en compte le mode de représentation des éléments : si deux éléments, en virgule fixe, n'ont pas le même facteur d'échelle (même position de la virgule), un décalage de bits est nécessaire entre chaque addition pour recadrer les éléments.

Ainsi, le nombre d'additions et de multiplications de la proposition 4.1 correspond aux opérations mathématiques à effectuer par le calculateur numérique, mais pas forcément exactement le nombre d'instructions *additions* et *multiplications* réalisées.
Ceci étant, cette première approche est tout à fait suffisante pour comparer différentes réalisations entre elles, grâce à la proposition 4.2 :

Proposition 4.2
Soit $\mathcal{R} := (Z, l, m, n, p)$ une réalisation.
Une itération, selon l'algorithme de la définition 3.1, nécessite

$$\begin{array}{ll} (l+n+p)(l+m+n-1) - l - n_Z^0 & \textit{additions} \\ (l+n+p)(l+m+n) - n_Z^1 & \textit{multiplications} \end{array}$$

où n_Z^0 est le nombre d'éléments nuls de Z et n_Z^1 est le nombre d'éléments triviaux (0,1,-1) de Z

Démonstration :
On applique la proposition 4.1 aux étapes [1], [2] et [3] de l'algorithme de calcul, regroupées ici :

[2] V étant une valeur variable, on ne peut pas supposer connu *a priori* le nombre de paramètres nuls ou triviaux qu'il contient.

$$
\begin{array}{ll}
1: & T(k+1) \leftarrow J^{-1} \begin{pmatrix} M & N \end{pmatrix} \begin{pmatrix} X(k) \\ U(k) \end{pmatrix} \\
2: & X(k+1) \leftarrow \begin{pmatrix} K & P & Q \end{pmatrix} \begin{pmatrix} T(k+1) \\ X(k) \\ U(k) \end{pmatrix} \\
3: & Y(k) \leftarrow \begin{pmatrix} L & R & S \end{pmatrix} \begin{pmatrix} T(k+1) \\ X(k) \\ U(k) \end{pmatrix}
\end{array}
$$

Les tableaux suivants regroupent le nombre d'opérations pour chaque étape

	additions
[1]	$l(n+m-1) - n_M^0 - n_N^0 + l(l-1) - n_J^0$
[2]	$n(l+n+m-1) - n_K^0 - n_P^0 - n_Q^0$
[3]	$p(l+n+m-1) - n_L^0 - n_R^0 - n_S^0$

	multiplications
[1]	$l(n+m) - n_M^1 - n_N^1 + l^2 - n_J^1$
[2]	$n(l+n+m) - n_K^1 - n_P^1 - n_Q^1$
[3]	$p(l+n+m) - n_L^1 - n_R^1 - n_S^1$

De plus

$$
\begin{aligned}
n_Z^0 &= n_J^0 + n_K^0 + n_L^0 + n_M^0 + n_N^0 + n_P^0 + n_Q^0 + n_R^0 + n_S^0 \\
n_Z^1 &= n_J^1 + n_K^1 + n_L^1 + n_M^1 + n_N^1 + n_P^1 + n_Q^1 + n_R^1 + n_S^1
\end{aligned}
$$

■

Si l'on considère les diverses réalisations vues au chapitre 2 et 3, on peut, en 1re approche, comparer leurs coûts de calcul respectifs. Le tableau 4.1 donne le nombre d'additions et de multiplications d'une réalisation SISO d'ordre n pour quelques structurations les plus simples. Notons à

structuration	nb additions	nb multiplications
forme d'état classique[3]	n^2+n	n^2+2n+1
Forme directe I[4]	$2n-1$	$2n$
Forme directe II[4]	$2n-1$	$2n$
Réalisation en δ	n^2+2n	n^2+3n+1
Forme directe II en δ	$3n+1$	$3n$

TAB. 4.1 – Coût de calcul de différentes réalisations d'ordre n

ce stade que les formes directes sont parmi celles qui présentent le moins d'opérations.

[3] Avec une réalisation minimale, sans présager de la valeur des coefficients.
[4] On suppose $a_0 = 1$.

4.2 Sensibilité de la fonction de transfert

4.2.1 Historique

Les premiers travaux sur l'implémentation en précision finie ([100, 31]) ont porté sur une mesure concernant la modification de la fonction de transfert lorsque ses coefficients sont modifiés par la quantification.
Considérons le système SISO décrit par la réalisation d'état classique (A, B, C, D), d'ordre n :

$$\begin{cases} X(k+1) &= AX(k) + BU(k) \\ Y(k) &= CX(k) \end{cases} \tag{4.2}$$

La fonction de transfert associée vérifie $H(z) = C(zI_n - A)^{-1}B$.
Puisque la quantification de la réalisation modifie les coefficients des matrices A, B, C et D et donc la fonction de transfert H, Tavşanoğlu et Thiele [100] ont étudié la déviation de la fonction de transfert induite et considéré pour cela la sensibilité de la fonction de transfert H vis-à-vis des coefficients A, B et C.

Une définition de la sensibilité est donné ci-dessous :

Définition 4.1 (Sensibilité d'une fonction vis-à-vis d'une matrice)
Soit $X \in \mathbb{R}^{m\times n}$ une matrice et $f : \mathbb{R}^{m\times n} \to \mathbb{C}$ une fonction matricielle à valeur dans $\mathbb{C}$, différentiable selon tous les éléments de X.
On définit la sensibilité de f par rapport à X comme la matrice de $\mathbb{K}^{m\times n}$ dont les $(i,j)^{ème}$ éléments vérifient :

$$S_X \triangleq \frac{\partial f}{\partial X} \qquad avec \qquad (S_X)_{i,j} \triangleq \frac{\partial f}{\partial X_{i,j}} \tag{4.3}$$

Cette sensibilité définira la dérivation matricielle de la fonction f par rapport à X.

Remarque : les définitions et propositions ayant trait aux dérivations matricielles sont regroupées dans l'annexe A.

Par extension, on définira la sensibilité d'une fonction de transfert :

Définition 4.2 (Sensibilité d'une fonction de transfert)
Soit $X \in \mathbb{R}^{m\times n}$ une matrice et $H : \mathbb{C} \to \mathbb{C}$ une fonction de transfert dont l'expression $H(z)$ est fonction de X (et différentiable pour tout z selon tous les éléments de X).
La sensibilité de la fonction de transfert H par rapport à X est la fonction de transfert $\mathbb{C} \to \mathbb{C}^{m\times n}$, noté $\frac{\partial H}{\partial X}$, et qui vérifie

$$\frac{\partial H}{\partial X}(z) = \frac{\partial (H(z))}{\partial X} \qquad \forall z \in \mathbb{C} \tag{4.4}$$

Ceci permet de définir les sensibilités S_A, S_B et S_C de la fonction de transfert H vis-à-vis de A, B, C, et on a facilement

$$S_A \triangleq \frac{\partial H}{\partial A} = H_1^\top H_2 \tag{4.5}$$

$$S_B \triangleq \frac{\partial H}{\partial B} = H_1^\top \tag{4.6}$$

$$S_C \triangleq \frac{\partial H}{\partial C} = H_2 \tag{4.7}$$

où

$$H_1 : z \mapsto (zI_n - A)^{-1}B \qquad \text{et} \qquad H_2 : z \mapsto C(zI_n - A)^{-1} \tag{4.8}$$

Deux définitions sont encore nécessaires pour mesurer l'effet global, sur toutes les fréquences, des sensibilités S_A, S_B et S_C :

Définition 4.3 (Norme de Frobenius)
Soit $M \in \mathbb{C}^{m\times n}$.
La norme de Frobenius $\|M\|_F$ de M est le réel défini par

$$\|M\|_F \triangleq \sqrt{\sum_{i,j} |M_{i,j}|^2} \tag{4.9}$$

Définition 4.4 (Norme L_p)
Soit $f : \mathbb{C} \to \mathbb{C}^{m\times n}$ une fonction de la variable complexe z.
La norme L_p de f est le réel positif défini par

$$\|f\|_p \triangleq \left(\frac{1}{2\pi}\int_0^{2\pi} \left\|f\left(e^{j\omega}\right)\right\|_F^p \, d\omega\right)^{\frac{1}{p}} \tag{4.10}$$

La mesure de sensibilité de la fonction de transfert proposée par V. Tavşanoğlu et L. Thiele est :

$$M_{L_{12}} \triangleq \left\|\frac{\partial H}{\partial A}\right\|_1^2 + \left\|\frac{\partial H}{\partial B}\right\|_2^2 + \left\|\frac{\partial H}{\partial C}\right\|_2^2 \tag{4.11}$$

On remarquera que cette mesure fait intervenir les normes L_1 et L_2, ce qui explique la notation $M_{L_{12}}$ utilisée[5].

M. Gevers et G. Li [31] ont montré que, parmi toutes les réalisations équivalentes de même dimension, une réalisation équilibrée (cf. définition 2.1) minimisait ce critère $M_{L_{12}}$.

[5]Pour illogique qu'il soit, le mélange des normes L_1 et L_2 est relativement répandu car des réalisations optimales au sens de la mesure $M_{L_{12}}$ peuvent être obtenues analytiquement, et que ces réalisations minimisent également les bruits de quantification.

Ils proposent toutefois une mesure de sensibilité n'utilisant que la norme L_2 :

$$M_{L_2} \triangleq \left\| \frac{\partial H}{\partial A} \right\|_2^2 + \left\| \frac{\partial H}{\partial B} \right\|_2^2 + \left\| \frac{\partial H}{\partial C} \right\|_2^2 \tag{4.12}$$

La solution au problème d'optimisation

$$\min_{\substack{T \in \mathbb{R}^{n \times n} \\ \det T \neq 0}} M_{L_2} \left(T^{-1}AT, T^{-1}B, CT \right) \tag{4.13}$$

ne peut être trouvée analytiquement dans ce cas, d'où l'utilisation d'une optimisation numérique, avec un algorithme du type gradients, par exemple (plus de détail dans [31]).

Il est également possible d'adjoindre aux ci-dessus des pondérations fréquentielles. En effet, les normes L_1 et L_2 considèrent de manière égale toutes les fréquences. En pratique, les exigences sur les détériorations dues à la précision finie portent sur une plage de fréquences restreinte. Ces pondérations fréquentielles, dénommées W_A, W_B et W_C permettent l'expression enrichie :

$$M_{L_2} \triangleq \left\| W_A \frac{\partial H}{\partial A} \right\|_2^2 + \left\| W_B \frac{\partial H}{\partial B} \right\|_2^2 + \left\| W_C \frac{\partial H}{\partial C} \right\|_2^2 \tag{4.14}$$

M. Gevers et G. Li ont également travaillé à expliciter ces mesures dans le cas de la paramétrisation en δ :

$$M_{\delta,L_{12}} \triangleq \left\| \frac{\partial H_\delta}{\partial A_\delta} \right\|_1^2 + \left\| \frac{\partial H_\delta}{\partial B_\delta} \right\|_2^2 + \left\| \frac{\partial H_\delta}{\partial C_\delta} \right\|_2^2 \tag{4.15}$$

où H_δ est la fonction de transfert exprimée au moyen de la variable complexe ρ, associée à l'opérateur δ :

$$H_\delta(\rho) = C_\delta \left(\rho I_n - A_\delta \right)^{-1} B_\delta \qquad \forall \rho \in \mathbb{C} \tag{4.16}$$

On remarquera que cette mesure étudie la sensibilité de la fonction de transfert H_δ exprimée avec la variable ρ et non la fonction de transfert H exprimée avec la variable z, vis-à-vis des coefficients A_δ, B_δ et C_δ. On verra aussi, par la suite, que cette sensibilité ne prend pas tous les paramètres en jeu : le paramètre Δ, qui intervient dans la définition de l'opérateur δ (cf. équation (2.20)), est aussi présent dans les calculs et donc assujetti à l'effet de la quantification.

4.2.2 Extension à la forme implicite

Ces mesures introduites par V. Tavşanoğlu et L. Thiele puis par M. Gevers et G. Li sont intéressantes parce qu'elles expriment la sensibilité de la fonction de transfert vis-à-vis des coefficients, et donc de sa modification lors

de la quantification des coefficients due à l'implémentation, mais présentent tout de même trois limitations que nous levons dans la suite de ce chapitre, en étendant cette mesure, ainsi que d'autre, à la forme implicite spécialisée :

- Ces mesures ne concernent que les réalisations sous forme d'état classique (explicite) et celles avec l'opérateur δ (on trouve aussi dans [31] une extension pour une implémentation particulière de la forme retour d'état observateur) et est donc spécifique à une réalisation, à une structuration donnée.
- De plus, les expressions de S_A, S_B et S_C (équations (4.5) à (4.7)) sont spécifiques aux lois SISO, bien qu'il soit possible de les étendre aux cas MISO[6], SIMO[7] et MIMO[8] en appliquant, pour chaque transfert élémentaire, le calcul de sensibilité et en utilisant la propriété suivante :

  Proposition 4.3 (Propriété de la norme L_2)
  Soit $G : \mathbb{C} \to \mathbb{C}^{m\times n}$ une fonction de transfert MIMO, et $(G_{i,j})_{\substack{1\leqslant i\leqslant m\\1\leqslant j\leqslant n}}$ les transferts élémentaires de G (tels que $(G(z))_{i,j} = G_{i,j}(z)\ \forall z \in \mathbb{C}$). On a alors
  $$\|G\|_2^2 = \sum_{\substack{1\leqslant i\leqslant m\\1\leqslant j\leqslant n}} \|G_{i,j}\|_2^2 \tag{4.17}$$

- Enfin, ces mesures de sensibilité de fonction de transfert ne prennent pas en compte le caractère creux des matrices A, B, C ni les paramètres triviaux qui les composent et qui ne seront pas quantifiés lors de l'implémentation et dont il ne faut pas se soucier de leur impact (ce sont des paramètres *transparents*). Pour une forme creuse, comme la forme canonique, la sensibilité de tous les coefficients y est considérée, y compris les 0 et les 1 dont on sait qu'ils ne seront pas réellement présents dans le calcul[69] : en effet, il n'y aura pas de multiplication par 1, ni d'addition et multiplication par 0, hormis si les calculs sont réellement implémentés sous forme matricielle (mais dans ce cas, les coefficients ne seront pas tronqués, mais représentés exactement).

4.2.2.1 Expression de la mesure dans le cas SISO

Ces constatations nous amènent à étendre la mesure de sensibilité de la fonction de transfert vis-à-vis des coefficients à la forme implicite spécialisée. Une première mesure considérée est une mesure M_{L_2} définie par

$$M_{L_2} \triangleq \sum_{X\in\{J,K,L,M,N,P,Q,R,S\}} \left\| \frac{\partial \tilde{H}}{\partial X} \right\|_2^2 \tag{4.18}$$

[6] *Multiple Input Single Output.*
[7] *Single Input Multiple Output.*
[8] *Multiple Input Multiple Output.*

avec

$$\tilde{H}(z) \triangleq H(z) - D = C(zI_n - A)^{-1}B \qquad \forall z \in \mathbb{C} \tag{4.19}$$

On considère ici la sensibilité de la fonction de transfert $\tilde{H}$ plutôt que la sensibilité de H, car le terme $D = LJ^{-1}N + S$ est indépendant du choix de réalisation choisie et n'a pas à être considéré. On remarquera que, dans le cas d'une réalisation sous forme d'état classique, cette mesure est identique à la mesure M_{L_2} de M. Gevers et G. Li (équation (4.12)).

Dans les nombreux exemples de réalisations exprimées sous forme implicite spécialisée (section 3.2), nous avons vu qu'un certain nombre de matrices J, K, L, M, N, P, Q, R, S étaient nulles ou égales à l'identité. Pour tenir compte de cette particularité dans la mesure de sensibilité (la sensibilité de la fonction de transfert vis-à-vis de ces coefficients ne doit pas être considérée puisqu'ils ne seront pas modifiés lors du processus de quantification), il nous faut donc utiliser les matrices de pondération W_J, W_K, ..., W_S de manière à distinguer les coefficients affectés ou non par la quantification (cf. définition 3.8).
On considère, en première approche, que les valeurs 0, 1, -1 et les puissances de 2 seront implémentées exactement.

On élabore donc une nouvelle mesure de sensibilité de la fonction de transfert H, prenant en compte ces pondérations [43] :

Définition 4.5 (Mesure de sensibilité pondérée)
On considère une réalisation SISO $\mathcal{R} = (J, K, L, M, N, P, Q, R, S)$ et les matrices de pondération W_J, W_K, W_L, W_M, W_N, W_P, W_Q, W_R et W_S associées. La mesure de sensibilité de la fonction de transfert, pondérée pour prendre en compte les coefficients non quantifiés, est définie par

$$M_{L_2}^W \triangleq \sum_{X \in \{J,K,L,M,N,P,Q,R,S\}} \left\| \frac{\partial \tilde{H}}{\partial X} \times W_X \right\|_2^2 \tag{4.20}$$

où $\times$ est le produit de Schur.
Cette mesure peut aussi s'écrire

$$M_{L_2}^W = \left\| \frac{\partial \tilde{H}}{\partial Z} \times W_Z \right\|_2^2 \tag{4.21}$$

Démonstration :
La définition 4.1 appliquée à l'équation 3.16 définissant Z donne

$$\frac{\partial}{\partial Z} = \begin{pmatrix} -\frac{\partial}{\partial J} & \frac{\partial}{\partial M} & \frac{\partial}{\partial N} \\ \frac{\partial}{\partial K} & \frac{\partial}{\partial P} & \frac{\partial}{\partial Q} \\ \frac{\partial}{\partial L} & \frac{\partial}{\partial R} & \frac{\partial}{\partial S} \end{pmatrix} \tag{4.22}$$

En appliquant alors la proposition 4.3, on démontre alors l'équivalence entre (4.20) et (4.21). ■

4.2.2.2 Expression de la mesure dans le cas MIMO

Une première manière d'étendre la mesure de sensibilité de la fonction de transfert pour la forme implicite spécialisée (équation (4.21)) au cas MIMO est d'appliquer cette mesure sur chacun des transferts SISO et d'utiliser la proposition 4.3.
Nous proposons dans ce qui suit une voie alternative, qui nécessite certes un effort particulier sur le champs des manipulations matricielles (dérivée d'une fonction matricielle par une matrice, produit tensoriel, produit $\circledast$, etc.) mais possède l'avantage de conduire au final à des expressions compactes et à des calculs optimisés : l'impact global de chaque coefficient sur la fonction de transfert MIMO sera alors considéré.

Nous utiliserons alors la définition de la dérivée matricielle d'une matrice suivante :

Définition 4.6 (Dérivation matricielle d'une matrice)
Soit $X \in \mathbb{R}^{m\times n}$ et $f : \mathbb{R}^{m\times n} \to \mathbb{C}^{p\times l}$. On suppose que chaque composante de f est différentiable selon tous les éléments de X.
La dérivation matricielle de f par rapport à X est une matrice de $\mathbb{C}^{mp\times nl}$ partitionnée en $m\times n$ blocs de matrices de $\mathbb{C}^{p\times l}$. Chaque $(i,j)^e$ bloc est défini par

$$\frac{\partial f}{\partial X_{i,j}} \text{ à valeur dans } \mathbb{C}^{p\times l} \tag{4.23}$$

On a ainsi

$$\frac{\partial f}{\partial X} \triangleq \begin{pmatrix} \frac{\partial f}{\partial X_{1,1}} & \frac{\partial f}{\partial X_{1,2}} & \cdots & \frac{\partial f}{\partial X_{1,n}} \\ \frac{\partial f}{\partial X_{2,1}} & \frac{\partial f}{\partial X_{2,2}} & \cdots & \frac{\partial f}{\partial X_{2,n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial X_{m,1}} & \frac{\partial f}{\partial X_{m,2}} & \cdots & \frac{\partial f}{\partial X_{m,n}} \end{pmatrix} \tag{4.24}$$

Remarque : cette définition, que l'on retrouve à l'annexe A, étend la définition 4.1 avec cette fois des blocs $\frac{\partial f}{\partial X_{i,j}}$ non plus scalaires, mais matriciels de taille $p \times l$.

Enfin, on remarquera que la définition 4.5 ne peut se ré-écrire tel quel dans le cas MIMO, car les dimensions de $\frac{\partial \tilde{H}}{\partial X}$ et de W_X ne sont plus compatibles (si $X \in \mathbb{R}^{k\times l}$ et $H(z) \in \mathbb{C}^{p\times m}$, alors $\frac{\partial \tilde{H}}{\partial X}(z) \in \mathbb{C}^{pk\times ml}$). Notons alors N_X^W la norme pondérée de la sensibilité de H par rapport à X :

$$N_X^W = \left\| \begin{pmatrix} \frac{\partial \tilde{H}}{\partial X_{1,1}} W_{1,1} & \cdots & \\ \vdots & \ddots & \vdots \\ \frac{\partial \tilde{H}}{\partial X_{k,1}} W_{k,1} & \cdots & \frac{\partial \tilde{H}}{\partial X_{k,l}} W_{k,l} \end{pmatrix} \right\|_2^2 \tag{4.25}$$

On peut relier N_X^W à $\frac{\partial \tilde{H}}{\partial X}$ en écrivant

$$\frac{\partial \tilde{H}}{\partial X_{i,j}} W_{i,j} = \begin{pmatrix} \frac{\partial \tilde{H}_{1,1}}{\partial X_{i,j}} W_{i,j} & \cdots & \\ \vdots & \ddots & \vdots \\ \frac{\partial \tilde{H}_{p,1}}{\partial X_{i,j}} W_{i,j} & \cdots & \frac{\partial \tilde{H}_{p,n}}{\partial X_{i,j}} W_{i,j} \end{pmatrix} \tag{4.26}$$

$$= \frac{\partial \tilde{H}}{\partial X_{i,j}} \times (W_{i,j} \otimes \mathscr{E}_{p,m}) \tag{4.27}$$

où $\mathscr{E}_{p,m}$ est la matrice de $\mathbb{R}^{p \times m}$ dont tous les éléments valent 1, et $\otimes$ le produit de Kronecker (cf. définition A.1).
Ainsi, on obtient le résultat

$$N_X^W = \left\| \frac{\partial \tilde{H}}{\partial X} \times (W \otimes \mathscr{E}_{p,m}) \right\|_2^2 \tag{4.28}$$

et

$$M_{L_2}^W = \sum_{X \in \{J,K,L,M,N,P,Q,R,S\}} \left\| \frac{\partial \tilde{H}}{\partial X} \times (W_X \otimes \mathscr{E}_{p,m}) \right\|_2^2 \tag{4.29}$$

$$= \left\| \frac{\partial \tilde{H}}{\partial Z} \times (W_Z \otimes \mathscr{E}_{p,m}) \right\|_2^2 \tag{4.30}$$

Une autre façon d'exprimer cette mesure de sensibilité en MIMO est de considérer directement la norme de la sensibilité de la fonction de transfert vis-à-vis de chacun des coefficients.

Définition 4.7 (Cartographie de sensibilité)
Soit $X \in \mathbb{R}^{k \times l}$ et $\tilde{H}$ une fonction de transfert MIMO de dimension $p \times m$. On note $\frac{\delta \tilde{H}}{\delta X}$ la matrice de $\mathbb{R}^{k \times l}$ définie par

$$\left(\frac{\delta \tilde{H}}{\delta X} \right)_{i,j} \triangleq \left\| \frac{\partial \tilde{H}}{\partial X_{i,j}} \right\|_2 \tag{4.31}$$

Cette matrice est appelée ici cartographie de sensibilité *de H vis-à-vis de X.*

Cette matrice nous informe distinctement sur la sensibilité globale de H vis-à-vis de chaque coefficient de X.

L'analyse de cette matrice sera commentée au paragraphe 4.2.3 car elle permet de déterminer quels sont les coefficients ayant un impact particulier dans la dégradation (fort de cette information, il sera possible d'en tenir compte en codant de manière plus précise les coefficients critiques).
Notons le résultat suivant

Proposition 4.4

$$\left\| \frac{\delta \tilde{H}}{\delta X} \right\|_F = \left\| \frac{\partial \tilde{H}}{\partial X} \right\|_2 \tag{4.32}$$

Démonstration :
Ce résultat découle directement de la définition 4.7 et de la proposition 4.3.

■

Ces résultats intermédiaires nous permettent désormais de définir avec rigueur et précision ce que nous appellerons *mesure de sensibilité pondérée* [40] :

Définition 4.8 (Mesure de sensibilité pondérée)
On considère une réalisation $\mathcal{R} := (J, K, L, M, N, P, Q, R, S)$ et les matrices de pondération W_J, W_K, W_L, W_M, W_N, W_P, W_Q, W_R et W_S associées. La mesure de sensibilité pondérée de la matrice de transfert H vis-à-vis des coefficients de $\mathcal{R}$, est définie par

$$M_{L_2}^W \triangleq \sum_{X \in \{J,K,L,M,N,P,Q,R,S\}} \left\| \frac{\delta \tilde{H}}{\delta X} \times W_X \right\|_F^2 \tag{4.33}$$

$$= \left\| \frac{\delta \tilde{H}}{\delta Z} \times W_Z \right\|_F^2 \tag{4.34}$$

Remarque : les pondérations permettent notamment de tenir compte du fait que certains coefficients ne seront pas modifiés par l'action de quantification.

4.2.2.3 Calcul de la sensibilité

Il nous faut maintenant donner les expressions des sensibilités (ou matrices de sensibilité) vis-à-vis de J, K, L, M, N, P, Q, R, S, ou encore de Z.
Une définition et une proposition, permettant de faciliter les expressions de dérivation matricielle, sont à ce stade nécessaires.

Définition 4.9
Soient $A \in \mathbb{C}^{m \times p}$ et $B \in \mathbb{C}^{l \times n}$.
On définit l'opérateur $\circledast$ par

$$A \circledast B \triangleq \mathrm{Vec}(A).\left(\mathrm{Vec}\left(B^\top\right)\right)^\top \tag{4.35}$$

où Vec *est l'opérateur classique qui transforme une matrice en vecteur colonne.*
$A \circledast B$ *a pour dimension* $mp \times ln$.

Cet opérateur *original* va permettre de simplifier l'écriture des calculs grâce au lemme suivant :

Lemme 4.5
On considère G *et* H *deux matrices de* $\mathbb{C}^{m\times p}$ *et* $\mathbb{C}^{l\times n}$ *et* $X \in \mathbb{R}^{p\times l}$. G *et* H *sont supposés indépendants de* X. *On a alors*

$$\frac{\partial(GXH)}{\partial X} = G \circledast H \tag{4.36}$$

$$\frac{\partial\left(GX^{-1}H\right)}{\partial X} = \left(GX^{-1}\right) \circledast \left(X^{-1}H\right) \tag{4.37}$$

Ces équations sont également valables lorsque G *et* H *sont des matrices de transfert.*

Démonstration :
La démonstration est donnée par les propositions A.6 et A.7, et repose sur l'équation

$$G \circledast H = (I_p \otimes G)\frac{\partial X}{\partial X}(I_l \otimes H) \tag{4.38}$$

■

Basé sur ce lemme, le théorème suivant nous donne l'expression de $\frac{\partial \tilde{H}}{\partial Z}$:

Théorème 4.6 (Calcul des sensibilités)
La sensibilité de la fonction de transfert H, *mise en œuvre par la réalisation* $\mathcal{R} := (J, K, L, M, N, P, Q, R, S)$, *a pour expression*

$$\frac{\partial H}{\partial Z} = \begin{pmatrix} H_3 & H_1 & I_p \end{pmatrix} \circledast \begin{pmatrix} H_4 \\ H_2 \\ I_m \end{pmatrix} \tag{4.39}$$

$$\frac{\partial D}{\partial Z} = \begin{pmatrix} LJ^{-1} & 0 & I_p \end{pmatrix} \circledast \begin{pmatrix} J^{-1}N \\ 0 \\ I_m \end{pmatrix} \tag{4.40}$$

avec

$$H_1 : z \mapsto C(zI_n - A)^{-1} \tag{4.41}$$

$$H_2 : z \mapsto (zI_n - A)^{-1}B \tag{4.42}$$

$$H_3 : z \mapsto C(zI_n - A)^{-1}KJ^{-1} + LJ^{-1} \tag{4.43}$$

$$H_4 : z \mapsto J^{-1}M(zI_n - A)^{-1}B + J^{-1}N \tag{4.44}$$

Les fonctions de transfert H_1 *à* H_4 *sont respectivement à valeur dans :* $\mathbb{C}^{p\times n}$, $\mathbb{C}^{n\times m}$, $\mathbb{C}^{p\times l}$ *et* $\mathbb{C}^{l\times m}$.

Démonstration :
La démonstration est donnée en annexe B ■

Remarque 1 : Notons que l'on retrouve les résultats de M. Gevers dans le cas particulier d'une structuration en q (sensibilités S_B et S_C données à l'équation (4.8)). Remarque 2 : Les sensibilités : H_1 et H_2 proviennent de la contribution de l'état $X(k)$ dans H et les sensibilités H_3 et H_4 de la contribution des variables intermédiaires $T(k)$ dans H.
Remarque 2 : l'intérêt de l'introduction de la matrice Z (équation (3.16)) apparaît ici : elle permet d'exprimer les sensibilités par rapport à toutes les matrices J à S de la réalisation sous une forme compacte. Le signe "-" qui apparaît dans l'expression de Z (équation (3.16)) permet d'écrire les équations (4.39) et (4.40) sous forme compacte, comme *produit* (avec l'opérateur $\circledast$), de deux termes. Il se justifie car, contrairement aux autres termes, seul l'inverse de J apparaît dans les calculs, et la dérivée de J^{-1} par rapport à J donne lieu à ce signe "-".

Dans le cas SISO, $H_1(z)$ et $H_3(z)$ sont des vecteurs lignes, et $H_2(z)$ et $H_4(z)$ des vecteurs colonnes. On a donc

$$\frac{\partial H}{\partial Z} = \begin{pmatrix} H_3^\top \\ H_1^\top \\ 1 \end{pmatrix} \begin{pmatrix} H_4^\top & H_2^\top & 1 \end{pmatrix} \tag{4.45}$$

$$\frac{\partial D}{\partial Z} = \begin{pmatrix} J^{-\top} L^\top \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} N^\top J^{-\top} & 0 & 1 \end{pmatrix} \tag{4.46}$$

Enfin, on peut calculer les matrices de sensibilité, et la mesure M_L^W, grâce à la proposition suivante :

Proposition 4.7
Soit une matrice X et trois fonctions de transfert G, A et B telles que

$$\frac{\partial G}{\partial X} = A \circledast B \tag{4.47}$$

Alors, la matrice de sensibilité de $\frac{\delta G}{\delta X}$ se calcule par

$$\left(\frac{\delta G}{\delta X}\right)_{i,j} = \|A_{\bullet,i} B_{j,\bullet}\|_2 \tag{4.48}$$

où $A_{\bullet,i}$ et $B_{j,\bullet}$ sont respectivement la i^e ligne de A et la j^e colonne de B.

Démonstration :

$$\left(\frac{\delta G}{\delta X}\right)_{i,j} = \left\| \frac{\partial G}{\partial X_{i,j}} \right\|_2 \tag{4.49}$$

$$= \sqrt{\sum_{k,l} \left\| \frac{\partial G_{k,l}}{\partial X_{i,j}} \right\|_2^2} \tag{4.50}$$

$$= \sqrt{\sum_{k,l} \|A_{k,i} B_{j,l}\|_2^2} \tag{4.51}$$

$$= \|A_{\bullet,i} B_{j,\bullet}\|_2 \tag{4.52}$$

■

Remarque : ce résultat est essentiel pour mettre calculer la matrice de sensibilité grâce au théorème 4.6.

4.2.3 Exemple

Pour évaluer numériquement, sur quelques réalisations, la mesure $M_{L_2}^W$, on propose l'exemple suivant :

$$H(z) = \frac{0.079306721z^2 + 0.023016947z + 0.0231752363}{z^3 - 1.974861148z^2 + 1.556161235z - 0.4537681314} \tag{4.53}$$

Il s'agit d'un filtre passe-bas choisi comme exemple par SY. Hwang [46, 47] et M. Gevers [31].
Nous utiliserons les caractères gras pour exprimer les coefficients significatifs qui risquent d'être tronqués lors du processus de quantification (la matrice de pondération W_Z sera représentée par ce biais : un paramètre $Z_{i,j}$ apparaîtra en gras si $(W_Z)_{i,j} = 1$). En plus de la mesure $M_{L_2}^W$, nous exhiberons aussi la matrice de sensibilité $\frac{\delta \tilde{H}}{\delta Z}$. Bien que seules les sensibilités en gras soient considérées pour le calcul de $M_{L_2}^W$ (via le produit direct $\times W_Z$), nous avons choisi de les donner toutes afin de noter l'impact potentiel d'une erreur sur chaque coefficient.
Enfin, dans tous les exemples suivants, et bien que ce ne soit pas toujours suffisant pour les reconstituer numériquement, les résultats ne seront affichés qu'avec 5 chiffres significatifs (avec arrondi au plus juste) afin de ne pas alourdir la présentation, bien que des mesures de sensibilités montreront, dans certains cas, des valeurs assez sensibles aux quantifications des coefficients. Tous les calculs ont été réalisés sous Matlab, en *flottant double précision*.

Les réalisations comparées sont :

– Forme directe I : $M_{L_2}^W = 93.714$

$$Z_1 = \left(\begin{array}{c|cccccc|c} -1 & \mathbf{0.07931} & \mathbf{0.02302} & \mathbf{0.02318} & \mathbf{1.97486} & \mathbf{-1.55616} & \mathbf{0.45377} & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right)$$

$$\left.\frac{\delta \tilde{H}}{\delta Z}\right|_{Z_1} = \left(\begin{array}{c|cccccc|c} 3.76512 & \mathbf{4.13062} & \mathbf{4.13062} & \mathbf{4.13062} & \mathbf{3.76512} & \mathbf{3.76512} & \mathbf{3.76512} & 4.00774 \\ \hline 0.44558 & 0.48067 & 0.48067 & 0.48067 & 0.44558 & 0.44558 & 0.44558 & 0.48067 \\ 0.16970 & 0.18462 & 0.18462 & 0.18462 & 0.16970 & 0.16970 & 0.16970 & 0.18462 \\ 0.08726 & 0.09573 & 0.09573 & 0.09573 & 0.08726 & 0.08726 & 0.08726 & 0.09573 \\ 3.48331 & 4.00774 & 4.00774 & 4.00774 & 3.48331 & 3.48331 & 3.48331 & 4.00774 \\ 4.37580 & 4.87926 & 4.87926 & 4.87926 & 4.37580 & 4.37580 & 4.37580 & 4.87926 \\ 1.70849 & 1.87434 & 1.87434 & 1.87434 & 1.70849 & 1.70849 & 1.70849 & 1.87434 \\ \hline 0.48067 & 1 & 1 & 1 & 0.48067 & 0.48067 & 0.48067 & 0 \end{array}\right)$$

Nous avons considéré ici que $a_0 = 1$ et d_0 étaient implémentés exactement, alors que la sensibilité par rapport à ces deux valeurs est assez élevée (3.76512 et 4.00774).

– Forme directe II : $M_{L_2}^W = 93.714$

$$Z_2 = \left(\begin{array}{ccc|c} & & & \\ \hline \mathbf{1.97486} & \mathbf{-1.55616} & \mathbf{0.45377} & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \hline \mathbf{0.07931} & \mathbf{0.02302} & \mathbf{0.02318} & 0 \end{array}\right)$$

$$\left.\frac{\delta \tilde{H}}{\delta Z}\right|_{Z_2} = \left(\begin{array}{ccc|c} & & & \\ \hline \mathbf{3.76512} & \mathbf{3.76512} & \mathbf{3.76512} & 0.48067 \\ 4.30815 & 4.30815 & 4.30815 & 0.55777 \\ 1.72459 & 1.72459 & 1.72459 & 0.21934 \\ \hline \mathbf{4.13062} & \mathbf{4.13062} & \mathbf{4.13062} & 0 \end{array}\right)$$

Remarque 1 : on peut comparer ce résultat avec celui obtenu avec une mesure de sensibilité non pondérée, telle que celle proposée par M. Gevers : la mesure pondérée est plus proche de la réalité et "diabolise" un peu moins la forme compagne.

Remarque 2 : nous retrouvons ici la même valeur de mesure de sensibilité (avec une même sensibilité pour chaque coefficient). Ceci est logique puisque la paramétrisation est inchangée. En revanche, le coût logiciel diffère et l'on peut suggérer, contrairement à ce qui est fait parfois chez PSA, de préférer la forme directe II à la forme directe I.

– Forme équilibrée : $M_{L_2}^W = 7.9447$

$$Z_3 = \left(\begin{array}{ccc|c} & & & \\ \hline \mathbf{0.82363} & \mathbf{-0.39994} & \mathbf{0.01645} & \mathbf{-0.44240} \\ \mathbf{0.39994} & \mathbf{0.59351} & \mathbf{0.34245} & \mathbf{0.37991} \\ \mathbf{0.01645} & \mathbf{-0.34245} & \mathbf{0.55772} & \mathbf{0.16710} \\ \hline \mathbf{-0.44240} & \mathbf{-0.37991} & \mathbf{0.16710} & \mathbf{0} \end{array}\right)$$

$$\left.\frac{\delta \tilde{H}}{\delta Z}\right|_{Z_3} = \left(\begin{array}{c|ccc|c} & & & & \\ \hline & \mathbf{1.47561} & \mathbf{0.92033} & \mathbf{0.46038} & \mathbf{0.91225} \\ & \mathbf{0.92033} & \mathbf{0.78475} & \mathbf{0.32426} & \mathbf{0.67051} \\ & \mathbf{0.46038} & \mathbf{0.32426} & \mathbf{0.15772} & \mathbf{0.34262} \\ \hline \mathbf{0.91225} & \mathbf{0.67051} & \mathbf{0.34262} & & \mathbf{0} \end{array}\right)$$

La forme équilibrée (cf. définition 2.1) possède une mesure de sensibilité nettement plus faible que les deux formes directes, alors même qu'elle utilise davantage de coefficients non triviaux. Ceci est du au fait que la sensibilité de chaque coefficient est bien plus petite. On notera que certains coefficients (par exemple $P_{1,1}$)) possèdent une sensibilité plus grande que d'autres et que cela pourrait être pris en compte dans le choix de leur représentation.

Pour les réalisations en δ, on remarquera que le paramètre Δ intervient bien dans les calculs (cf. chapitre 3.2.3.1). Celui-ci peut-être quantifié et donc modifié, et il importe de pouvoir tenir compte de la sensibilité de la fonction de transfert vis-à-vis de ce paramètre, ce qui n'était pas fait dans les travaux antérieurs. Ce paramètre étant délibérément fixé, il est possible et intéressant, lors de l'étape de synthèse, de le choisir implémentable exactement, et si possible avec une valeur facilitant les calculs, comme par exemple une puissance (négative) de 2.
On trouvera dans [31] la démonstration de la meilleure sensibilité des réalisations en δ (si on ne tient pas compte de la sensibilité à Δ) sur les réalisations en q classiques dès que $\Delta < 1$.
On choisit dans cet exemple $\Delta = 2^{-1}$

– Forme directe II avec l'opérateur δ : $M_{L_2}^W = 9.0211$

$$Z_4 = \left(\begin{array}{ccc|ccc|c} -1 & 0 & 0 & \mathbf{-2.05028} & \mathbf{-2.42576} & \mathbf{-1.02026} & 1 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ \hline \mathbf{0.5} & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & \mathbf{0.5} & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \mathbf{0.5} & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & \mathbf{0.15861} & \mathbf{0.72652} & \mathbf{1.00399} & 0 \end{array}\right)$$

$$\left.\frac{\delta \tilde{H}}{\delta Z}\right|_{Z_4} = \left(\begin{array}{ccc|ccc|c} 0.60659 & 0.47621 & 0.41221 & \mathbf{0.47621} & \mathbf{0.41221} & \mathbf{0.47064} & 0.48067 \\ 1.33015 & 1.00683 & 0.84800 & 1.00683 & 0.84800 & 0.93048 & 0.97010 \\ 1.12237 & 0.79055 & 0.65507 & 0.79055 & 0.65507 & 0.75010 & 0.82481 \\ \hline \mathbf{1.21319} & 0.95241 & 0.82443 & 0.95241 & 0.82443 & 0.94128 & 0.96134 \\ 2.66030 & \mathbf{2.01366} & 1.69599 & 2.01366 & 1.69599 & 1.86096 & 1.94019 \\ 2.24473 & 1.58109 & \mathbf{1.31014} & 1.58109 & 1.31014 & 1.50020 & 1.64962 \\ \hline 1.33974 & 0.78817 & 0.52147 & \mathbf{0.78817} & \mathbf{0.52147} & \mathbf{0.51633} & 0 \end{array}\right)$$

Pour cette réalisation canonique, on a pris ici en compte la sensibilité par rapport à Δ : on remarque que celle-ci est plus élevée que pour les autres coefficients. Nous obtenons tout de même une sensibilité assez basse, surtout en regard de la forme directe II classique.

– Réalisation δ depuis la forme équilibrée $M_{L_2}^W = 3.0355$

$$Z_5 = \left(\begin{array}{ccc:ccc:c} -1 & 0 & 0 & \mathbf{-0.35274} & \mathbf{-0.79989} & \mathbf{0.03289} & \mathbf{-0.88480} \\ 0 & -1 & 0 & \mathbf{0.79989} & \mathbf{-0.81298} & \mathbf{0.68491} & \mathbf{0.75983} \\ 0 & 0 & -1 & \mathbf{0.03289} & \mathbf{-0.68491} & \mathbf{-0.88456} & \mathbf{0.33420} \\ \hdashline 0.5 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0 & 1 & 0 \\ \hdashline 0 & 0 & 0 & \mathbf{-0.44240} & \mathbf{-0.37991} & \mathbf{0.16710} & \mathbf{0} \end{array}\right)$$

$$\left.\frac{\delta \tilde{H}}{\delta Z}\right|_{Z_5} = \left(\begin{array}{ccc:ccc:c} 0.67007 & 0.56833 & 0.23747 & \mathbf{0.73780} & \mathbf{0.46017} & \mathbf{0.23019} & \mathbf{0.45612} \\ 0.56833 & 0.55231 & 0.23415 & \mathbf{0.46017} & \mathbf{0.39237} & \mathbf{0.16213} & \mathbf{0.33525} \\ 0.23747 & 0.23415 & 0.10554 & \mathbf{0.23019} & \mathbf{0.16213} & \mathbf{0.07886} & \mathbf{0.17131} \\ \hdashline 1.34014 & 1.13665 & 0.47494 & 1.47561 & 0.92033 & 0.46038 & 0.91225 \\ 1.13665 & 1.10461 & 0.46830 & 0.92033 & 0.78475 & 0.32426 & 0.67051 \\ 0.47494 & 0.46830 & 0.21109 & 0.46038 & 0.32426 & 0.15772 & 0.34262 \\ \hdashline 0.62556 & 0.94057 & 0.55104 & \mathbf{0.91225} & \mathbf{0.67051} & \mathbf{0.34262} & \mathbf{0} \end{array}\right)$$

Le tableau 4.2 résume les mesures de sensibilité et coûts de calcul associées à chaque réalisation.

Réalisation	sensibilité $M_{L_2}^W$	coût de calcul
Forme directe I (Z_1)	93.714	6 × 5+
Forme directe II (Z_2)	93.714	6 × 5+
Forme équilibrée (Z_3)	7.9447	16 × 12+
Forme directe II en δ (Z_4)	9.0211	9 × 8+
Réalisation Z_3 exprimée en δ (Z_5)	3.0355	19 × 15+

TAB. 4.2 – Mesure de sensibilité $M_{L_2}^W$ et coût de calcul

En plus de la mesure de sensibilité paramétrique pondérée $M_{L_2}^W$, il est possible *a posteriori* d'évaluer la distance entre la réalisation initiale et la réalisation quantifiée.
On suppose, par exemple, que les coefficients de Z sont implémentés en virgule fixe sur 7 bits[9] (avec un format commun à tous les coefficients). Le tableau 4.3 donne le nombre de bits minimum nécessaire pour ces cinq réalisations en virgule fixe avec format unique (cf. paragraphe 3.4.3).

La figure 4.1 et le tableau 4.4 montrent, pour la fonction de transfert H, les différences entre les réalisations initiales Z_1, Z_2, Z_3 et Z_4 et quantifiées Z_1^*, Z_2^*, Z_3^* et Z_4^*.

On remarque que la quantification a un impact assez important sur H_1^* et H_2^*, assez limité sur H_3^* et très faible sur H_4^*, en accord avec la mesure *a priori* de sensibilité $M_{L_2}^W$.

[9]Nous exagérons volontairement la quantification en ne retenant que 7 bits, pour que les effets de la quantification soient très visibles sur H.

Réalisation	nb de bits minimum
Z_1	7
Z_2	7
Z_3	6
Z_4	5
Z_5	5

TAB. 4.3 – Nombre de bits minimum (virgule fixe avec format commun)

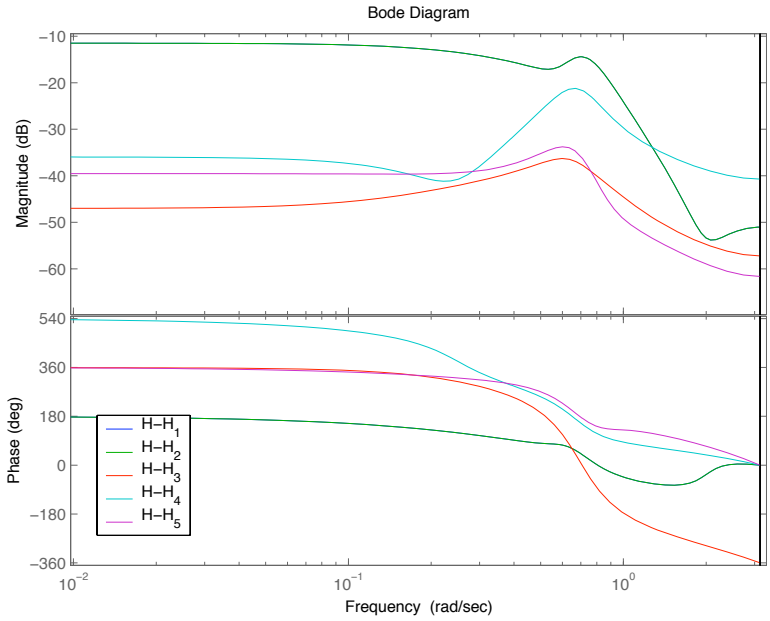


FIG. 4.1 – Différences entre la fonction de transfert de référence et les différentes réalisations

Réalisation	$\|H - H^*\|_2$	$\|H - H^*\|_\infty$
Z_1	$1.0369e{-}01$	$2.6594e{-}01$
Z_2	$1.0369e{-}01$	$2.6594e{-}01$
Z_3	$6.1514e{-}03$	$1.5315e{-}02$
Z_4	$2.9798e{-}02$	$8.6903e{-}02$
Z_5	$7.4008e{-}03$	$2.0513e{-}02$

TAB. 4.4 – Ecart entre la fonction de transfert initiale et les fonctions de transfert quantifiées

En reprenant la notation de la section 3.4 qui montre que Z est modifié en $Z + r_Z \times \Delta$ par la quantification, et en notant $\tilde{H}^*$ la fonction de transfert de la réalisation de coefficients $Z + r_Z \times \Delta$ (on a donc $\tilde{H}^* = \tilde{H}$ pour $\Delta = 0$), on peut remarquer (pour le cas SISO, mais généralisable) que[10]

$$\tilde{H}^*(z) - \tilde{H}(z) = \sum_{i,j} \Delta_{i,j} \left. \frac{\partial \tilde{H}^*(z)}{\partial \Delta_{i,j}} \right|_{\Delta=0} + o\left(\|\Delta\|_{\max}\right) \qquad \forall z \in \mathbb{C} \tag{4.54}$$

et

$$\left\| \tilde{H}^* - \tilde{H} \right\|_2 \leqslant \|\Delta\|_{\max} \left\| \left. \frac{\partial \tilde{H}^*}{\partial \Delta} \right|_{\Delta=0} \right\|_2 \tag{4.55}$$

Or, on notera que

$$\left. \frac{\partial \tilde{H}^*}{\partial \Delta} \right|_{\Delta=0} = \frac{\partial \tilde{H}}{\partial Z} \times r_Z \tag{4.56}$$

et que $\|\Delta\|_{\max}$ est majorée par une constante dépendant du format de représentation des données uniquement (cf. équation (3.96)).
On pourra écrire

$$\left\| \left. \tilde{H} \right|_Z - \left. \tilde{H} \right|_{Z + r_Z \times \Delta} \right\|_2 \leqslant \|\Delta\|_{\max} \left\| \frac{\partial \tilde{H}}{\partial Z} \times r_Z \right\|_2 \tag{4.57}$$

Dans le cas de la virgule fixe ($r_Z = W_Z$), nous retrouvons la mesure de sensibilité paramétrique pondérée telle que nous l'avons défini. L'équation (4.57) permet aussi de justifier la mesure initiale de M. Gevers et G. Li (équation (4.12)) que nous avons étendue ici à la forme implicite spécialisée et complétée avec la pondération W_Z.

Mais l'équation (4.57) permet aussi de construire une autre mesure de sensibilité pondérée, mieux adaptée à l'étude de la sensibilité paramétrique lorsque l'implémentation se fait en virgule flottante :

$$M^W_{L_2,\text{flottant}} \triangleq \left\| \frac{\delta \tilde{H}}{\delta Z} \times W_Z \times \eta_Z \right\|_F^2 \tag{4.58}$$

4.3 Sensibilité en boucle fermée

4.3.1 Contexte

Dans le cadre de la régulation ou de l'asservissement, la loi mise en œuvre intéragit avec le système qu'elle cherche à contrôler. Il est important, alors,

[10] Le symbole o est appelé symbole de Landau (comme le symbole $\mathcal{O}$). L'écriture $f(x) = o\left(g(x)\right)$ pour $x \to a$ signifie que f est négligeable devant g au voisinage de a, c.-à-d. que $\lim_{x \to a} \frac{f}{g} = 0$.

d'évaluer l'impact global que peut avoir la quantification sur le système bouclé.
On étudiera donc, dans cette partie, la sensibilité de la fonction de transfert de la boucle fermée vis-à-vis des coefficients mis en jeu dans la réalisation numérique, que l'on supposera, naturellement, exprimée sous la forme implicite spécialisée.
On considère donc un système $\mathcal{P}$, de dimension n_p, avec p entrées et m_1 sorties, régi par l'équation d'état

$$\left\{ \begin{array}{rcl} X_p(k+1) & = & A_p X_p(k) + B_p U(k) \\ Y(k) & = & C_p X_p(k) \end{array} \right. \tag{4.59}$$

Ce système est contrôlé par un régulateur $\mathcal{C}$ de réalisation $\mathcal{R} := (J,\ K,\ L,\ M,\ N,\ P,\ Q,\ R,\ S)$, selon la figure 4.2, de dimension (l, m, n, p), avec $m = m_1 + m_2$.
Si nécessaire, on peut découper les matrices N, Q et S en N_1, N_2, Q_1, Q_2, S_1

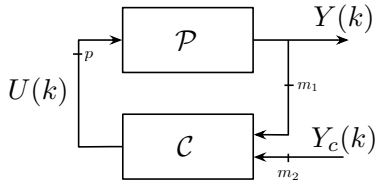


FIG. 4.2 – Le système bouclé

et S_2 si l'on veut différencier les entrées Y et Y_c (la consigne) du régulateur à deux degrés de libertés.
On retrouve comme cas particulier le schéma classiquement utilisé dans la plupart des articles sur l'implémentation en précision finie ([52], [118], [105] par exemple, cf. figure 4.3), en posant $N_2 = 0$, $Q_2 = 0$ et $S_2 = I$. Nous avons choisi d'inclure l'addition de la consigne avec la sortie du régulateur *dans* le régulateur afin, éventuellement, d'en tenir compte dans l'effet de la quantification, et pour obtenir une forme aussi générale que possible.

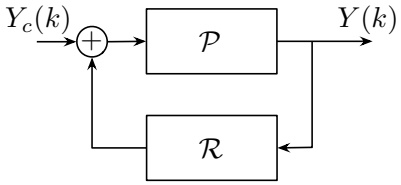


FIG. 4.3 – Le système bouclé vu dans [52, 118, 105]

Le système bouclé équivalent a pour fonction de transfert $\bar{H}$:

$$\bar{H}(z) \triangleq \bar{C}\left(zI_{n+n_p} - \bar{A}\right)^{-1}\bar{B} \qquad \forall z \in \mathbb{C} \tag{4.60}$$

avec $\bar{A} \in \mathbb{R}^{(n+n_p)\times(n+n_p)}$, $\bar{B} \in \mathbb{R}^{(n+n_p)\times m_2}$, $\bar{C} \in \mathbb{R}^{m_1\times(n+n_p)}$, $\tilde{C}_p \in \mathbb{R}^{m\times n_p}$ et $\tilde{I} \in \mathbb{R}^{m\times m_2}$ définies par

$$\bar{A} \triangleq \begin{pmatrix} A_p + B_p D\tilde{C}_p & B_p C \\ B\tilde{C}_p & A \end{pmatrix} \quad \bar{B} \triangleq \begin{pmatrix} B_p D\tilde{I} \\ B\tilde{I} \end{pmatrix} \quad \bar{C} \triangleq \begin{pmatrix} C_p & 0 \end{pmatrix} \tag{4.61}$$

avec

$$\tilde{C}_p \triangleq \begin{pmatrix} C_p \\ 0 \end{pmatrix} \quad \tilde{I} \triangleq \begin{pmatrix} 0 \\ I_{m_2} \end{pmatrix} \tag{4.62}$$

(les matrices A, B, C, D sont celles associées à $\mathcal{R}$ et définies aux équations (3.8) à (3.11)).

4.3.2 Expression de la sensibilité

La mesure de sensibilité de la fonction de transfert en boucle fermée s'écrit, par analogie avec la définition 4.8 :

Définition 4.10 (Mesure de sensibilité en boucle fermée)
On considère le système en boucle fermée défini à la figure 4.2. La mesure de sensibilité de la fonction de transfert en boucle fermée est définie par

$$\bar{M}_{L_2}^{W} \triangleq \left\| \frac{\delta \bar{H}}{\delta Z} \times W_Z \right\|_F^2 \tag{4.63}$$

ou encore

$$\bar{M}_{L_2}^{W} \triangleq \left\| \frac{\delta \bar{H}}{\delta Z} \times W_Z \times \eta_Z \right\|_F^2 \tag{4.64}$$

si les coefficients de Z sont représentés en virgule flottante.

Le théorème suivant, original, donne l'expression de la sensibilité de $\bar{H}$:

Théorème 4.8
La sensibilité de la fonction de transfert en boucle fermée est donnée par

$$\frac{\partial \bar{H}}{\partial Z} = \left(\bar{H}_1 \bar{M}_1\right) \circledast \left(\bar{M}_2 \bar{H}_2 + \bar{M}_3\right) \tag{4.65}$$

avec $\bar{H}_1$ et $\bar{H}_2$ deux matrices de transfert (ainsi dénommées par analogie à H_1 et H_2 définies au théorème 4.6) définies par

$$\bar{H}_1 : \quad z \mapsto \quad \bar{C}\left(zI_{n+n_p} - \bar{A}\right)^{-1} \tag{4.66}$$

$$\bar{H}_2 : \quad z \mapsto \quad \left(zI_{n+n_p} - \bar{A}\right)^{-1}\bar{B} \tag{4.67}$$

et $\bar{M}_1 \in \mathbb{R}^{(n+n_p)\times(l+n+p)}$, $\bar{M}_2 \in \mathbb{R}^{(l+n+m)\times(n+n_p)}$ *et* $\bar{M}_3 \in \mathbb{R}^{(l+n+m)\times(m_2)}$ *définies par*

$$\bar{M}_1 \triangleq \begin{pmatrix} B_p L J^{-1} & 0 & B_p \\ K J^{-1} & I_n & 0 \end{pmatrix} \tag{4.68}$$

$$\bar{M}_2 \triangleq \begin{pmatrix} J^{-1} N \tilde{C}_p & J^{-1} M \\ 0 & I_n \\ \tilde{C}_p & 0 \end{pmatrix} \tag{4.69}$$

$$\bar{M}_3 \triangleq \begin{pmatrix} J^{-1} N \tilde{I} \\ 0 \\ \tilde{I} \end{pmatrix} \tag{4.70}$$

Démonstration :

$$\begin{aligned} \frac{\partial \bar{H}}{\partial Z}(z) &= \left(I_{l+n+p} \otimes \bar{C}\right) \frac{\partial \left(zI - \bar{A}\right)^{-1}}{\partial Z} \left(I_{l+n+m} \otimes \bar{B}\right) \\ &\quad + \left(I_{l+n+p} \otimes \bar{C}\left(zI - \bar{A}\right)^{-1}\right) \frac{\partial \bar{B}}{\partial Z} \\ &= \left(I_{l+n+p} \otimes \bar{H}_1(z)\right) \left(\frac{\partial \bar{A}}{\partial Z}\left(I_{l+n+m} \otimes \bar{H}_2(z)\right) + \frac{\partial \bar{B}}{\partial Z}\right) \end{aligned} \tag{4.71}$$

Enfin, la proposition B.3, présente à l'annexe B, permet de montrer que

$$\frac{\partial \bar{A}}{\partial Z} = \bar{M}_1 \circledast \bar{M}_2 \qquad \text{et} \qquad \frac{\partial \bar{B}}{\partial Z} = \bar{M}_1 \circledast \bar{M}_3 \tag{4.72}$$

■

Comme pour la mesure $M_{L_2}^W$, $\frac{\delta \tilde{H}}{\delta Z}$ s'obtient grâce à la proposition 4.7 et au théorème 4.8.

4.4 Mesure de stabilité

4.4.1 Historique

En complément des mesures de sensibilité de fonction de transfert initiées par V. Tavşanoğlu et L. Thiele [100], d'autres mesures d'évaluation de la dégradation due à la représentation en précision finie des coefficients ont été développées : le déplacement induit des pôles de la fonction de transfert est tout particulièrement étudié. Ceci est dû au fait que les pôles sont non seulement des indicateurs structurants du système mais caractérisent également la stabilité.

Dans [95], une première mesure de sensibilité des valeurs propres $(\lambda_k)_{1\leqslant k\leqslant n}$ de A par rapport aux coefficients de A est énoncée :

$$\Psi_p \triangleq \sum_{k=1}^{n} \left\| \frac{\partial \lambda_k}{\partial A} \right\|_F^2 \tag{4.73}$$

La sensibilité des pôles d'un système est reprise ensuite, dans des mesures relatives à la stabilité d'un système en boucle fermée [105] : en effet, si l'on considère un système contrôlé par un régulateur, la position des pôles du système en boucle fermée vis-à-vis du cercle unité nous indique la stabilité du système rebouclé.
Une modification des paramètres de la réalisation, par exemple par quantification, entraîne alors un déplacement des pôles, parfois au delà de la stabilité. L'exemple numérique exhibé dans [84, 106, 118] est un régulateur *fragile* [57] très sensible à la quantification : pour une réalisation sous forme canonique, une représentation (virgule fixe) avec plus de 20 bits est nécessaire pour que le système bouclé reste stable.
Cet exemple montre donc la nécessité de disposer d'une mesure permettant d'exprimer la sensibilité des pôles en boucle fermée vis-à-vis des coefficients en jeu ainsi qu'une évaluation du nombre de bits minimum nécessaires pour coder/représenter les coefficients d'une réalisation sans perdre la stabilité.

4.4.2 Mesure de stabilité

On considère dans cette section le même système bouclé qu'au paragraphe 4.3 : un système $\mathcal{P}$ décrit par les matrices systèmes (A_p, B_p, C_p) et régulé par le contrôleur de réalisation $\mathcal{R} := (J, K, L, M, N, P, Q, R, S)$ (figure 4.4).

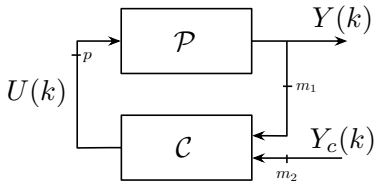


FIG. 4.4 – Le système bouclé

On note de nouveau $(\bar{A}, \bar{B}, \bar{C})$ les matrices d'état du système bouclé, et $(\lambda_k)_{1\leqslant k\leqslant n}$ les valeurs propres de $\bar{A}$ (les pôles du système bouclé).

Nous avons vu au chapitre 3.4.4 que les coefficient de Z sont perturbés et modifiés en $Z + r_Z \times \Delta$.

La mesure de stabilité, qui indique la quantification maximum qu'il est possible d'accepter sans perdre la stabilité du système en boucle fermée, est définie par :

Définition 4.11

$$\mu_0(Z) \triangleq \inf\{\|\Delta\|_{\max} / \textit{ le système paramétré par } Z + r_Z \times \Delta \textit{ soit instable}\}$$

Si on note $\lambda_k(Z)$ les valeurs propres de $\bar{A}$ (fonction de Z), alors

$$\mu_0(Z) = \inf\{\|\Delta\|_{\max} / \ |\lambda_k(Z + r_Z \times \Delta)| > 1, \quad \forall\ 1 \leqslant k \leqslant n + n_p\} \tag{4.74}$$

On remarquera que si Δ est tel que $\|\Delta\|_{\max} \leqslant \mu_0(Z)$, alors $\bar{A}(Z + r_Z \times \Delta)$ est stable[11].
Cette mesure n'est malheureusement pas facilement évaluable (voir [115]), et il nous faut donc trouver une mesure approchante. On se propose, en supposant que l'erreur paramétrique amenée par la quantification est faible, d'écrire pour tout k le développement limité au 1er ordre :

$$|\lambda_k(Z + r_Z \times \Delta)| - |\lambda_k(Z)| = \sum_{i,j} \Delta_{i,j} \left.\frac{\partial\,|\lambda_k|}{\partial \Delta_{i,j}}\right|_{\Delta=0} + o\left(\|\Delta\|_{\max}\right) \tag{4.75}$$

avec, comme au paragraphe 4.2.3

$$\left.\frac{\partial\,|\lambda_k|}{\partial \Delta}\right|_{\Delta=0} = \frac{\partial\,|\lambda_k|}{\partial Z} \times r_Z \tag{4.76}$$

On peut alors majorer la perturbation sur le k^{e} pôle par

$$\big|\,|\lambda_k(Z + r_Z \times \Delta)| - |\lambda_k(Z)|\big| \leqslant \|\Delta\|_{\max} \left\| \frac{\partial\,|\lambda_k|}{\partial Z} \times r_Z \right\|_S \tag{4.77}$$

où $\|.\|_S$ est une norme 1 matricielle[12] définie par :

$$\|X\|_S \triangleq \sum_{i,j} |X_{i,j}| \tag{4.78}$$

On introduit alors la mesure suivante

$$\mu_1(Z) \triangleq \min_{1 \leqslant k \leqslant n+n_p} \frac{1 - |\lambda_k|}{\left\| \frac{\partial |\lambda_k|}{\partial Z} \times r_Z \right\|_S} \tag{4.79}$$

[11] *c.a.d* le système bouclé paramètré par $Z + r_Z \times \Delta$ est stable.
[12] Au même titre que la norme de Frobenius est une norme 2 matricielle.

qui permet d'écrire

$$\begin{aligned} \|\Delta\|_{\max} < \mu_1(Z) &\Rightarrow \big|\,|\lambda_k(Z + r_Z \times \Delta)| - |\lambda_k(Z)|\big| < 1 - |\lambda_k| && (4.80)\\ &\Rightarrow |\lambda_k(Z + r_Z \times \Delta)| < 1 && (4.81)\\ &\Rightarrow \bar{A}(Z + r_Z \times \Delta) \text{ est stable} && (4.82) \end{aligned}$$

Autrement dit, pour une réalisation $\mathcal{R} := (Z)$, le système en boucle fermée peut tolérer une perturbation paramétrique Δ due à la précision finie, dont la norme $\|.\|_{\max}$ est inférieure à $\mu_1(Z)$. Ainsi, plus $\mu_1(Z)$ est élevé et plus le système en boucle fermée peut tolérer des erreurs paramétriques élevées sans devenir instable. Cette mesure, relative à la stabilité est donc bien un indicateur du degré de résilience d'une réalisation vis-à-vis de l'opération de quantification.
Cette mesure a été développée successivement et enrichie par G. Li, J. Wu, S. Chen, R. Istepanian, J. Whidborne, J. Chu, etc. et étendue ici à la forme implicite spécialisée (les calculs sont écrits sous une forme compacte grâce à la matrice Z cf. [41]). Plusieurs formes de cette mesure ont été successivement exposées, les premières ne prennent pas en compte la pondération W_Z présente ici dans le terme r_Z, d'autres évaluent la sensibilité des valeurs propres $\frac{\partial \lambda_k}{\partial Z}$ au lieu de la sensibilité du module des valeurs propres $\frac{\partial |\lambda_k|}{\partial Z}$, d'autres encore utilisent l'inégalité de Cauchy pour majorer l'écart sur le k^{e} pôle, ce qui induit une mesure μ_2 définie par

$$\mu_2(Z) \triangleq \min_{1 \leqslant k \leqslant n+n_p} \frac{1 - |\lambda_k|}{\|W_Z\|_F \left\| \frac{\partial |\lambda_k|}{\partial Z} \times r_Z \right\|_F} \tag{4.83}$$

La proposition 4.11 montrera que

$$\left| \frac{\partial |\lambda_k|}{\partial Z_{i,j}} \right| \leqslant \left| \frac{\partial \lambda_k}{\partial Z_{i,j}} \right| \tag{4.84}$$

et que, donc, les mesures utilisant la sensibilité du module des valeurs propres sont moins conservatives que celles sans le module. [119] montre aussi que ces mesures peuvent nécessiter un effort de calcul moindre.
De ces deux mesures utilisées, on définit ici la mesure de stabilité suivante :

Définition 4.12 (Mesure de stabilité)
On considère le système en boucle fermée défini à la figure 4.2. La mesure de stabilité du système en boucle fermée est définie par

$$\mu(Z) = \max\left(\mu_1(Z), \mu_2(Z)\right) \tag{4.85}$$

où

$$\mu_2(Z) \triangleq \min_{1\leqslant k\leqslant n+n_p} \frac{1-|\lambda_k|}{\|W_Z\|_F \left\| \frac{\partial|\lambda_k|}{\partial Z} \times r_Z \right\|_F} \tag{4.86}$$

$$\mu_1(Z) \triangleq \min_{1\leqslant k\leqslant n+n_p} \frac{1-|\lambda_k|}{\left\| \frac{\partial|\lambda_k|}{\partial Z} \times r_Z \right\|_S} \tag{4.87}$$

La section 4.4.6 relie ces mesures au nombre de bits minimum, suivant la représentation choisie, nécessaire pour assurer la stabilité.

4.4.3 Expression de la mesure de sensibilité

Le calcul de μ_1 ou μ_2 nécessite l'évaluation de $\frac{\partial|\lambda_k|}{\partial Z}$ que l'on obtient grâce aux lemmes et propositions suivantes :

Lemme 4.9
On considère une fonction $f : \mathbb{R}^{m\times n} \to \mathbb{C}$ différentiable et deux matrices $M \in \mathbb{R}^{m\times n}$ et $X \in \mathbb{R}^{p\times l}$.
On considère aussi M_0, M_1 et M_2 des matrices constantes (vis-à-vis de X) de dimensions appropriées. On a les résultats suivants

- *Si $M = M_0 + M_1 X M_2$, alors*

$$\frac{\partial f(M)}{\partial X} = M_1^\top \frac{\partial f(M)}{\partial M} M_2^\top \tag{4.88}$$

- *Si $M = M_0 + M_1 X^{-1} M_2$, alors*

$$\frac{\partial f(M)}{\partial X} = -\left(M_1 X^{-1}\right)^\top \frac{\partial f(M)}{\partial M} \left(X^{-1} M_2\right)^\top \tag{4.89}$$

Démonstration :
On trouvera la démonstration dans [69] et [52]. ■

Remarque : ce lemme est à rapprocher du lemme 4.5 permettant de calculer la sensibilité $\frac{\partial H}{\partial Z}$.

Théorème 4.10
La sensibilité du module des pôles en boucle fermée vis-à-vis de Z est donnée par

$$\frac{\partial\,|\lambda_k|}{\partial Z} = \bar{M}_1^\top \frac{\partial|\lambda_k|}{\partial \bar{A}} \bar{M}_2^\top \tag{4.90}$$

où les matrices $\bar{M}_1 \in \mathbb{R}^{(n+n_p)\times(l+n+p)}$ *et* $\bar{M}_2 \in \mathbb{R}^{(l+n+m)\times(n+n_p)}$ *sont définies par*

$$\bar{M}_1 \triangleq \begin{pmatrix} B_p L J^{-1} & 0 & B_p \\ K J^{-1} & I_n & 0 \end{pmatrix} \tag{4.91}$$

$$\bar{M}_2 \triangleq \begin{pmatrix} J^{-1} N \tilde{C}_p & J^{-1} M \\ 0 & I_n \\ \tilde{C}_p & 0 \end{pmatrix} \tag{4.92}$$

Ces matrices sont évidemment celles introduites à la proposition B.3 sur la sensibilité de la fonction de transfert en boucle fermée.

Démonstration :
On applique le lemme 4.9 sur les équations (B.16) à (B.24) (qui servent à démontrer le théorème 4.8 et dont la démonstration complète est à l'annexe B), ce qui donne les sensibilités de $|\lambda_k|$ par rapport à J, K, L, M, N, P, Q, R, S :

$$\frac{\partial |\lambda_k|}{\partial K} = \begin{pmatrix} 0 \\ I_n \end{pmatrix}^\top \frac{\partial |\lambda_k|}{\partial \bar{A}} \begin{pmatrix} J^{-1} N \tilde{C}_p & J^{-1} M \end{pmatrix}^\top \tag{4.93}$$

$$\frac{\partial |\lambda_k|}{\partial L} = \begin{pmatrix} B_p \\ 0 \end{pmatrix}^\top \frac{\partial |\lambda_k|}{\partial \bar{A}} \begin{pmatrix} J^{-1} N \tilde{C}_p & J^{-1} M \end{pmatrix}^\top \tag{4.94}$$

$$\frac{\partial |\lambda_k|}{\partial M} = \begin{pmatrix} B_p L J^{-1} \\ K J^{-1} \end{pmatrix}^\top \frac{\partial |\lambda_k|}{\partial \bar{A}} \begin{pmatrix} 0 & I_n \end{pmatrix}^\top \tag{4.95}$$

$$\frac{\partial |\lambda_k|}{\partial N} = \begin{pmatrix} B_p L J^{-1} \\ K J^{-1} \end{pmatrix}^\top \frac{\partial |\lambda_k|}{\partial \bar{A}} \begin{pmatrix} \tilde{C}_p & 0 \end{pmatrix}^\top \tag{4.96}$$

$$\frac{\partial |\lambda_k|}{\partial P} = \begin{pmatrix} 0 \\ I_n \end{pmatrix}^\top \frac{\partial |\lambda_k|}{\partial \bar{A}} \begin{pmatrix} 0 & I_n \end{pmatrix}^\top \tag{4.97}$$

$$\frac{\partial |\lambda_k|}{\partial Q} = \begin{pmatrix} 0 \\ I_n \end{pmatrix}^\top \frac{\partial |\lambda_k|}{\partial \bar{A}} \begin{pmatrix} \tilde{C}_p & 0 \end{pmatrix}^\top \tag{4.98}$$

$$\frac{\partial |\lambda_k|}{\partial R} = \begin{pmatrix} B_p \\ 0 \end{pmatrix}^\top \frac{\partial |\lambda_k|}{\partial \bar{A}} \begin{pmatrix} 0 & I_n \end{pmatrix}^\top \tag{4.99}$$

$$\frac{\partial |\lambda_k|}{\partial S} = \begin{pmatrix} B_p \\ 0 \end{pmatrix}^\top \frac{\partial |\lambda_k|}{\partial \bar{A}} \begin{pmatrix} \tilde{C}_p & 0 \end{pmatrix}^\top \tag{4.100}$$

$$\frac{\partial |\lambda_k|}{\partial J} = -\begin{pmatrix} B_p L J^{-1} \\ K J^{-1} \end{pmatrix}^\top \frac{\partial |\lambda_k|}{\partial \bar{A}} \begin{pmatrix} J^{-1} N \tilde{C}_p & J^{-1} M \end{pmatrix}^\top \tag{4.101}$$

On regroupe cela grâce à l'équation (4.22). ■

Remarque 1 : Bien que l'on puisse écrire

$$\bar{A} = \begin{pmatrix} A_p & 0 \\ 0 & 0 \end{pmatrix} + \bar{M}_1 Z \bar{M}_2 \tag{4.102}$$

$\bar{M}_1$ et $\bar{M}_2$ sont eux même fonction de Z et l'on ne peut appliquer le lemme 4.9 sur cette relation.
Remarque 2 : on remarquera aussi que $\frac{\partial|\lambda_k|}{\partial Z}$ ne dépend pas du choix de la réalisation (A_p, B_p, C_p).

Finalement, le calcul de $\frac{\partial|\lambda_k|}{\partial \bar{A}}$ se fait grâce à la proposition suivante :

Proposition 4.11
Soit $M \in \mathbb{R}^{n\times n}$ une matrice diagonalisable. On note $(\lambda_k)_{1\leqslant k\leqslant n}$ ses valeurs propres et $(x_k)_{1\leqslant k\leqslant n}$ les vecteurs propres à droite correspondants.
En notant

$$M_x \triangleq \begin{pmatrix} x_1 x_2 \dots x_n \end{pmatrix} \tag{4.103}$$

et

$$M_y = \begin{pmatrix} y_1 y_2 \dots y_n \end{pmatrix} \triangleq M_x^{-H} \tag{4.104}$$

on a alors $\forall k,\ 1 \leqslant k \leqslant n$:

$$\frac{\partial \lambda_k}{\partial M} = y_k^* x_k^\top \tag{4.105}$$

$$\frac{\partial\,|\lambda_k|}{\partial M} = \frac{1}{|\lambda_k|} Re\left(\lambda_k^* \frac{\partial \lambda_k}{\partial M}\right) \tag{4.106}$$

où $.^*$ est l'opérateur conjugué, $Re(.)$ la partie réelle et $.^H$ l'opérateur de transconjugué.

Démonstration :
On trouvera la démonstration dans [31] et [118] ■

4.4.4 Cas particuliers

On peut facilement retrouver comme cas particulier les mesures de sensibilités de la littérature obtenues dans le cas de q-structurations et de δ-structurations [117].
En partitionnant $\frac{\partial|\lambda_k|}{\partial \bar{A}}$, qui est de taille $(n+n_p)\times(n+n_p)$ de la manière suivante

$$\frac{\partial|\lambda_k|}{\partial \bar{A}} = \begin{pmatrix} \alpha_k & \beta_k \\ \gamma_k & \theta_k \end{pmatrix} \tag{4.107}$$

avec $\alpha_k \in \mathbb{R}^{n_p\times n_p}$, $\beta_k \in \mathbb{R}^{n_p\times n}$, $\gamma_k \in \mathbb{R}^{n\times n_p}$ et $\theta_k \in \mathbb{R}^{n\times n}$, on peut écrire :

– Dans le cas de la q-structuration (cf. 3.2.1) :

$$\frac{\partial|\lambda_k|}{\partial Z} = \begin{pmatrix} . & . & . \\ . & \theta_k & \gamma_k \tilde{C}_p^\top \\ . & B_p^\top \beta_k & B_p^\top \alpha_k \tilde{C}_p^\top \end{pmatrix} \tag{4.108}$$

et

$$\left\| \frac{\partial|\lambda_k|}{\partial Z} \times W_Z \right\|_F^2 = \|\theta_k\|_F^2 + \left\|\gamma_k \tilde{C}_p^\top\right\|_F^2 + \left\|B_p^\top \beta_k\right\|_F^2 + \left\|B_p^\top \alpha_k \tilde{C}_p^\top\right\|_F^2$$

– Pour une δ-structuration (cf. 3.2.3.1) :

$$\frac{\partial|\lambda_k|}{\partial Z} = \begin{pmatrix} \Delta\gamma_k \tilde{C}_p^\top B_\delta^\top + \Delta\theta_k A_\delta^\top & \Delta\theta_k & \Delta\gamma_k \tilde{C}_p^\top \\ \gamma_k \tilde{C}_p^\top B_\delta^\top + \theta_k A_\delta^\top & \theta_k & \gamma_k \tilde{C}_p^\top \\ B_p^\top \alpha_k \tilde{C}_p^\top B_\delta^\top + B_p^\top \beta_k A_\delta^\top & B_p^\top \beta_k & B_p^\top \alpha_k \tilde{C}_p^\top \end{pmatrix} \tag{4.109}$$

et

$$\left\| \frac{\partial|\lambda_k|}{\partial Z} \times W_Z \right\|_F^2 = \Delta^2 \|\theta_k\|_F^2 + \Delta^2 \left\|\gamma_k \tilde{C}_p^\top\right\|_F^2 + \left\|B_p^\top \beta_k\right\|_F^2 + \left\|B_p^\top \alpha_k \tilde{C}_p^\top\right\|_F^2$$

Ces deux résultats sont conformes à ceux de [117] qui comparent la mesure de stabilité pour des réalisations en q et en δ.
On remarquera aussi que le paramètre Δ qui intervient dans la réalisation en δ est supposé, dans tous les travaux précédents sur cet opérateur ([17, 19, 31, 70, 77, 97, 117]), comme implémenté exactement : jamais la sensibilité de la fonction de transfert ou des pôles vis-à-vis de ce paramètre Δ n'est étudié. Il est possible ici de le prendre en compte, car la sensibilité exprimée par ce paramètre nous est donnée par $\frac{\partial|\lambda_k|}{\partial K}$ qui vaut $\gamma_k \tilde{C}_p^\top B_\delta^\top + \theta_k A_\delta^\top$. Toutefois, ce paramètre pouvant être choisi (on ne le prend pas nécessairement égal à la période d'échantillonnage), on s'arrangera, en pratique, pour choisir une valeur représentable exactement en précision finie (clasiquement une puissance de 2).

4.4.5 Exemple académique

Pour appliquer ces mesures de stabilité, nous reprenons l'exemple que I. Njabeleke et J. Whidborne[13] [84, 106, 118] exhibent. Il s'agit d'un régulateur SISO de machine hydraulique, obtenu par discrétisation d'un régulateur H_∞ optimal.
Le système discret est donné par

$$A_p = \begin{pmatrix} 9.9988e{-}1 & 1.9432e{-}5 & 5.9320e{-}5 & -6.2286e{-}5 \\ -4.9631e{-}7 & 2.3577e{-}2 & 2.3709e{-}5 & 2.3672e{-}5 \\ -1.5151e{-}3 & 2.3709e{-}2 & 2.3751e{-}5 & 2.3898e{-}5 \\ 1.5908e{-}3 & 2.3672e{-}2 & 2.3898e{-}5 & 2.3667e{-}5 \end{pmatrix} \tag{4.110}$$

[13] Nous remercions James Whidborne de nous avoir fourni le source définissant le système et le régulateur avec la précision suffisante.

$$B_p = \begin{pmatrix} 3.0504e{-}3 \\ -1.2373e{-}2 \\ -1.2375e{-}2 \\ -8.8703e{-}2 \end{pmatrix} C_p = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}^\top \tag{4.111}$$

Le régulateur est donné ici par une réalisation canonique

$$Z_0 = \left(\begin{array}{cccc:cc} & & & & & \\ \hdashline 0 & 0 & 0 & \mathbf{-3.307e{-}1} & 1 & 0 \\ 1 & 0 & 0 & \mathbf{1.987e{+}0} & 0 & 0 \\ 0 & 1 & 0 & \mathbf{-3.982e{+}0} & 0 & 0 \\ 0 & 0 & 1 & \mathbf{3.325e{+}0} & 0 & 0 \\ \hdashline \mathbf{-1.611e{-}3} & \mathbf{-1.600e{-}3} & \mathbf{-1.589e{-}3} & \mathbf{-1.577e{-}3} & \mathbf{-8.084e{-}4} & 1 \end{array}\right)$$

On notera que la précision de l'affichage des coefficients n'est pas suffisante ici pour définir le système : on le verra, la stabilité du système peut être très vite compromise par les quantifications des coefficients de Z_0.
Les pôles du système boucle fermée sont

$$\lambda_1 = 9.9956e{-}01 + 2.6126e{-}04i \tag{4.112}$$
$$\lambda_2 = 9.9956e{-}01 - 2.6126e{-}04i \tag{4.113}$$
$$\lambda_3 = 9.9956e{-}01 \tag{4.114}$$
$$\lambda_4 = 9.9333e{-}01 \tag{4.115}$$
$$\lambda_5 = 3.3333e{-}01 \tag{4.116}$$
$$\lambda_6 = 2.3625e{-}02 \tag{4.117}$$
$$\lambda_7 = 9.7531e{-}19 \tag{4.118}$$
$$\lambda_8 = -3.8735e{-}09 \tag{4.119}$$

Les pôles λ_1 à λ_4, qui sont en module proche de 1 ont pour sensibilité[14] :

$$\left.\frac{\partial|\lambda_1|}{\partial Z}\right|_{Z_0} = \left(\begin{array}{cccc:cc} & & & & & \\ \hdashline -1.747e{+}5 & 8.749e{+}5 & -1.228e{+}6 & \mathbf{5.282e{+}5} & -4.439e{-}4 & 0 \\ -1.718e{+}5 & 8.604e{+}5 & -1.208e{+}6 & \mathbf{5.194e{+}5} & -4.402e{-}4 & 0 \\ -1.689e{+}5 & 8.460e{+}5 & -1.188e{+}6 & \mathbf{5.107e{+}5} & -4.365e{-}4 & 0 \\ -1.661e{+}5 & 8.317e{+}5 & -1.168e{+}6 & \mathbf{5.021e{+}5} & -4.329e{-}4 & 0 \\ \hdashline \mathbf{1.780e{+}6} & \mathbf{-8.913e{+}6} & \mathbf{1.251e{+}7} & \mathbf{-5.380e{+}6} & \mathbf{2.158e{-}3} & 0 \end{array}\right)$$

$$\left.\frac{\partial|\lambda_2|}{\partial Z}\right|_{Z_0} = \left(\begin{array}{cccc:cc} & & & & & \\ \hdashline -1.747e{+}5 & 8.749e{+}5 & -1.228e{+}6 & \mathbf{5.282e{+}5} & -4.439e{-}4 & 0 \\ -1.718e{+}5 & 8.604e{+}5 & -1.208e{+}6 & \mathbf{5.194e{+}5} & -4.402e{-}4 & 0 \\ -1.689e{+}5 & 8.460e{+}5 & -1.188e{+}6 & \mathbf{5.107e{+}5} & -4.365e{-}4 & 0 \\ -1.661e{+}5 & 8.317e{+}5 & -1.168e{+}6 & \mathbf{5.021e{+}5} & -4.329e{-}4 & 0 \\ \hdashline \mathbf{1.780e{+}6} & \mathbf{-8.913e{+}6} & \mathbf{1.251e{+}7} & \mathbf{-5.380e{+}6} & \mathbf{2.158e{-}3} & 0 \end{array}\right)$$

[14] On donne la sensibilité pour tous les coefficients, en exhibant en gras les coefficients dont il faut tenir compte.

$$\left.\frac{\partial|\lambda_3|}{\partial Z}\right|_{Z_0} = \left(\begin{array}{ccccc|cc} & & & & & & \\ \hline 3.631e{+}5 & -1.818e{+}6 & 2.553e{+}6 & \mathbf{-1.098e{+}6} & 9.041e{-}5 & 0 \\ 3.572e{+}5 & -1.789e{+}6 & 2.511e{+}6 & \mathbf{-1.080e{+}6} & 8.894e{-}5 & 0 \\ 3.514e{+}5 & -1.760e{+}6 & 2.470e{+}6 & \mathbf{-1.062e{+}6} & 8.748e{-}5 & 0 \\ 3.456e{+}5 & -1.730e{+}6 & 2.429e{+}6 & \mathbf{-1.044e{+}6} & 8.603e{-}5 & 0 \\ \hline \mathbf{-3.566e{+}6} & \mathbf{1.786e{+}7} & \mathbf{-2.507e{+}7} & \mathbf{1.078e{+}7} & \mathbf{-8.879e{-}4} & 0 \end{array}\right)$$

$$\left.\frac{\partial|\lambda_4|}{\partial Z}\right|_{Z_0} = \left(\begin{array}{ccccc|cc} & & & & & & \\ \hline -1.363e{+}4 & 6.818e{+}4 & -9.549e{+}4 & \mathbf{4.095e{+}4} & 7.974e{-}4 & 0 \\ -1.353e{+}4 & 6.767e{+}4 & -9.478e{+}4 & \mathbf{4.064e{+}4} & 7.914e{-}4 & 0 \\ -1.343e{+}4 & 6.717e{+}4 & -9.408e{+}4 & \mathbf{4.034e{+}4} & 7.856e{-}4 & 0 \\ -1.333e{+}4 & 6.667e{+}4 & -9.338e{+}4 & \mathbf{4.004e{+}4} & 7.797e{-}4 & 0 \\ \hline \mathbf{6.366e{+}3} & \mathbf{-3.184e{+}4} & \mathbf{4.459e{+}4} & \mathbf{-1.912e{+}4} & \mathbf{-3.723e{-}4} & 0 \end{array}\right)$$

On peut remarquer que celles-ci sont particulièrement élevées (surtout pour λ_3), et qu'elles sont égales pour les pôles complexes conjugués.
On peut observer la valeur de sensibilité pour chaque pôle

$$\left(\frac{1-|\lambda_k|}{\left\|\left.\frac{\partial|\lambda_k|}{\partial Z}\right|_{Z_0} \times W_Z\right\|_S}\right)_{1\leqslant k\leqslant n+n_p} = \begin{pmatrix} 1.4409e{-}11 \\ 1.4409e{-}11 \\ 7.1128e{-}12 \\ 2.5273e{-}8 \\ 1.2859e{-}1 \\ 1.2732e{+}5 \\ 5.6213e{+}3 \\ 6.5386e{+}3 \end{pmatrix} \tag{4.120}$$

le minimum étant atteint pour λ_3 (ce pôle-là est donc celui qui amènera, potentiellement, en premier, à l'instabilité).
Ceci nous donne les mesures (la représentation virgule fixe est considérée)

$$\mu_1(Z_0) = 7.1128e-12 \qquad \mu_2(Z_0) = 4.4392e-12 \tag{4.121}$$

ce qui est conforme aux résultats de [118].

Nous pouvons aussi appliquer cette mesure sur une autre réalisation, par exemple une forme en δ (le chapitre 5 nous permettra d'appliquer cette mesure sur beaucoup d'autres réalisations, bien plus intéressantes que celles-ci).
On fixe $\Delta = 2^{-3}$, et on considère la réalisation canonique

$$Z_1 = \left(\begin{array}{cccc|ccccc} -1 & 0 & 0 & 0 & \mathbf{-5.396e{+}0} & \mathbf{-3.357e{-}1} & \mathbf{-1.209e{-}3} & \mathbf{-9.685e{-}9} & 1 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ \hline \Delta & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & \Delta & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \Delta & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \Delta & 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & \mathbf{-1.289e{-}2} & \mathbf{-6.883e{-}2} & \mathbf{-4.468e{-}4} & \mathbf{-6.868e{-}7} & \mathbf{-8.084e{-}4} \end{array}\right)$$

Dans ce cas, la sensibilité est plus faible

$$\left(\frac{1-|\lambda_k|}{\left\| \frac{\partial|\lambda_k|}{\partial Z}\Big|_{Z_1} \times W_Z \right\|_S} \right)_{1\leqslant k\leqslant n+n_p} = \begin{pmatrix} 4.2680e{+}0 \\ 4.3051e{-}4 \\ 3.0519e{-}7 \\ 3.0519e{-}7 \\ 1.5047e{-}7 \\ 3.1537e{+}6 \\ 4.2685e{+}24 \\ 1.4672e{+}5 \end{pmatrix} \tag{4.122}$$

Cette fois, le minimum est atteint pour $k = 5$, bien que λ_5 ne soit pas le pôle le plus proche de 1 (mais c'est le pôle dont le rapport distance de l'unité / sensibilité est le plus petit).
On obtient les mesures

$$\mu_1(Z_0) = 1.5047e-07 \qquad \mu_2(Z_0) = 5.5171e-08 \tag{4.123}$$

4.4.6 Nombre de bits pour la précision

Nous avons vu (équation (4.80)) que

$$\|\Delta\|_{\max} < \mu(Z) \Rightarrow \bar{A}(Z + r_Z \times \Delta) \text{ est stable} \tag{4.124}$$

et que l'écart de précision engendré par la quantification de Z vérifie (cf. (3.96))

$$\|\Delta\|_{\max} \leqslant 2^{-(\beta_p+1)} \tag{4.125}$$

où β_p est le nombre de bits de précision du format de représentation ($\beta_p = \beta_f$ pour la virgule fixe, et $\beta_p = \beta_m$ pour la virgule flottante). On pourra donc estimer le nombre de bits de précision β_p minimum nécessaire pour assurer la stabilité du système bouclé par

$$\hat{\beta}_p^{\min} \triangleq -\lceil \log_2 \mu(Z) \rceil \tag{4.126}$$

Appliqué aux réalisations de l'exemple précédent, ceci donne

Réalisation	virgule fixe	virgule flottante	virgule flottante par bloc
Z_0	38	36	37
Z_1	23	6	22

TAB. 4.5 – Nombre de bits minimum selon le format

On pourra se reporter à [120] pour la comparaison des nombres de bits nécessaires dans les différents cas.

4.5 Conclusion

Ce chapitre nous a permis de présenter dans le détail différents critères d'évaluation de la dégradation due à l'implémentation en précision finie. Développés pour l'occasion ou simplement généralisés, à partir de ceux élaborés dans le cadre de travaux précurseurs (Gevers, Williamson, Wu, etc.) au cas de la forme implicite spécialisée, ces critères permettent d'évaluer la sensibilité de la relation entrées/sorties vis-à-vis des coefficients ou encore la sensibilité des pôles et la distance à l'instabilité. Critères pondérés ou non, en boucle ouverte ou boucle fermée dans le cas de régulateurs ont été distingués.

S'appliquant à la forme implicite, ces critères sont d'une portée plus générale que précédemment et ne nécessite pas de développements différentiés comme c'était le cas jusqu'à présent pour les réalisations en q, δ, retour d'état-observateur, etc.

Chapitre

5 Synthèse de réalisations

Résumé :

Nous avons présenté, au chapitre précédent, des mesures permettant d'évaluer l'impact du processus de quantification sur une réalisation donnée. Ce chapitre va nous permettre de déterminer, pour une loi initiale de *référence*, un ensemble de réalisations équivalentes afin de pouvoir choisir, parmi elles, la mieux appropriée en vue de l'implémentation. En particulier, le problème de l'optimisation consistant à rechercher la *meilleure* réalisation en terme de résilience, sera détaillé et illustré de manière pratique. Différentes hypothèses seront envisagées : optimisation sur une classe d'équivalence restreinte en terme d'ordre ou de structure, différents critères, etc.

Sommaire

5.1 Réalisations optimales

On considère ici une loi LTI que l'on cherche à implémenter. Celle-ci pourra être donnée sous la forme d'une matrice de transfert H ou d'une réalisation spécifique $\mathcal{R}_0$.

Le problème de synthèse de réalisation optimale consiste à trouver, parmi les réalisations équivalentes à $\mathcal{R}_0$ (toutes les réalisations associées à la même matrice de transfert H), celles qui optimisent un critère donné noté $\mathcal{J}$. Ce critère peut être un de ceux exprimés au chapitre 4 (le critère de coût de calcul ne sera pas considéré ici comme un critère retenu dans la recherche de réalisations optimales, car assez peu discriminatoire. Une solution pour prendre en compte ce paramètre important sera présentée – mais pas développée – au paragraphe 5.4.3).
On supposera par la suite que ce critère $\mathcal{J}$ est à minimiser (dans le cas de la mesure de stabilité – qui est un critère à maximiser, cf. section 4.4, on considérera son inverse).
Ce problème s'écrit donc

Problème 5.1 (Recherche d'une réalisation *optimale*)

$$\mathcal{R}_{opt} = \underset{\mathcal{R}\in\mathscr{R}_H}{arg\ min}\ \mathcal{J}(\mathcal{R}) \tag{5.1}$$

où $\mathscr{R}_H$ est l'ensemble des réalisations sous forme implicite spécialisée de transfert H (cf. définition 3.3).

Le choix de la forme implicite spécialisée permet d'accroître considérablement la taille de cet ensemble. Avantage indéniable en ce qu'il augmente fortement les chances de trouver une réalisation support d'une implémentation efficace (bonne résilience, faible coût), ce choix complique cependant la résolution du problème d'optimisation.
On se restreindra, en pratique, à un sous-ensemble de $\mathscr{R}_H$. Dans la plupart des cas, on s'imposera de chercher parmi des réalisations de dimension maximale fixée (pas nécessairement l'ordre de matrice de transfert) ou encore en privilégiant certaines structures.
On considère ainsi une structuration $\mathscr{S}$. On pourra trouver une solution sous-optimale du problème de synthèse en considérant le problème de synthèse de réalisation structurée optimale suivant :

Problème 5.2 (Recherche d'une réalisation *sous-optimale*)

$$\mathcal{R}^{\mathscr{S}}_{opt} = \underset{\mathcal{R}\in\mathscr{R}^{\mathscr{S}}_H}{arg\ min}\ \mathcal{J}(\mathcal{R}) \tag{5.2}$$

où $\mathscr{R}^{\mathscr{S}}_H$ est l'ensemble des réalisations structurées selon $\mathscr{S}$ et ayant H pour transfert entrées/sorties (cf. définition 3.5).

La problématique repose donc sur la description de $\mathscr{R}_H^{\mathscr{S}}$ et sur les possibilités de parcourir cet ensemble. La description des classes d'équivalences de la section 3.3 va donc être appliquée ici, pour divers cas.
Nous opérons une distinction selon que $\mathscr{R}_H^{\mathscr{S}}$ est fini ou non.

5.2 Cas où $\mathscr{R}_H^{\mathscr{S}}$ est fini

On traitera ici trois exemples simples pour lesquels, selon la structuration choisie, il n'existe qu'un nombre fini de réalisations équivalentes à la réalisation $\mathcal{R}_0$ de référence.

5.2.1 Décomposition en cascade

Comme décrit au paragraphe 2.1.8.1, nous considérerons une loi SISO décomposée en produit de systèmes du second ordre (pour simplifier le propos, mais sans perte de généralité, on supposera l'ordre n du système pair, et les numérateurs et dénominateurs de dimensions égales) :

$$H(z) = K \prod_{j=1}^{\frac{n}{2}} H_j(z) \qquad \forall z \in \mathbb{C} \tag{5.3}$$

où chaque système H_j est du 2^{nd} ordre.

En nommant $(p_i)_{1\leqslant i\leqslant n}$ et $(q_i)_{1\leqslant i\leqslant n}$ les pôles et zéros de H, le choix du découpage revient à choisir deux pôles et deux zéros parmi $(p_i)_{1\leqslant i\leqslant n}$ et $(q_i)_{1\leqslant i\leqslant n}$ pour chaque système H_j, sans séparer les pôles et zéros complexes conjugués.
Chaque fonction de transfert H_j s'écrira donc

$$H_j(z) = \frac{\left(z - p_{\sigma_1(i)}\right)\left(z - p_{\sigma_2(i)}\right)}{\left(z - q_{\rho_1(i)}\right)\left(z - q_{\rho_2(i)}\right)} \tag{5.4}$$

où σ_1, σ_2, ρ_1 ,ρ_2 sont des fonctions de découpage $[\![1; \frac{n}{2}]\!] \to [\![1; n]\!]$ qui vérifient :

$$\sigma_1\left([\![1; \frac{n}{2}]\!]\right) \cap \sigma_2\left([\![1; \frac{n}{2}]\!]\right) = \varnothing \tag{5.5}$$

$$\sigma_1\left([\![1; \frac{n}{2}]\!]\right) \cup \sigma_2\left([\![1; \frac{n}{2}]\!]\right) = [\![1; n]\!] \tag{5.6}$$

$$\rho_1\left([\![1; \frac{n}{2}]\!]\right) \cap \rho_2\left([\![1; \frac{n}{2}]\!]\right) = \varnothing \tag{5.7}$$

$$\rho_1\left([\![1; \frac{n}{2}]\!]\right) \cup \rho_2\left([\![1; \frac{n}{2}]\!]\right) = [\![1; n]\!] \tag{5.8}$$

$$\text{si } p_{\sigma_1(j)} \text{ est complexe, alors } p_{\sigma_2(j)} = \overline{p_{\sigma_1(j)}} \tag{5.9}$$

$$\text{si } q_{\rho_1(j)} \text{ est complexe, alors } q_{\rho_2(j)} = \overline{q_{\rho_1(j)}} \tag{5.10}$$

(les valeurs de σ_1 et σ_2 sont disjointes ; leur réunion forme l'ensemble des indices $[\![1;n]\!]$ et les complexes conjugués sont associés dans la même paire). Ainsi constituées, les lois de découpage définissent les sous-systèmes $(H_j)_{1\leqslant j\leqslant \frac{n}{2}}$. Notons que ce découpage distingue la mise en cascade de H_1 et H_2 de la mise en cascade de H_2 et H_1. Les fonctions de découpage peuvent amener à deux systèmes où les pôles et zéros sont regroupés de la même manière, mais dans un ordre différent.

On pourra montrer que si $(p_i)_{1\leqslant i\leqslant n}$ (respectivement $(q_i)_{1\leqslant i\leqslant n}$) ne possède aucune valeur complexe, alors il existe

$$\frac{n!}{2^{\frac{n}{2}}} \tag{5.11}$$

fonctions de découpage (σ_1,σ_2) (resp. (ρ_1,ρ_2)) vérifiant les propriétés (5.5) à (5.10) [1]. Enfin, si on a m pôles (respectivement zéros) complexes conjugués, on peut montrer qu'il existe alors

$$\frac{(n-m)!}{2^{\frac{n-m}{2}}}\frac{\left(\frac{n}{2}\right)!}{\left(\frac{n-m}{2}\right)!} \tag{5.12}$$

fonctions de découpages possibles.
Pour avoir un ordre de grandeur, le tableau 5.1 donne quelques valeurs.

		m					
		10	8	6	4	2	0
n	10	120	120	360	1800	12600	113400
	8	×	24	24	72	360	2520
	6	×	×	6	6	18	90
	4	×	×	×	2	2	6

TAB. 5.1 – Quelques valeurs du nombre de découpage possibles

Enfin, puisqu'il faut deux fonctions de découpage (une pour les zéros, une pour les pôles), il y a donc

$$\frac{\left(\frac{n}{2}\right)!}{2^{n-\frac{m_p+m_q}{2}}}\frac{(n-m_p)!(n-m_q)!}{\left(\frac{n-m_p}{2}\right)!\left(\frac{n-m_q}{2}\right)!} \tag{5.13}$$

possibilités (en notant m_p le nombre de pôles complexes conjugués et m_q le nombre de zéros complexes conjugués). En pratique, il devient impossible d'examiner toutes les décompositions lorsque $n-m>10$.

[1] En effet, il y a $\frac{n(n-1)}{2}$ possibilités pour choisir une paire $(\sigma_1(1),\sigma_2(1))$ d'indices parmi n. Il y a ensuite $\frac{(n-2)(n-3)}{2}$ possibilités pour choisir $(\sigma_1(2),\sigma_2(2))$, etc.

Enfin, on choisira, par exemple[2], pour réalisation de chaque fonction de transfert élémentaire H_j, une réalisation sous forme équilibrée (le paragraphe 4.2.3 ayant montré que les formes équilibrées présentaient une bonne sensibilité paramétrique).
Pour chaque fonction de découpage σ_1, σ_2, ρ_1, ρ_2 choisie, il nous est possible de construire la réalisation équivalente notée $\mathcal{R}_{\sigma_1,\sigma_2,\rho_1,\rho_2}$ (sous forme implicite spécialisée) : la mise en cascade est donnée au paragraphe 3.2.4, et les réalisations équilibrées aux sections 3.2.1 et 2.1.2.

En choisissant, par exemple, la mesure de sensibilité paramétrique $M_{L_2}^W$ comme critère d'évaluation, le problème de synthèse de réalisation structurée en cascade s'écrit

$$\mathcal{R}_{opt} = \underset{\sigma_1,\sigma_2,\rho_1,\rho_2}{\arg\min}\ M_{L_2}^W\left(\mathcal{R}_{\sigma_1,\sigma_2,\rho_1,\rho_2}\right) \tag{5.14}$$

On considère l'exemple suivant (cf. [31]) :

$$H(z) = \frac{z^4 + 1.0706z^3 + 0.1016z^2 + 0.0460z + 0.0230}{z^4 - 2.9003z^3 + 3.3837z^2 - 1.8938z + 0.4199} \tag{5.15}$$

Les pôles et zéros de H sont

$$\begin{array}{lllllll} p_1 & = & -0.9913 & & q_1 & = & 0.6585 + 0.5061i \\ p_2 & = & 0.1026 + 0.2663i & & q_2 & = & 0.6585 - 0.5061i \\ p_3 & = & 0.1026 - 0.2663i & & q_3 & = & 0.9254 \\ p_4 & = & -0.2846 & & q_4 & = & 0.6579 \end{array} \tag{5.16}$$

Suivant les règles (5.5) à (5.10), les décompositions possibles sont :

$$G_1(z) = \frac{(z-p_1)(z-p_4)}{(z-q_1)(z-q_2)} \cdot \frac{(z-p_2)(z-p_3)}{(z-q_3)(z-q_4)} \tag{5.17}$$

$$G_2(z) = \frac{(z-p_1)(z-p_4)}{(z-q_3)(z-q_4)} \cdot \frac{(z-p_2)(z-p_3)}{(z-q_1)(z-q_2)} \tag{5.18}$$

$$G_3(z) = \frac{(z-p_2)(z-p_3)}{(z-q_1)(z-q_2)} \cdot \frac{(z-p_1)(z-p_4)}{(z-q_3)(z-q_4)} \tag{5.19}$$

$$G_4(z) = \frac{(z-p_2)(z-p_3)}{(z-q_3)(z-q_4)} \cdot \frac{(z-p_1)(z-p_4)}{(z-q_1)(z-q_2)} \tag{5.20}$$

ce qui correspond aux réalisations Z_1 à Z_4 :

$$Z_1 = \left(\begin{array}{c|cc|cc|c} -1 & \mathbf{-1.75628} & \mathbf{0.70116} & 0 & 0 & \mathbf{1} \\ \hdashline 0 & \mathbf{0.67325} & \mathbf{0.50631} & 0 & 0 & \mathbf{-1.75628} \\ 0 & \mathbf{-0.50631} & \mathbf{0.64374} & 0 & 0 & \mathbf{-0.70116} \\ \mathbf{-1.42306} & 0 & 0 & \mathbf{0.94258} & \mathbf{0.06995} & 0 \\ \mathbf{-0.80443} & 0 & 0 & \mathbf{-0.06995} & \mathbf{0.64069} & 0 \\ \hdashline \mathbf{1} & 0 & 0 & \mathbf{-1.42306} & \mathbf{0.80443} & 0 \end{array}\right)$$

[2]Il est tout à fait possible, encore une fois, de choisir une autre forme de réalisation : une forme directe, réalisation en δ, etc.

$$Z_2 = \left(\begin{array}{c|cc|cc|c} -1 & \mathbf{-2.37002} & \mathbf{1.66067} & 0 & 0 & \mathbf{1} \\ \hline 0 & \mathbf{0.94757} & \mathbf{0.08014} & 0 & 0 & \mathbf{-2.37002} \\ 0 & \mathbf{-0.08014} & \mathbf{0.63570} & 0 & 0 & \mathbf{-1.66067} \\ \mathbf{-1.07059} & 0 & 0 & \mathbf{0.59147} & \mathbf{0.51052} & 0 \\ \mathbf{-0.18559} & 0 & 0 & \mathbf{-0.51052} & \mathbf{0.72552} & 0 \\ \hline \mathbf{1} & 0 & 0 & \mathbf{-1.07059} & \mathbf{0.18559} & 0 \end{array}\right)$$

$$Z_3 = \left(\begin{array}{c|cc|cc|c} -1 & \mathbf{-1.07059} & \mathbf{0.18559} & 0 & 0 & \mathbf{1} \\ \hline 0 & \mathbf{0.59147} & \mathbf{0.51052} & 0 & 0 & \mathbf{-1.07059} \\ 0 & \mathbf{-0.51052} & \mathbf{0.72552} & 0 & 0 & \mathbf{-0.18559} \\ \mathbf{-2.37002} & 0 & 0 & \mathbf{0.94757} & \mathbf{0.08014} & 0 \\ \mathbf{-1.66067} & 0 & 0 & \mathbf{-0.08014} & \mathbf{0.63570} & 0 \\ \hline \mathbf{1} & 0 & 0 & \mathbf{-2.37002} & \mathbf{1.66067} & 0 \end{array}\right)$$

$$Z_4 = \left(\begin{array}{c|cc|cc|c} -1 & \mathbf{-1.42306} & \mathbf{0.80443} & 0 & 0 & \mathbf{1} \\ \hline 0 & \mathbf{0.94258} & \mathbf{0.06995} & 0 & 0 & \mathbf{-1.42306} \\ 0 & \mathbf{-0.06995} & \mathbf{0.64069} & 0 & 0 & \mathbf{-0.80443} \\ \mathbf{-1.75628} & 0 & 0 & \mathbf{0.67325} & \mathbf{0.50631} & 0 \\ \mathbf{-0.70116} & 0 & 0 & \mathbf{-0.50631} & \mathbf{0.64374} & 0 \\ \hline \mathbf{1} & 0 & 0 & \mathbf{-1.75628} & \mathbf{0.70116} & 0 \end{array}\right)$$

Bien qu'équivalentes, ces réalisations présentent une mesure $M_{L_2}^W$ différente, que nous présentons au tableau 5.2.

Réalisation	$M_{L_2}^W$
Z_1	$2.4250e+05$
Z_2	$2.6920e+05$
Z_3	$2.6920e+05$
Z_4	$2.4250e+05$

TAB. 5.2 – Mesure de sensibilité pour les quatre réalisations étudiées

On remarque tout d'abord que, bien que l'ordre de grandeur de la mesure de sensibilité soit le même pour les quatre réalisations, Z_1 et Z_4 possèdent une plus faible sensibilité que Z_2 et Z_3. Ensuite, les réalisations Z_1, Z_4 et Z_2, Z_3 possèdent la même sensibilité : on peut montrer sans difficultés (à partir des expressions des sensibilités et de l'expression (3.50)) que l'ordre de la mise en cascade n'a pas d'incidence sur la sensibilité paramétrique (par contre, l'ordre aura un impact très important sur le bruit de quantification). Il est ainsi possible de choisir le découpage en cascade qui propose la meilleure valeur d'un critère de résilience en précision finie.
L'exemple du paragraphe 5.5 nous montrera qu'avec une réalisation d'ordre plus élevé, les différentes réalisations cascade peuvent présenter des sensibilités bien différentes.
Une décomposition en parallèle peut également être considérée. Ces approches de décomposition peuvent amener à une méthodologie d'implémentation à base de bibliothèque de blocs élémentaires optimisés que l'on as-

semblerait. Cette voie est une perspective qui mériterait d'être explorée, pour répondre à certains besoins exprimés dans le domaine automobile et aéronautique.

5.2.2 Retour d'état / Observateur

Il peut être intéressant, pour diverses raisons déjà évoquées au paragraphe 2.1.3 dont nous reprenons les notations, de mettre en œuvre un régulateur sous forme retour d'état / observateur.
Cette mise en forme revient à résoudre une équation de Riccati généralisée [4, 64]. L'utilisateur a alors besoin de répartir les pôles de la boucle fermée dans deux sous-espaces différents qui sont le sous-espace d'observation $(A_p - K_o C_p)$ et le sous-espace de commande $(A_p - B_p K_c)$ (dans le cas où l'on cherche à obtenir une forme retour d'état / observateur de dimension plus importante que l'observateur seul, il est possible d'introduire un paramètre de Youla Q : les pôles supplémentaires seront placés dans le sous-espace de Youla associé [4]). Les règles suivantes sont à respecter :

- afin de conserver les coefficients de cette représentation réels, il est nécessaire de ne pas séparer les paires de pôles complexes conjugués ;
- les pôles qui ne sont pas commandables sont affectés au sous-espace de commande ;
- les pôles qui ne sont pas observables sont affectés au sous-espace d'observation.

Afin d'obtenir une seule répartition des pôles de la boucle fermée, il est d'usage d'inter-classer les pôles de la boucle fermée, selon leur indice de commandabilité, et d'affecter les pôles de la boucle fermée proches du système en boucle ouverte au sous-espace de commande, et les pôles les plus rapides au sous-espace d'observation.
Ces dernières recommandations sont souvent appliquées afin d'avoir unicité de la répartition des pôles, et donc de la forme retour d'état / observateur. Ceci étant, certains pôles ont souvent un indice de commandabilité ou d'observabilité proche, et d'autres répartitions sont envisageables.

Ainsi, il est possible de considérer toutes les réalisations sous forme retour d'état / observateur d'un régulateur donné en considérant les différentes répartitions possibles. Celles-ci sont en nombre fini et dépendent du nombre de pôles, du nombre de pôles complexes conjugués et des pôles inobservables et non commandables.
On considère à nouveau l'exemple de la section 4.4.5.
Le tableau 5.3 donne les pôles de la boucle fermée, ainsi que les indices de commandabilité.
Remarque : on notera d'abord la difficulté que présente cet exemple, qui contient à la fois des pôles très rapides et très lents. En pratique, on pourrait s'interroger sur l'origine de ce conditionnement et le moyen de l'éviter en

	pôle	indice d'observabilité
λ_1	$9.9956e{-}1 + 2.6126e{-}04i$	1
λ_2	$9.9956e{-}1 - 2.6126e{-}04i$	1
λ_3	$9.9956e{-}1$	1
λ_4	$9.9333e{-}1$	1
λ_5	$3.3333e{-}1$	1
λ_6	$2.3625e{-}2$	$1.8937e{-}4$
λ_7	$9.7531e{-}19$	$3.6356e{-}13$
λ_8	$-3.8735e{-}9$	$1.0673e{-}3$

TAB. 5.3 – Pôles de la boucle fermée, indices d'observabilité

posant mieux le problème ou en se contentant d'une approximation d'ordre réduit. Telle n'est pas la question ici.

Ainsi, le pôle λ_7 étant quasi-inobservable, et les pôles λ_1 et λ_2 ne devant pas être séparés, il existe 15 partitionnements possibles, donnés par le tableau 5.4.

Les partionnements les plus *naturels*, c'est à dire ceux qui respectent les

partitionnement	sous-espace d'observation	sous-espace de commandabilité
1	λ_5, λ_6, λ_7, λ_8	λ_1, λ_2, λ_3, λ_4
2	λ_4, λ_6, λ_7, λ_8	λ_1, λ_2, λ_3, λ_5
3	λ_4, λ_5, λ_7, λ_8	λ_1, λ_2, λ_3, λ_6
4	λ_4, λ_5, λ_6, λ_7	λ_1, λ_2, λ_3, λ_8
5	λ_3, λ_6, λ_7, λ_8	λ_1, λ_2, λ_4, λ_5
6	λ_3, λ_5, λ_7, λ_8	λ_1, λ_2, λ_4, λ_6
7	λ_3, λ_5, λ_6, λ_7	λ_1, λ_2, λ_4, λ_8
8	λ_3, λ_4, λ_7, λ_8	λ_1, λ_2, λ_5, λ_6
9	λ_3, λ_4, λ_6, λ_7	λ_1, λ_2, λ_5, λ_8
10	λ_3, λ_4, λ_5, λ_7	λ_1, λ_2, λ_6, λ_8
11	λ_1, λ_2, λ_7, λ_8	λ_3, λ_4, λ_5, λ_6
12	λ_1, λ_2, λ_6, λ_7	λ_3, λ_4, λ_5, λ_8
13	λ_1, λ_2, λ_5, λ_7	λ_3, λ_4, λ_6, λ_8
14	λ_1, λ_2, λ_3, λ_7	λ_3, λ_5, λ_6, λ_8
15	λ_1, λ_2, λ_3, λ_7	λ_4, λ_5, λ_6, λ_8

TAB. 5.4 – Les 15 partionnements possibles

recommandations énoncées plus haut, sont les partitionnement n°1, 2 et 5. Le tableau 5.5 donne les mesures de stabilité respectives.

partionnement	μ_1	partionnement	μ_1
1	$2.9251e{-8}$	9	$3.6330e{-10}$
2	$2.5098e{-7}$	10	$1.8741e{-10}$
3	$1.9686e{-8}$	11	$2.9871e{-8}$
4	$5.9886e{-9}$	12	$6.8258e{-10}$
5	$6.5152e{-9}$	13	$1.4803e{-10}$
6	$6.9062e{-9}$	14	$1.2657e{-10}$
7	$3.0390e{-6}$	15	$9.4324e{-10}$
8	$5.6425e{-9}$		

TAB. 5.5 – Mesures de stabilité pour les différents partitionnements

5.2.3 Opérateur δ-modifié

On considère de nouveau l'exemple du paragraphe 5.2.1 :

$$H(z) = \frac{z^4 + 1.0706z^3 + 0.1016z^2 + 0.0460z + 0.0230}{z^4 - 2.9003z^3 + 3.3837z^2 - 1.8938z + 0.4199} \tag{5.21}$$

et on cherche à l'implémenter avec une réalisation en δ, par exemple une forme directe II. On choisit $\Delta = 7.6210e{-1}$, ce qui donne la réalisation Z_0

$$Z_0 = \left(\begin{array}{cccc:cccc:c}
-1 & 0 & 0 & 0 & \mathbf{-1.443e+0} & \mathbf{-1.176e+0} & \mathbf{-3.903e-1} & \mathbf{-2.820e-2} & 1 \\
0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\
\hdashline
\mathbf{7.621e-1} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & \mathbf{7.621e-1} & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & \mathbf{7.621e-1} & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & \mathbf{7.621e-1} & 0 & 0 & 0 & 1 & 0 \\
\hdashline
0 & 0 & 0 & 0 & \mathbf{5.210e+0} & \mathbf{1.486e+1} & \mathbf{1.647e+1} & \mathbf{6.616e+0} & 1
\end{array}\right)$$

La sensibilité paramétrique pondérée $\left.\frac{\delta\tilde{H}}{\delta Z}\right|_{Z_0} \times W_{Z_0}$

$$\left.\frac{\delta\tilde{H}}{\delta Z}\right|_{Z_0} \times W_{Z_0} = \left(\begin{array}{cccc:cccc:c}
0 & 0 & 0 & 0 & \mathbf{2.485e+1} & \mathbf{3.856e+1} & \mathbf{1.088e+2} & \mathbf{1.151e+3} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hdashline
\mathbf{2.538e+1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \mathbf{5.104e+1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \mathbf{5.557e+1} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \mathbf{4.596e+1} & 0 & 0 & 0 & 0 & 0 \\
\hdashline
0 & 0 & 0 & 0 & \mathbf{1.242e+0} & \mathbf{1.246e+0} & \mathbf{1.772e+0} & \mathbf{6.763e+0} & 0
\end{array}\right)$$

nous indique que la sensibilité de $\tilde{H}$ au paramètre Δ (via le paramètre K) est non négligeable, dans le cas où l'on considère que Δ puisse être modifié par le processus de quantification.

En effet, d'après l'équation (4.39), $\frac{\partial \tilde{H}}{\partial K} = H_1 \circledast H_4$, et H_4 est fonction de Δ. On a dans ce cas $M_{L_2}^W = 1.8680e{+}5$.

On s'intéresse alors à l'opérateur δ-modifié défini par

$$\delta_m \triangleq \frac{q-c}{\Delta} \qquad \text{avec } 0 < c \leqslant 1 \tag{5.22}$$

et présenté au paragraphe 2.1.7.2. La valeur de c est susceptible de modifier la sensibilité.
Une réalisation avec cet opérateur s'écrit

$$Z = \begin{pmatrix} -I_n & A_{\delta_m} & B_{\delta_m} \\ \Delta I_n & cI_n & 0 \\ 0 & C_{\delta_m} & D_{\delta_m} \end{pmatrix} \tag{5.23}$$

Si l'on choisit la valeur de c comme étant exactement implémentée, par exemple comme une valeur comprise entre 2^{-8} et 1, avec un pas de 2^{-8}, on peut examiner les 256 réalisations construites, et comparer leur sensibilité paramétrique. La figure 5.1 donne la sensibilité paramétrique $M_{L_2}^W$ en fonction de c (pour c compris entre $\frac{150}{256}$ et 1).

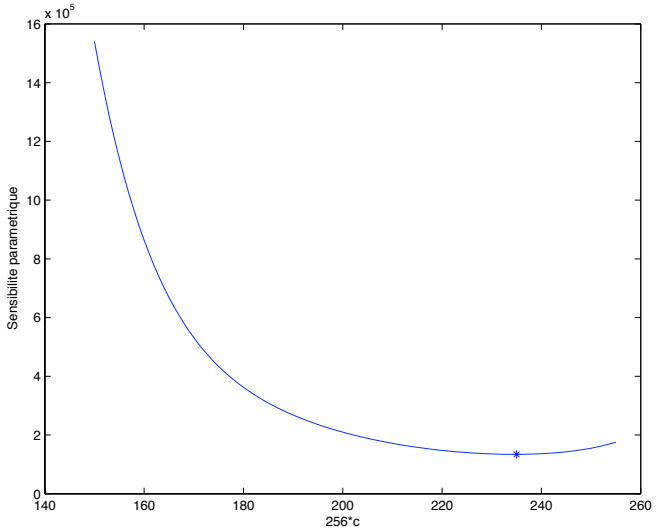


FIG. 5.1 – Les différentes sensibilités paramétriques en fonction de c

On constate un minimum de $1.3414e{+}5$ pour $c = \frac{235}{256}$, différent de 1.

Cette réalisation optimale s'écrit

$$Z_{opt} = \left(\begin{array}{cccc:cccc:c} -1 & 0 & 0 & 0 & \mathbf{-4.760e{+}0} & \mathbf{-1.723e{+}1} & \mathbf{-1.895e{+}1} & \mathbf{9.055e{-}1} & 1 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ \hdashline \mathbf{1.621e{-}1} & 0 & 0 & 0 & 9.180e-01 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1.621e{-}1} & 0 & 0 & 0 & 9.180e-01 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1.621e{-}1} & 0 & 0 & 0 & 9.180e-01 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1.621e{-}1} & 0 & 0 & 0 & 9.180e-01 & 0 \\ \hdashline 0 & 0 & 0 & 0 & \mathbf{2.450e{+}1} & \mathbf{2.913e{+}2} & \mathbf{1.397e{+}3} & \mathbf{2.447e{+}3} & 1 \end{array}\right)$$

et a pour sensibilité

$$\left.\frac{\delta \tilde{H}}{\delta Z}\right|_{Z_{opt}} \times W_{Z_0} \left(\begin{array}{cccc:cccc:c} 0 & 0 & 0 & 0 & \mathbf{5.543e{+}1} & \mathbf{1.459e{+}2} & \mathbf{4.061e{+}2} & \mathbf{1.151e{+}3} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hdashline \mathbf{3.240e{+}1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{3.538e{+}1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{9.231e{+}1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{3.606e{+}1} & 0 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 & \mathbf{1.045e{+}0} & \mathbf{1.425e{+}0} & \mathbf{2.777e{+}0} & \mathbf{6.763e{+}0} & 0 \end{array}\right)$$

Ainsi, l'opérateur δ-modifié peut permettre d'obtenir une mesure de sensibilité paramétrique plus faible, au prix d'un calcul supplémentaire.

5.3 Cas où $\mathscr{R}_H^{\mathscr{S}}$ est infini

Dans de nombreux cas, l'ensemble des réalisations équivalentes structurées selon $\mathscr{S}$ est infini. Nous avons vu, au chapitre 3.3, qu'il était possible de formaliser les classes d'équivalence de réalisations grâce au *Principe d'Inclusion*, afin d'expliciter, $\mathscr{R}_H^{\mathscr{S}}$, en vue de le parcourir.

5.3.1 Similarité sur Z

Nous avons mentionné, à la proposition 3.7 qu'il était possible de caractériser un sous-ensemble de réalisations équivalentes à partir d'une réalisation initiale, et d'une similarité sur la matrice Z (paragraphe 3.3.3.3).
On considère donc une réalisation initiale $\mathcal{R}_0 := (Z_0, l, m, n, p)$ et $\mathcal{R}_1 := (Z_1, l, m, n, p)$ une réalisation équivalente à $\mathcal{R}_0$ telle que

$$Z_1 = \mathcal{T}_1 Z_0 \mathcal{T}_2 \tag{5.24}$$

avec

$$\mathcal{T}_1 = \begin{pmatrix} \mathcal{Y} & & \\ & \mathcal{U}^{-1} & \\ & & I_p \end{pmatrix} \quad \text{et} \quad \mathcal{T}_2 = \begin{pmatrix} \mathcal{W} & & \\ & \mathcal{U} & \\ & & I_m \end{pmatrix} \tag{5.25}$$

Le résultat suivant permet de faciliter les calculs des mesures du chapitre 4,en reliant entre elles les différentes sensibilités :

Proposition 5.3
On considère les réalisations $\mathcal{R}_0 := (Z_0, l, m, n, p)$ *et* $\mathcal{R}_1 := (Z_1, l, m, n, p)$ *reliées par l'équation (5.24). En reprenant les notations des théorèmes 4.6, 4.8 et 4.10, on a alors*

$$\left.\frac{\partial \tilde{H}}{\partial Z}\right|_{Z_1} = \left(\mathcal{T}_1^{-\top} \otimes I_p\right) \left.\frac{\partial \tilde{H}}{\partial Z}\right|_{Z_0} \left(\mathcal{T}_2^{-\top} \otimes I_m\right) \tag{5.26}$$

$$\left.\frac{\partial \bar{H}}{\partial Z}\right|_{Z_1} = \left(\mathcal{T}_1^{-\top} \otimes I_p\right) \left.\frac{\partial \bar{H}}{\partial Z}\right|_{Z_0} \left(\mathcal{T}_2^{-\top} \otimes I_m\right) \tag{5.27}$$

$$\left.\frac{\partial |\lambda_k|}{\partial Z}\right|_{Z_1} = \mathcal{T}_1^{-\top} \left.\frac{\partial |\lambda_k|}{\partial Z}\right|_{Z_0} \mathcal{T}_2^{-\top} \tag{5.28}$$

Démonstration :
On remarque que

$$\left(H_3|_{Z_1} \quad H_1|_{Z_1} \quad I_p\right) = \left(H_3|_{Z_0} \quad H_1|_{Z_0} \quad I_p\right) \mathcal{T}_1^{-1}$$

$$\begin{pmatrix} H_4|_{Z_1} \\ H_2|_{Z_1} \\ I_m \end{pmatrix} = \mathcal{T}_2^{-1} \begin{pmatrix} H_4|_{Z_0} \\ H_2|_{Z_0} \\ I_m \end{pmatrix}$$

d'où, d'après la proposition A.10 et l'équation (4.39)

$$\left.\frac{\partial H}{\partial Z}\right|_{Z_1} = \left(\mathcal{T}_1^{-\top} \otimes I_p\right) \left.\frac{\partial H}{\partial Z}\right|_{Z_0} \left(\mathcal{T}_2^{-\top} \otimes I_m\right)$$

Enfin, il suffit de constater que $\frac{\partial D}{\partial Z}|_{Z_1} = \frac{\partial D}{\partial Z}|_{Z_0}$
De la même manière, on remarque que

$$\bar{H}_1|_{Z_1} = \bar{H}_1|_{Z_0} \begin{pmatrix} I_{n_p} & \\ & \mathcal{U} \end{pmatrix} \text{ et } \bar{H}_2|_{Z_1} = \begin{pmatrix} I_{n_p} & \\ & \mathcal{U} \end{pmatrix}^{-1} \bar{H}_2|_{Z_0}$$

et

$$\bar{M}_1|_{Z_1} = \begin{pmatrix} I_{n_p} & \\ & \mathcal{U} \end{pmatrix}^{-1} \bar{M}_1|_{Z_0} \mathcal{T}_1^{-1}$$

$$\bar{M}_2|_{Z_1} = \mathcal{T}_2^{-1} \bar{M}_2|_{Z_0} \begin{pmatrix} I_{n_p} & \\ & \mathcal{U} \end{pmatrix}$$

$$\bar{M}_3|_{Z_1} = \mathcal{T}_2^{-1} \bar{M}_3|_{Z_0} \begin{pmatrix} I_{n_p} & \\ & \mathcal{U} \end{pmatrix}$$

d'où

$$\begin{aligned}\left.\frac{\partial \bar{H}}{\partial Z}\right|_{Z_1} &= \left(\bar{H}_1|_{Z_0}\ \bar{M}_1|_{Z_0}\ \mathcal{T}_1^{-1}\right) \circledast \left(\mathcal{T}_2^{-1}\left(\bar{M}_2|_{Z_0}\ \bar{H}_2|_{Z_0} + \bar{M}_3|_{Z_0}\right)\right) \\ &= \left(\mathcal{T}_1^{-\top} \otimes I_p\right) \left.\frac{\partial \bar{H}}{\partial Z}\right|_{Z_0} \left(\mathcal{T}_2^{-\top} \otimes I_m\right)\end{aligned}$$

Enfin, on notera que, les réalisations $\mathcal{R}_0$ et $\mathcal{R}_1$ étant équivalentes, les valeurs propres du système bouclé sont les mêmes pour les deux réalisations : $\lambda_k(Z_0) = \lambda_k(Z_1) \quad \forall k$. On a aussi

$$\bar{A}(Z_1) = \begin{pmatrix} I_{n_p} & \\ & \mathcal{U} \end{pmatrix}^{-1} \bar{A}(Z_0) \begin{pmatrix} I_{n_p} & \\ & \mathcal{U} \end{pmatrix}$$

d'où

$$\left.\frac{\partial |\lambda_k|}{\partial \bar{A}}\right|_{Z_1} = \begin{pmatrix} I_{n_p} & \\ & \mathcal{U} \end{pmatrix}^{\top} \left.\frac{\partial |\lambda_k|}{\partial \bar{A}}\right|_{Z_0} \begin{pmatrix} I_{n_p} & \\ & \mathcal{U} \end{pmatrix}^{-\top}$$

et

$$\left.\frac{\partial |\lambda_k|}{\partial Z}\right|_{Z_1} = \mathcal{T}_1^{-\top} \left.\frac{\partial |\lambda_k|}{\partial Z}\right|_{Z_0} \mathcal{T}_2^{-\top} \tag{5.29}$$

grâce aux relations entre $\bar{M}_1|_{Z_1}$ et $\bar{M}_1|_{Z_0}$, $\bar{M}_2|_{Z_1}$ et $\bar{M}_2|_{Z_0}$, et l'équation (4.90). ■

5.3.2 Recherche de l'optimalité dans le cadre de la représentation classique, en q ou en δ

Nous nous intéresserons d'abord aux réalisations structurées selon la représentation *d'état* classique. Il s'agit du problème initial posé par les premiers travaux sur la recherche de réalisations optimales en précision finie (par C. Mullis et R. Roberts [82], V. Tavşanoğlu et L. Thiele [100], etc.).
En considérant un système initial (A_0, B_0, C_0, D_0), les systèmes équivalents (de même dimension) sont décrits par $(T^{-1}A_0T, T^{-1}B_0, C_0T, D_0)$ où T est une matrice non singulière.
Le problème s'écrit

$$T_{opt} = \underset{T \text{ tq } \det(T)\neq 0}{arg\ min}\ \mathcal{J}(T^{-1}A_0T, T^{-1}B_0, C_0T, D_0) \tag{5.30}$$

où $\mathcal{J}$ est la mesure choisie pour le problème. Le système optimal est donné par $(T_{opt}^{-1}A_0T_{opt}, T_{opt}^{-1}B_0, C_0T_{opt}, D_0)$.

Ce problème *historique* se reformule au moyen de la forme implicite spécialisée en un problème de recherche de réalisation q-structurée optimale

$$\mathcal{R}_{opt} = \underset{\mathcal{R}\in\mathscr{R}_H^{\mathscr{S}_q}}{\arg\min}\ \mathcal{J}(\mathcal{R}) \tag{5.31}$$

où $\mathcal{R} \in \mathscr{R}_H^{\mathscr{I}_q}$ peut être parcouru par une similarité sur la matrice Z_0 de la réalisation q-structurée initiale $\mathcal{R}_0 := (Z_0, l, m, n, p)$:

$$\mathscr{R}_H^{\mathscr{I}_q} = \left\{ \mathcal{R} := (Z, l, m, n, p) \;\middle\backslash\; \begin{array}{l} Z = \begin{pmatrix} I_l & & \\ & \mathcal{U}^{-1} & \\ & & I_p \end{pmatrix} Z_0 \begin{pmatrix} I_l & & \\ & \mathcal{U} & \\ & & I_m \end{pmatrix} \\ \forall \mathcal{U} \in \mathbb{R}^{n \times n} \text{ inversible} \end{array} \right\}$$

Il est également possible de considérer une structuration en δ, et de rechercher la réalisation δ-structurée optimale, au sens d'un critère choisi :

$$\mathscr{R}_H^{\mathscr{I}_\delta} = \left\{ \mathcal{R} := (Z, n, m, n, p) \;\middle\backslash\; \begin{array}{l} Z = \begin{pmatrix} \mathcal{U}^{-1} & & \\ & \mathcal{U}^{-1} & \\ & & I_p \end{pmatrix} Z_0 \begin{pmatrix} \mathcal{U} & & \\ & \mathcal{U} & \\ & & I_m \end{pmatrix} \\ \forall \mathcal{U} \in \mathbb{R}^{n \times n} \text{ inversible} \end{array} \right\}$$

De manière généralisée, tous les problèmes de réalisation structurée optimale s'écrivent simplement grâce à la forme implicite spécialisée.

5.3.2.1 Algorithme d'optimisation

Afin de résoudre ces problèmes d'optimisation, et puisque le critère $\mathcal{J}$ utilisé peut-être non convexe (comme pour la mesure de stabilité), un algorithme d'optimisation globale, tel que le *Recuit Simulé Adapatif* (ASA : *Adaptive Simulated Annealing*) de L. Ingber [51, 18] a été utilisé ici, comme dans de nombreux problèmes de recherche de réalisation optimale en précision finie ([118], [19], etc.).

Sur cette base, nous avons mis en œuvre, sous Matlab, une *boîte à outils*[3] permettant de parcourir l'ensemble des réalisations structurées équivalentes au moyen des relations de similarité et de chercher à l'aide d'une version de ASA pour Matlab [92] la meilleure réalisation.

En reprenant l'exemple du paragraphe 4.2.3, on obtient ainsi les réalisations $\mathcal{R}_{opt}^q$ et $\mathcal{R}_{opt}^\delta$ données par :

$$Z_{opt}^q = \left(\begin{array}{cccc} & & & \\ \hline \mathbf{6.172e{-}1} & \mathbf{-3.552e{-}1} & \mathbf{-3.642e{-}2} & \mathbf{2.266e{-}1} \\ \mathbf{3.403e{-}1} & \mathbf{5.681e{-}1} & \mathbf{-3.650e{-}1} & \mathbf{4.375e{-}1} \\ \mathbf{-4.098e{-}2} & \mathbf{4.117e{-}1} & \mathbf{7.895e{-}1} & \mathbf{4.221e{-}1} \\ \hline \mathbf{1.644e{-}1} & \mathbf{-3.833e{-}1} & \mathbf{4.969e{-}1} & \mathbf{0} \end{array} \right)$$

$$\left.\frac{\delta \tilde{H}}{\delta Z}\right|_{Z_{opt}^q} = \left(\begin{array}{cccc} & & & \\ \hline \mathbf{1.586e{-}1} & \mathbf{3.285e{-}1} & \mathbf{4.526e{-}1} & \mathbf{3.409e{-}1} \\ \mathbf{3.387e{-}1} & \mathbf{7.890e{-}1} & \mathbf{9.017e{-}1} & \mathbf{6.747e{-}1} \\ \mathbf{4.283e{-}1} & \mathbf{9.034e{-}1} & \mathbf{1.457e{+}0} & \mathbf{9.166e{-}1} \\ \hline \mathbf{3.617e{-}1} & \mathbf{6.775e{-}1} & \mathbf{9.065e{-}1} & \mathbf{0} \end{array} \right)$$

[3] La boîte à outils permet aussi d'utiliser la fonction `fminsearch` de Matlab, qui aboutit parfois sur un optimum local.

et

$$Z^{\delta}_{opt} = \left(\begin{array}{ccc:cccc} -1 & 0 & 0 & \mathbf{-7.945e{-}1} & \mathbf{6.418e{-}1} & \mathbf{-2.389e{-}1} & \mathbf{-9.477e{-}2} \\ 0 & -1 & 0 & \mathbf{-6.662e{-}1} & \mathbf{-8.509e{-}1} & \mathbf{-7.235e{-}1} & \mathbf{6.983e{-}1} \\ 0 & 0 & -1 & \mathbf{3.706e{-}1} & \mathbf{7.843e{-}1} & \mathbf{-4.049e{-}1} & \mathbf{5.696e{-}1} \\ \hdashline 0.5 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0 & 1 & 0 \\ \hdashline 0 & 0 & 0 & \mathbf{-4.310e{-}1} & \mathbf{-3.958e{-}1} & \mathbf{6.919e{-}1} & \mathbf{0} \end{array}\right)$$

$$\left.\frac{\delta \tilde{H}}{\delta Z}\right|_{Z^{\delta}_{opt}} = \left(\begin{array}{ccc:cccc} 0 & 0 & 0 & \mathbf{9.086e{-}2} & \mathbf{1.998e{-}1} & \mathbf{2.161e{-}1} & \mathbf{2.924e{-}1} \\ 0 & 0 & 0 & \mathbf{1.856e{-}1} & \mathbf{3.534e{-}1} & \mathbf{4.651e{-}1} & \mathbf{4.470e{-}1} \\ 0 & 0 & 0 & \mathbf{2.730e{-}1} & \mathbf{4.244e{-}1} & \mathbf{7.284e{-}1} & \mathbf{6.468e{-}1} \\ \hdashline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & \mathbf{2.513e{-}1} & \mathbf{4.802e{-}1} & \mathbf{6.403e{-}1} & \mathbf{0} \end{array}\right)$$

On obtient les mesures suivantes, que l'on pourra comparer à celles du tableau 4.2 :

Réalisation	sensibilité $M^W_{L_2}$
Réalisation optimale en q (Z^q_{opt})	7.8337
Réalisation optimale en δ (Z^{δ}_{opt})	2.6631

TAB. 5.6 – Mesure de sensibilité $M^W_{L_2}$ optimale structurée en q ou en δ

Le résultat est légèrement amélioré par rapport à la réalisation équilibrée.

5.3.3 Espace d'état avec paramètre implicite E

Nous avons évoqué, au paragraphe 3.1.2, la possibilité d'écrire le calcul de $X(k+1)$ sous une forme implicite

$$EX(k+1) = KT(k+1) + PX(k) + QU(k) \tag{5.32}$$

avec E une matrice triangulaire inférieure avec des 1 sur la diagonale, et non pas nécessairement la matrice identité pour offrir une plus grande paramétrisation possible.
Si on se restreint à une structuration *d'état* classique, auquel on rajoute ce terme implicite, nous arrivons à la réalisation :

$$\left(\begin{array}{c:c:c} E & 0 & 0 \\ \hdashline -I_n & I_n & 0 \\ \hdashline 0 & 0 & I_p \end{array}\right) \begin{pmatrix} T(k+1) \\ X(k+1) \\ Y(k) \end{pmatrix} = \left(\begin{array}{c:c:c} 0 & A & B \\ \hdashline 0 & 0 & 0 \\ \hdashline 0 & C & D \end{array}\right) \begin{pmatrix} T(k) \\ X(k) \\ U(k) \end{pmatrix} \tag{5.33}$$

Si on note Ω^n_E l'ensemble des matrices de dimension n, triangulaires inférieures avec une diagonale unitaire, il est possible de décrire l'ensemble des

réalisations équivalentes grâce à la similarité

$$Z = \begin{pmatrix} \mathcal{Y} & & \\ & \mathcal{U}^{-1} & \\ & & I_p \end{pmatrix} Z_0 \begin{pmatrix} \mathcal{U} & & \\ & \mathcal{U} & \\ & & I_m \end{pmatrix} \tag{5.34}$$

où $\mathcal{Y}$ est telle que $\mathcal{Y}E\mathcal{U} \in \Omega_E^n$.
Si l'on choisit une réalisation initiale $\mathcal{R}_0 := (Z_0)$ avec

$$Z_0 = \begin{pmatrix} -I_n & A_0 & B_0 \\ I_n & 0 & 0 \\ 0 & C_0 & D_0 \end{pmatrix} \tag{5.35}$$

il nous faut choisir, d'un point de vue pratique, les coefficients de E et ceux de $\mathcal{U}$ comme paramètres d'optimisation , et construire $\mathcal{Y}$ par $\mathcal{Y} = E\mathcal{U}^{-1}$.

En considérant de nouveau le même exemple, nous obtenons, après optimisation, la réalisation suivante :

$$Z_{opt} = \left(\begin{array}{ccc:ccc:c} -1 & 0 & 0 & \mathbf{7.058e{-1}} & \mathbf{-3.764e{-1}} & \mathbf{3.579e{-2}} & \mathbf{-2.780e{-1}} \\ \mathbf{8.314e{-1}} & -1 & 0 & \mathbf{1.763e{-1}} & \mathbf{8.807e{-1}} & \mathbf{-6.786e{-2}} & \mathbf{5.393e{-1}} \\ \mathbf{1.999e{+0}} & \mathbf{1.271e{+0}} & -1 & \mathbf{-2.230e{+0}} & \mathbf{-4.824e{-1}} & \mathbf{6.782e{-1}} & \mathbf{6.082e{-1}} \\ \hdashline 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & \mathbf{-3.329e{-1}} & \mathbf{-6.423e{-1}} & \mathbf{4.158e{-1}} & \mathbf{0} \end{array}\right)$$

qui a pour matrice de sensibilité paramétrique

$$\left.\frac{\delta \tilde{H}}{\delta Z}\right|_{Z_{opt}} \times W_Z = \left(\begin{array}{ccc:ccc:c} 0 & 0 & 0 & \mathbf{5.518e{-1}} & \mathbf{6.315e{-1}} & \mathbf{4.505e{-1}} & \mathbf{6.963e{-1}} \\ \mathbf{5.066e{-1}} & 0 & 0 & \mathbf{5.066e{-1}} & \mathbf{5.167e{-1}} & \mathbf{5.152e{-1}} & \mathbf{5.415e{-1}} \\ \mathbf{4.174e{-1}} & \mathbf{4.085e{-1}} & 0 & \mathbf{4.174e{-1}} & \mathbf{4.085e{-1}} & \mathbf{6.305e{-1}} & \mathbf{5.732e{-1}} \\ \hdashline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & \mathbf{5.137e{-1}} & \mathbf{5.878e{-1}} & \mathbf{6.955e{-1}} & \mathbf{0} \end{array}\right)$$

et 5.2264 pour mesure $M_{L_2}^W$ (à comparer avec 7.8337 pour la réalisation optimale avec E égal à l'identité, par exemple).
On remarquera que nous avons pris aussi en considération la sensibilité paramétrique des coefficients triangulaires inférieurs de E, ce que nous aurions pu éviter, afin d'obtenir une mesure de sensibilité encore plus faible, en ne considérant, pour E que des coefficients triviaux (comme pour la valeur de c du paragraphe 5.2.3).

5.4 Discussion

Nous avons illustré, dans les paragraphes précédents, quelques possibilités d'obtenir des réalisations optimisées, selon un critère donné (ici une mesure de sensibilité paramétrique ou une mesure de stabilité) et sous contraintes

d'ordre ou de structuration (décomposition en cascade, forme d'état, opérateur δ, réalisation avec terme implicite, etc.). Nous discutons ici du choix du critère et de la structure, c'est à dire de la méthode à adopter en pratique et des perfectionnements à inclure.

5.4.1 Choix du critère

Nous avons envisagé, dans cette thèse, trois critères permettant d'évaluer l'impact de la quantification des coefficients (en tenant en compte de la nature de la représentation de ces coefficients : virgule fixe ou flottante, blocs, etc.). D'autres devraient être considérés, tels que, bien entendu, les bruits de quantification (en 1^{re} approche ou de manière plus fine avec une description augmentée du mode opératoire des opérations dans le calculateur), mais aussi la sensibilité des critères de régulation (marge de phase, de gain, ...) ou de validation (dépassement pour une réponse à un échelon [94], norme L_∞ de l'erreur, ...).
Le choix du critère doit se faire bien entendu en fonction des spécifications initiales du filtre ou du régulateur.

Des critères mixtes – ou une optimisation multicritère [104] – peuvent être également mis en place afin de tenir compte simultanément de différents indices de dégradation. Il est à noter cependant que différents critères, tels que la sensibilité paramétrique, le bruit de quantification, etc., vont peu ou prou "dans le même sens".

5.4.2 Choix de la structure

Nous avons pu montrer de nombreuses structurations possibles, toutes exprimables avec la forme implicite spécialisée, et ainsi considérer un éventail de réalisations bien plus larges que la forme d'état classiquement utilisée, ouvrant des possibilités nombreuses dans la recherche de *meilleures* réalisations.
Toutes ces réalisations diffèrent principalement par le coût de calcul qui leur est associé (nous avons vu à la section 4.1.2 que ce coût est fonction du nombre de coefficients non triviaux utilisés pour le calcul).

La recherche d'un compromis résilience/coût de calcul n'est pas, *a priori*, une chose aisée. Une première voie consiste à n'envisager que des structurations parcimonieuses lors de l'optimisation. On pourra choisir par exemple la forme modale (décomposition en cascade ou parallèle qui présente un faible coût de calcul),en q ou δ et avec ou sans utilisation d'une matrice E différente de l'identité afin d'obtenir une résilience améliorée, au prix d'un coût de calcul légèrement augmenté.
Nous n'avons pas encore, à ce stade, arrêté une méthodologie définitive dans

la recherche de ce compromis. D'autres formes de réalisations sont intéressantes à étudier, notamment en mixant les réalisations en q et δ, avec un paramètre implicite, en décomposant en cascade/parallèle, etc.

Une première méthodologie peut consister en l'examen de quelques structurations que l'on sait réaliser un bon compromis résilience/coût de calcul, et en adaptant selon les besoins : le coût de calcul maximal admissible peut être un critère de choix pour restreindre le champs des possibilités. Une seconde voie consiste à *creuser* les réalisations *a posteriori* comme indiqué dans la section suivante.

5.4.3 Réalisations creuses

Une autre possibilité pour améliorer le compromis résilience/coût de calcul est de considérer une réalisation dans une structuration donnée et de trouver, parmi les réalisations équivalentes de même structuration, celles qui possèdent d'avantage de paramètres triviaux.
Cette technique de *creusement de matrices* est détaillée dans [54, 114, 118] (elle a été développée uniquement pour la forme d'état classique et pour la mesure de stabilité) et consiste, à partir d'une réalisation optimale mais non creuse, à trouver le coefficient le plus proche d'un paramètre trivial et à appliquer un changement de base qui rapproche le plus possible ce coefficient du paramètre trivial, tout en garantissant que le critère n'évolue que très faiblement. En réitérant de la sorte, il est possible d'obtenir une transformation changeant certains coefficients en paramètres triviaux sans trop élever le critère.

Il est naturellement possible d'appliquer ce principe à la forme implicite spécialisée et généraliser ainsi le cadre d'application de l'algorithme proposé dans [54, 114, 118] :

Étape 0 : On considère une réalisation Z_0. On note

$$Z(\mathcal{T}) = \begin{pmatrix} \mathcal{Y} & & \\ & \mathcal{U}^{-1} & \\ & & I_p \end{pmatrix} Z_0 \begin{pmatrix} \mathcal{W} & & \\ & \mathcal{U} & \\ & & I_m \end{pmatrix} \tag{5.36}$$

où $\mathcal{T} = \begin{pmatrix} \mathcal{U} & \mathcal{W} & \mathcal{Y} \end{pmatrix}$ On considère aussi une mesure $\mathcal{J}$, ainsi qu'une réalisation Z_1 qui minimise $\mathcal{J}$.
On fixe $\tau = 10^{-5}$ et $\epsilon = 10^{-10}$ par exemple.

Étape 1 : On trouve les paramètres $(\eta_i)_{1 \leqslant i \leqslant r}$ les éléments triviaux de Z, ainsi que ξ le coefficient non trivial de Z qui est le plus proche d'un élément trivial (un nombre sera donné trivial s'il est proche de 0, -1 ou 1 à ϵ près).

Étape 2 : On cherche $\mathcal{S}$ tels que

i) $\mathcal{J}(Z(\mathcal{T} + \tau\mathcal{S}))$ est proche de $\mathcal{J}(Z(\mathcal{T}))$

ii) les éléments triviaux $(\eta_i)_{1\leqslant i\leqslant r}$ restent inchangés pour $Z(\mathcal{T}+\tau\mathcal{S})$

iii) ξ est rapproché le plus possible d'un élément trivial

iv) $\|\mathcal{S}\|_F = 1$

Si $\mathcal{S}$ n'existe pas, alors $\mathcal{T}_{sopt} = \mathcal{T}$, l'algorithme se termine, et la solution sous-optimale est $\mathcal{T}_{sopt}$.

Étape 3 : $\mathcal{T} \leftarrow \mathcal{T} + \tau\mathcal{S}$. Si ξ est devenu trivial, on va à l'étape 1, sinon à l'étape 2.

De plus, $\mathcal{S}$ s'obtient de la manière suivante :
Pour τ très faible, la condition i) s'écrit

$$\left(\mathrm{Vec}\left(\frac{\partial\mathcal{J}}{\partial\mathcal{T}}\right)\right)^\top \mathrm{Vec}(\mathcal{S}) = 0 \tag{5.37}$$

Et la condition ii) signifie que

$$\left(\mathrm{Vec}\left(\frac{\partial\eta_i}{\partial\mathcal{T}}\right)\right)^\top \mathrm{Vec}(\mathcal{S}) = 0 \qquad \forall 1 \leqslant i \leqslant r \tag{5.38}$$

Ainsi, si on définit la matrice E par

$$E \triangleq \begin{pmatrix} \left(\mathrm{Vec}\left(\frac{\partial\mathcal{J}}{\partial\mathcal{T}}\right)\right)^\top \\ \left(\mathrm{Vec}\left(\frac{\partial\eta_1}{\partial\mathcal{T}}\right)\right)^\top \\ \cdots \\ \left(\mathrm{Vec}\left(\frac{\partial\eta_r}{\partial\mathcal{T}}\right)\right)^\top \end{pmatrix} \tag{5.39}$$

alors $\mathrm{Vec}(\mathcal{S})$ doit appartenir au noyau de E, noté $\mathcal{N}(E)$ (construit grâce au théorème de Fredholm).
Si $\mathcal{N}(E)$ est vide, alors $\mathcal{S}$ n'existe pas, et l'algorithme prend fin. Sinon, on peut obtenir une base $(b_i)_{1\leqslant i\leqslant t}$ de $\mathcal{N}(E)$. La condition iii), qui demande de déplacer le coefficient ξ le plus proche d'un élément trivial, nous fait choisir $\mathrm{Vec}(\mathcal{S})$ comme la projection de $\mathrm{Vec}\left(\frac{\partial\xi}{\partial\mathcal{T}}\right)$ sur $\mathcal{N}$.
Avec la condition iv), nous pouvons construire $\mathrm{Vec}(\mathcal{S})$ selon

$$a_i = b_i^\top \mathrm{Vec}\left(\frac{\partial\xi}{\partial\mathcal{T}}\right) \qquad \forall 1 \leqslant i \leqslant t \tag{5.40}$$

$$w = \sum_{i=1}^{t} a_i b_i \tag{5.41}$$

$$\mathrm{Vec}(\mathcal{S}) = \pm\frac{w}{\sqrt{w^\top w}} \tag{5.42}$$

Le signe de l'équation (5.42) est choisi pour rapprocher ξ de l'élément trivial choisi.

Les calculs de cet algorithme décrit ici pour la forme implicite ne sont pas développés, mais $\frac{\partial \mathcal{J}}{\partial \mathcal{I}}$ peut s'obtenir à partir des équations de la proposition 5.3, tandis que la dérivée d'un élément $Z_{r,s}$ par rapport à $\mathcal{T}$ s'écrit :

$$\frac{\partial Z_{r,s}}{\partial \mathcal{Y}} = \left(e_r^\top \begin{pmatrix} I_l \\ 0 \\ 0 \end{pmatrix}\right) \circledast \left(\begin{pmatrix} I_l & 0 & 0 \end{pmatrix} Z_0 U_2 e_s \right) \tag{5.43}$$

$$\frac{\partial Z_{r,s}}{\partial \mathcal{U}} = \left(e_r^\top U_1 Z_0 \begin{pmatrix} 0 \\ I_l \\ 0 \end{pmatrix}\right) \circledast \left(\begin{pmatrix} 0 & I_l & 0 \end{pmatrix} e_s \right) \tag{5.44}$$

$$- \left(e_r^\top \begin{pmatrix} 0 \\ \mathcal{U}^{-1} \\ 0 \end{pmatrix}\right) \circledast \left(\begin{pmatrix} 0 & \mathcal{U}^{-1} & 0 \end{pmatrix} Z_0 U_2 e_s \right) \tag{5.45}$$

$$\frac{\partial Z_{r,s}}{\partial \mathcal{W}} = \left(e_r^\top U_1 Z_0 \begin{pmatrix} I_l \\ 0 \\ 0 \end{pmatrix}\right) \circledast \left(\begin{pmatrix} I_l & 0 & 0 \end{pmatrix} e_s \right) \tag{5.46}$$

où $U_1 = \begin{pmatrix} \mathcal{Y} & & \\ & \mathcal{U}^{-1} & \\ & & I_p \end{pmatrix}$ et $U_2 = \begin{pmatrix} \mathcal{W} & & \\ & \mathcal{U} & \\ & & I_m \end{pmatrix}$
(ces équations sont déduites de l'égalité $Z_{r,s} = e_r^\top Z e_s$ et de l'équation (5.36)).

Remarque 1 : Pour être valables, y compris lorsque $\mathcal{U}$, $\mathcal{W}$ et $\mathcal{Y}$ ne sont pas indépendants (comme c'est par exemple le cas pour les structurations particulières), ces calculs mériteraient d'être réécrits dans ce cadre plus général.
Remarque 2 : à l'étape 1,pour le choix de ξ, la sensibilité vis-à-vis de ce paramètre pourrait aussi être pris en compte (pour rendre trivial un paramètre dont la sensibilité est forte). Cette idée pourrait permettre d'améliorer le compromis résilience/coût de calcul.

5.5 Application au domaine automobile

Afin d'examiner et de comparer toutes ces possibilités sur un seul et même exemple, nous avons considéré le cas d'un régulateur assurant le contrôle actif des oscillations longitudinales d'un véhicule.
Ce régulateur, développé par D. Lefebvre (PSA Peugeot-Citroën) [65, 64, 66] dans le cadre d'une étude sur l'amélioration de l'agrément de conduite, atténue les oscillations longitudinales du véhicule (de 0 à 10Hz), apparaissant en phase de transition lorsqu'une forte demande de couple est sollicitée :

- à la suite d'une accélération brutale (production importante de couple moteur) ;

– d'une décélération brutale (annulation de couple moteur) ;
– ou d'un engagement brutal de l'embrayage lors d'un changement de rapport descendant ou montant.

Le régulateur agit sur le couple moteur à partir de la mesure du régime moteur (cf. figure 5.2).

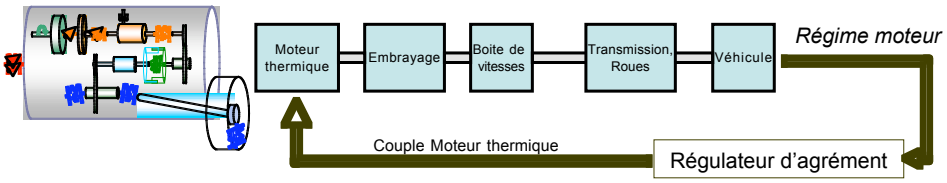


FIG. 5.2 – Principe d'action du régulateur d'oscillations longitudinales

Le châssis à été modélisé en temps continu, et un régulateur continu H_∞-optimal a été synthétisé [64]. Le modèle discret est donné sous forme d'état (A_p, B_p, C_p) (équations (5.47) et (5.48)) et le régulateur discret par (A, B, C, D) (équations (5.49) et (5.50))

$$A_p = \begin{pmatrix} 8.384e{-1} & 1.600e{-1} & -3.294e{-1} & -4.833e{-2} & 0 & 0 & 0 & 0 & 0 & 0 \\ -3.927e{-1} & 7.144e{-1} & 5.040e{-2} & -8.245e{-3} & 0 & 0 & 0 & 0 & 0 & 0 \\ -1.566e{-1} & -6.105e{-1} & 3.683e{-2} & 4.195e{-1} & 0 & 0 & 0 & 0 & 0 & 0 \\ -1.444e{-1} & 1.772e{-1} & -6.798e{-1} & 6.508e{-1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 1.929e{-1} & 1.512e{-1} & 4.030e{-1} & 3.898e{-1} & 9.773e{-1} & 1.037e{-2} & -6.170e{-2} & 0 & 0 & 0 \\ 2.768e{-4} & 2.170e{-4} & 5.783e{-4} & 5.594e{-4} & 2.837e{-3} & 9.971e{-1} & 1.698e{-2} & 0 & 0 & 0 \\ 3.238e{-2} & 2.539e{-2} & 6.767e{-2} & 6.545e{-2} & 3.320e{-1} & -3.341e{-1} & 9.868e{-1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.000e{+0} & -1.000e{-10} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.000e{-2} & 1.000e{+0} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9.417e{-1} \end{pmatrix} \quad (5.47)$$

$$B_p = \begin{pmatrix} -4.007e{+0} \\ -5.769e{+0} \\ -6.522e{+0} \\ 2.490e{+0} \\ 8.562e{-1} \\ 1.229e{-3} \\ 1.438e{-1} \\ 1.000e{+0} \\ 5.000e{-3} \\ 0 \end{pmatrix}, \quad C_p = \begin{pmatrix} 9.209e{-3} \\ 7.221e{-3} \\ 1.924e{-2} \\ 1.861e{-2} \\ 9.441e{-2} \\ 4.953e{-4} \\ -2.946e{-3} \\ 0 \\ 0 \\ -3.495e{-1} \end{pmatrix}^\top \quad (5.48)$$

$$A = \begin{pmatrix}
8.195e{-1} & 2.812e{-1} & -3.317e{-2} & 2.699e{-2} & -1.649e{-1} & 1.318e{-1} \\
-2.812e{-1} & -4.817e{-1} & -1.668e{-1} & 8.654e{-2} & -5.403e{-1} & 1.469e{-1} \\
3.317e{-2} & -1.668e{-1} & 9.749e{-1} & 1.696e{-2} & -9.104e{-2} & 7.638e{-2} \\
2.699e{-2} & -8.654e{-2} & -1.696e{-2} & 9.601e{-1} & 2.528e{-1} & 5.956e{-2} \\
1.649e{-1} & -5.403e{-1} & -9.104e{-2} & -2.528e{-1} & 6.022e{-1} & 3.888e{-1} \\
1.318e{-1} & -1.469e{-1} & -7.638e{-2} & 5.956e{-2} & -3.888e{-1} & 4.664e{-1} \\
1.059e{-2} & -1.837e{-2} & -3.357e{-3} & 1.654e{-3} & -1.150e{-2} & -6.206e{-2} \\
6.733e{-2} & -1.211e{-1} & -2.006e{-2} & 9.085e{-3} & -6.420e{-2} & -4.224e{-1} \\
-1.750e{-3} & 1.942e{-3} & 8.441e{-4} & -6.046e{-4} & 3.945e{-3} & 8.490e{-4} \\
-6.525e{-5} & 2.134e{-5} & 4.548e{-5} & -3.843e{-5} & 2.454e{-4} & -3.703e{-4}
\end{pmatrix}$$

$$\begin{pmatrix}
1.059e{-2} & -6.733e{-2} & 1.750e{-3} & 6.525e{-5} \\
1.837e{-2} & -1.211e{-1} & 1.942e{-3} & 2.134e{-5} \\
3.357e{-3} & -2.006e{-2} & 8.441e{-4} & 4.548e{-5} \\
1.654e{-3} & -9.085e{-3} & 6.046e{-4} & 3.843e{-5} \\
1.150e{-2} & -6.420e{-2} & 3.945e{-3} & 2.454e{-4} \\
-6.206e{-2} & 4.224e{-1} & -8.490e{-4} & 3.703e{-4} \\
9.832e{-1} & 1.258e{-1} & 7.737e{-3} & 6.392e{-4} \\
-1.258e{-1} & -4.483e{-2} & 7.258e{-2} & 5.631e{-3} \\
-7.737e{-3} & 7.258e{-2} & 9.838e{-1} & -2.474e{-3} \\
-6.392e{-4} & 5.631e{-3} & -2.474e{-3} & 9.418e{-1}
\end{pmatrix} \tag{5.49}$$

$$B = \begin{pmatrix}
-2.372e{+0} \\ -2.540e{+0} \\ -1.210e{-1} \\ -1.565e{-4} \\ -6.245e{-2} \\ 1.151e{+0} \\ 4.083e{-2} \\ 2.255e{-1} \\ -1.528e{-2} \\ -9.720e{-4}
\end{pmatrix}, \; C = \begin{pmatrix}
-2.372e{-2} \\ 2.540e{-2} \\ 1.210e{-3} \\ -1.565e{-6} \\ 6.245e{-4} \\ 1.151e{-2} \\ 4.083e{-4} \\ -2.255e{-3} \\ 1.528e{-4} \\ 9.720e{-6}
\end{pmatrix}^{\top}, \; D = -2.140e{-1} \tag{5.50}$$

Le tableau 5.7, qui ne prétend pas être exhaustif, regroupe les mesures de sensibilité, stabilité et coût de calcul pour diverses réalisations possibles (nous nous restreindrons à la virgule fixe, pour ne pas alourdir le tableau). Il faut noter que les optimisations, qui manipulent ici un vecteur d'optimisation de taille 100, demandent un temps de calcul non négligeable (12h sur un PC de bureau), car l'évaluation de $M_{L_2}^W$ ou $\bar{M}_{L_2}^W$ est coûteuse (norme H_2 d'une matrice de transfert de grande dimension).

- Pour la décomposition en cascade, nous avons repris exactement la forme proposée au paragraphe 5.2.1. Il y a ici 43200 possibilités. Les mesures de sensibilités varient fortement d'une réalisation à l'autre : la sensibilité paramétrique "boucle ouverte varie de 3.563 à 3331, la sensibilité paramétrique "boucle fermée" de 554.7 à $2.444e+7$ et la mesure de stabilité de $5.622e{-4}$ à $2.081e{-16}$. De plus, comme on peut le remarquer sur la figure 5.3, il y a une corrélation assez étroite entre la sensibilité boucle ouverte et la sensibilité boucle fermée, même s'il existe des réalisations avec une bonne sensibilité boucle ouverte et une mauvaise sensibilité boucle fermée, et vice et versa. On peut faire les mêmes remarques avec la mesure de stabilité.
- Pour la mise sous forme retour d'état observateur (cf. paragraphe 5.2.2), nous avons étudié tous les partionnements possibles : 120 réalisations ont été construites : $M_{L_2}^W$ varie de $4.9301e{+04}$ à $1.5168e{+11}$, $\bar{M}_{L_2}^W$ de à et μ de à .

	forme	$M_{L_2}^W$	$\bar{M}_{L_2}^W$	$1/\mu$	Nb d'op.
Réalisations en q	directe II	$5.664e{+}15$	$6.8023e{+}21$	$5.9355e{-}12$	20+ 21×
	équilibrée	4.881	$6.438e{+}4$	$5.1582e{-}5$	110+ 121×
	optimale selon $M_{L_2}^W$	4.840	$6.328e{+}4$	$6.482e{-}5$	110+ 121×
	optimale selon $\bar{M}_{L_2}^W$	5.214	$9.617e{+}4$	$4.933e{-}5$	110+ 121×
	optimale selon μ	4.932	$6.476e{+}4$	$8.236e{-}5$	110+ 121×
Réalisations en cascade	optimale selon $M_{L_2}^W$	3.563	724.0	$5.576e{-}4$	31+ 45×
	optimale selon $\bar{M}_{L_2}^W$	4.743	554.7	$5.399e{-}4$	31+ 45×
	optimale selon μ	15.523	1005.3	$5.622e{-}4$	31+ 45×
Retour d'état Observateur	optimale selon $M_{L_2}^W$	$1.9054e{+}3$	$3.2395e{+}5$	$4.0919e{-}7$	121+ 131×
	optimale selon $\bar{M}_{L_2}^W$	$2.7257e{+}4$	$8.4415e{+}4$	$7.1789e{-}6$	121+ 131×
	optimale selon μ	$2.0244e{+}4$	$1.5982e{+}6$	$1.2566e{-}5$	121+ 131×

TAB. 5.7 – Tableau comparatif de diverses réalisations équivalentes

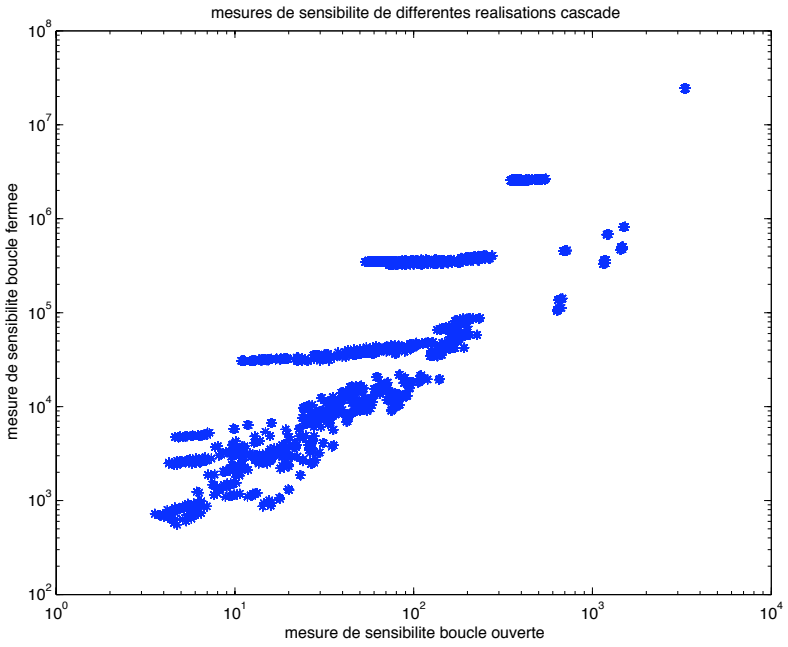


FIG. 5.3 – Sensibilité paramétrique boucle ouverte $M_{L_2}^W$ et boucle fermée $\bar{M}_{L_2}^W$ pour les 43200 réalisations cascade

5.6 Conclusion

Ce chapitre présente quelques possibilités de synthèse de réalisation optimale, notamment autour de différentes possibilités de découpage ou partionnement, ou autour de changements de base pour une structuration donnée : il convient alors de faire appel à un algorithme d'optimisation.
On voit bien ici que le champ des réalisations possibles est très vaste : découpage cascade, parallèle, réalisations optimales en δ, en q, avec un terme implicite ou bien encore en imaginant des réalisations mixtes.
Le coût de calcul associé à chaque réalisation est bien entendu important, et la synthèse de réalisation ne pourra se faire qu'à travers un compromis coût de calcul/résilience.

Conclusion et perspectives

Conclusion

L'implémentation de lois de contrôle-commande, qu'elles soient issues de l'*automatique* ou du *traitement du signal*, dans des calculateurs embarqués est une tâche difficile et complexe, tant les contraintes peuvent être fortes.
En ne considérant que les contraintes d'ordre numérique – les contraintes temporelles, tout aussi importantes, n'ont pas été abordées – la transformation d'un régulateur ou d'un filtre exprimé par une équation d'entrées/sorties, d'une réalisation spécifique ou encore d'une planche Simulink, en un code logiciel s'exécutant sur un calculateur cible, entraîne une dégradation de la loi et de ses caractéristiques fonctionnelles :

- les coefficients exprimant les relations mathématiques sont modifiés par le biais du processus de quantification, inévitable lors de la mise en œuvre des calculs en précision finie et limitée ;
- des bruits de quantification peuvent intervenir à chaque étape de calcul, à chaque opération et à chaque itération.

Et, puisqu'il existe une infinité de possibilités – toutes équivalentes – de décrire mathématiquement une loi donnée et que celles-ci ne le sont plus dès qu'elles sont implémentées en précision finie, la problématique de l'implémentation consiste en la recherche d'une réalisation numérique dont la dégradation est minime.

Après avoir détaillé le contexte industriel de la thèse et les contraintes numériques d'implémentation qui en découlent, nous avons passé en revue différentes réalisations possibles dans le domaine de l'automatique et du trai-

tement du signal, ainsi que les techniques logicielles possibles.
De nombreuses études sur l'impact de la mise en œuvre numérique de certaines de ces possibilités ont été réalisées depuis de nombreuses années – principalement regroupées dans les livres de D. Williamson [111], M. Gevers et G. Li [31] et dernièrement R. Istepanian et J. Whidborne [52] – mais ne couvrent pas l'ensemble des implémentations possibles.
La forme implicite spécialisée, introduite au chapitre 3, permet de regrouper, dans un formalisme macroscopique mais suffisamment proche du code logiciel mis en œuvre, les diverses réalisations classiquement utilisées en automatique et traitement du signal, ainsi que certaines techniques du génie logiciel. Il est ainsi possible d'analyser le comportement en précision finie de ces réalisations et de les comparer, au moyen de mesures qui évaluent leur degré de sensibilité à la quantification. Les critères proposés sont des extensions et améliorations de mesures *précision finie* existantes. De plus, elles prennent en compte, non seulement la réalisation, mais aussi le format de donnée utilisé – virgule fixe ou virgule flottante.

Enfin, nous avons appliqué ces mesures à des exemples académiques et du domaine automobile, à des fins d'analyse mais aussi de synthèse, explorant un large panel de réalisations possibles. Les résultats obtenus permettent et permettront de dégager certaines lignes méthodologiques d'implémentation. Les programmes ayant pour but l'analyse et la synthèse de réalisations en vue de l'implémentation ont été organisés autour d'une *boîte à outils* logicielle disponible sur demande (thibault.hilaire@irccyn.ec-nantes.fr, philippe.chevrel@emn.fr), et ont vocation à assister le concepteur d'une implémentation. Quelques réflexions sur l'organisation chez PSA de la chaîne de développement des lois de contrôle-commande, à l'interface entre l'*automaticien-physicien* et l'*automaticien-informaticien*, ont été consignées dans un document annexe [37].

Perspectives

De par la taille du champ d'investigation de cette thématique transversale qu'est l'implémentation en précision finie, de nombreux thèmes n'ont pu être abordés avec la profondeur qu'ils méritaient, ce qui ouvre de multiples perspectives pour la poursuite de ces travaux.

Tout d'abord, parmi les thèmes qui n'ont été qu'abordés et nécessiteraient davantage d'investigations, on trouve les travaux liés à l'évaluation précise des bruits de quantification, aux méthodologies de compilation sous contrainte de précision, etc. Il conviendrait, à cette fin, d'utiliser une modélisation plus microscopique de l'algorithme de réalisation, afin de tenir plus particulièrement compte de l'architecture matérielle du calculateur.

Afin d'être exhaustif, d'autres réalisations encore pourraient être reconsidérées à l'aide de la forme implicite spécialisée (e.g. les réalisations Q-paramétrée [112]). Des bibliothèques de composants préprogrammés offrant diverses possibilités de compromis (résilience, bruits, coût de calcul, etc.) pourraient être construites. Un ensemble élargi de lois de contrôle-commande, devrait être considéré, telles les lois de commande Linéaires à Paramètres Variants [21] et non-linéaires. Ceci peut se situer dans le prolongement directe de ce qui a été développé ici, en mélangeant des non-linéarités statiques implémentées sous forme de cartographies et un sous-système linéaire dans le cadre d'une représentation LFR.

La démarche de synthèse de réalisations peut également être enrichie en se plaçant dans le cadre d'une optimisation multi-objectifs de manière à atteindre, lorsque cela est nécessaire, le meilleur compromis.

Enfin, la *boîte à outils* proposée mériterait un effort de développement supplémentaire avant de la mettre au service des informaticiens et automaticiens en charge de l'implémentation.

Publications

Publications internationales

- [44] T. Hilaire, P. Chevrel, and Y. Trinquet. Implicit state-space representation : a unifying framework for FWL implementation of LTI systems. In *IFAC05 World Congress*, Juillet 2005.
- [43] T. Hilaire, P. Chevrel, and Y. Trinquet. Designing low parametric sensitivity FWL realizations of LTI controllers/filters within the implicit state-space framework. In *CDC-ECC'05*, Décembre 2005.
- [40] T. Hilaire, P. Chevrel, and J.P. Clauzel. Low parametric sensitivity realization design for FWL implementation of MIMO controllers : Theory and application to the active control of vehicle longitudinal oscillations. In *CAO'O6*, Avril 2006.
- [41] T. Hilaire, P. Chevrel, and J.P. Clauzel. Pole sensitivity stability related measure of FWL realization with the implicit state-space formalism. In *Proc. ROCOND'06*, Juillet 2006.

Communications

- [38] T. Hilaire and P. Chevrel. L'optimisation convexe pour la conception et l'analyse des lois de commande - formalisation grâce aux LMI. Technical report, Août 2003.
- [37] T. Hilaire and P. Chevrel. Implémentation des lois de contrôle-commande dans les calculateurs PSA : "etat des lieux, analyse des besoins". Technical report, rapport interne PSA, Octobre 2003.
- [42] T. Hilaire, P. Chevrel, and Y. Trinquet. Implémentation des lois

de contrôle/commande dans les calculateurs PSA. JDOC : Journée des Doctorants de l'EDSTIM, 2004.

- [45] T. Hilaire, P. Chevrel, Y. Trinquet, and JP. Clauzel. Implémentation en précision finie : modélisation et recherche de réalisations optimales. GdR MACS : GT MOSAR et GT SAR, Juin 2005.
- [39] T. Hilaire, P. Chevrel, and J.P. Clauzel. Description macroscopique unifiée de l'implémentation de régulateurs et filtres linéaires. CCT PAF du CNES - Implantation des lois de commande, Novembre 2005.

Annexe

A Dérivation matricielle

Cette annexe présente les principales définitions de la dérivée matricielle, ainsi que les démonstrations de quelques propositions utiles pour le calcul de sensibilité (cf chapitre 4). On se rapportera à [96, 83] pour plus de compléments.
Dans les notations suivantes, $\mathbb{K}$ représentera $\mathbb{R}$, $\mathbb{C}$.

A.1 Divers opérateurs

Les définitions suivantes seront utiles pour l'exploitation des calculs de dérivations matricielles :

Définition A.1 (Produit de Kronecker)
Soient $A \in \mathbb{K}^{m\times n}$ et $B \in \mathbb{K}^{p\times l}$. Le produit de Kronecker de A par B, noté $A \otimes B$, est la matrice de $\mathbb{K}^{mp\times nl}$ définie par

$$A \otimes B \triangleq \begin{pmatrix} a_{11}B & \cdots & a_{1n}B \\ & & \vdots \\ & & a_{mn}B \end{pmatrix} \tag{A.1}$$

Définition A.2 (Opérateur Vec)
On note Vec *l'opérateur usuel transformant une matrice de $\mathbb{K}^{m\times n}$ en un vecteur colonne de $\mathbb{K}^{mn\times 1}$. Il vérifie*

$$\big(\mathrm{Vec}(M)\big)_{(j-1)m+i} = (M)_{i,j} \qquad \forall\ 1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n \tag{A.2}$$

Les trois propositions suivantes sur le produit de Kronecker sont des résultats classiques utilisés par la suite.

Proposition A.1
Soient les matrices A, B, C, D telles que les produits suivants existent. On a alors

$$(AC) \otimes (BD) = (A \otimes B)(C \otimes D) \tag{A.3}$$

Proposition A.2
Soient A, B, C des matrices telles que le produit ABC existe. On a

$$\mathrm{Vec}(ABC) = (C^\top \otimes A)\mathrm{Vec}(B) \tag{A.4}$$

Proposition A.3
Le transposé du produit de Kronecker de A par B est égal au produit de Kronecker des transposés de A et B

$$(A \otimes B)^\top = A^\top \otimes B^\top \tag{A.5}$$

A.2 Dérivation matricielle

A.2.1 Définitions

Définition A.3 (Dérivation matricielle dans $\mathbb{R}$ ou $\mathbb{C}$)
Soit $X \in \mathbb{R}^{m\times n}$ une matrice et $f : \mathbb{R}^{m\times n} \to \mathbb{K}$ une fonction matricielle à valeur dans $\mathbb{K}$, différentiable selon tous les éléments de X.
On définit la dérivation matricielle de f par rapport à X comme la matrice de $\mathbb{K}^{m\times n}$ dont les $(i,j)^{\text{ème}}$ éléments vérifient

$$\left(\frac{\partial f}{\partial X}\right)_{i,j} \triangleq \frac{\partial f}{\partial X_{i,j}} \tag{A.6}$$

Cette dérivation matricielle définira la sensibilité de la fonction f par rapport à X.

La définition de dérivation matricielle dans $\mathbb{R}$ ou $\mathbb{C}$ peut être étendue pour une fonction à valeur dans $\mathbb{K}^{p\times l}$:

Définition A.4 (Dérivation matricielle d'une matrice)
Soit $X \in \mathbb{R}^{m\times n}$ et $f : \mathbb{R}^{m\times n} \to \mathbb{K}^{p\times l}$. On suppose que chaque composante de f est différentiable selon tous les éléments de X.
La dérivation matricielle de f par rapport à X est une matrice de $\mathbb{K}^{mp\times nl}$ partitionnée en $m \times n$ blocs de matrices de $\mathbb{K}^{p\times l}$. Chaque $(i,j)^{\text{ème}}$ bloc est défini par

$$\frac{\partial f}{\partial X_{i,j}} \in \mathbb{K}^{p\times l} \tag{A.7}$$

On a ainsi

$$\frac{\partial f}{\partial X} \triangleq \begin{pmatrix} \frac{\partial f}{\partial X_{1,1}} & \frac{\partial f}{\partial X_{1,2}} & \cdots & \frac{\partial f}{\partial X_{1,n}} \\ \frac{\partial f}{\partial X_{2,1}} & \frac{\partial f}{\partial X_{2,2}} & \cdots & \frac{\partial f}{\partial X_{2,n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial X_{m,1}} & \frac{\partial f}{\partial X_{m,2}} & \cdots & \frac{\partial f}{\partial X_{m,n}} \end{pmatrix} \tag{A.8}$$

Ou encore

$$\left(\frac{\partial f}{\partial X}\right)_{(k-1)m+i,(l-1)n+j} = \frac{\partial f_{k,l}}{\partial X_{i,j}} \quad \begin{array}{l} 1 \leqslant k \leqslant p \\ 1 \leqslant l \leqslant q \\ 1 \leqslant i \leqslant m \\ 1 \leqslant j \leqslant n \end{array} \tag{A.9}$$

Cela peut aussi s'écrire

$$\frac{\partial f}{\partial X} = \sum_{r=1}^{m} \sum_{s=1}^{n} E_{r,s}^{m,n} \otimes \frac{\partial f}{\partial X_{r,s}} \tag{A.10}$$

avec les matrices $E_{r,s}^{n,m}$ *de* $\mathbb{R}^{n \times m}$ *définies par*

$$\left(E_{r,s}^{n,m}\right)_{i,j} \triangleq \delta_{r,i}\delta_{s,j} \tag{A.11}$$

et

$$E_{r,s}^{n,m} = \begin{pmatrix} & & \vdots & \\ & & \vdots & \\ \cdots & \cdots & 1 & \cdots \\ & & \vdots & \end{pmatrix} \begin{array}{l} \\ \\ \leftarrow r^e \text{ ligne} \\ \\ \uparrow \ s^e \text{ colonne} \end{array}$$

On remarquera que lorsque X est un vecteur ligne et $f(X)$ un vecteur colonne, $\frac{\partial f}{\partial X}$ est la matrice jacobienne de f. D'ailleurs, dans la littérature, on trouve parfois une définition différente de la dérivation matricielle par une matrice : $\frac{\partial A}{\partial X} \triangleq \frac{\partial \mathrm{Vec}(A)}{\partial \mathrm{Vec}(X)}$ ou bien $\frac{\partial A}{\partial X} \triangleq \frac{\partial \mathrm{Vec}(A)}{\partial \mathrm{Vec}(X)^\top}$, pour se ramener à une matrice jacobienne. Cela évite d'avoir à considérer des sous-blocs de matrices (ou à considérer $\frac{\partial A}{\partial X}$ comme une matrice de matrice), mais les règles de dérivations (comme les propositions A.4 et A.6) ne s'énoncent alors plus aussi facilement.

A.2.2 Règles de dérivation

Proposition A.4 (dérivation d'un produit)
Soient $X \in \mathbb{R}^{k \times l}$, $Y \in \mathbb{R}^{l \times m}$ *et* $Z \in \mathbb{R}^{p \times l}$. *La dérivée du produit* XY *par rapport à* Z *s'écrit*

$$\frac{\partial (XY)}{\partial Z} = \frac{\partial X}{\partial Z}(I_l \otimes Y) + (I_p \otimes X)\frac{\partial Y}{\partial Z} \tag{A.12}$$

avec $\frac{\partial(XY)}{\partial Z} \in \mathbb{R}^{kp \times ml}$.

Démonstration :
Les règles classiques de la dérivation d'une matrice par un scalaire donnent, pour $1 \leqslant r \leqslant p$ et $1 \leqslant s \leqslant l$

$$\frac{\partial(XY)}{\partial Z_{r,s}} = \frac{\partial X}{\partial Z_{r,s}} Y + X \frac{\partial Y}{\partial Z_{r,s}}$$

D'où

$$\begin{aligned} \frac{\partial(XY)}{\partial Z} &= \sum_{r,s} E_{r,s}^{p,l} \otimes \left(\frac{\partial X}{\partial Z_{r,s}} Y + X \frac{\partial Y}{\partial Z_{r,s}} \right) \\ &= \sum_{r,s} E_{r,s}^{p,l} \otimes \left(\frac{\partial X}{\partial Z_{r,s}} Y \right) + \sum_{r,s} E_{r,s}^{p,l} \otimes \left(X \frac{\partial Y}{\partial Z_{r,s}} \right) \end{aligned}$$

On peut noter que $\sum\limits_{r,s} E_{r,s}^{p,l} I_l = \sum\limits_{r,s} I_p E_{r,s}^{p,l} = \sum\limits_{r,s} E_{r,s}^{p,l}$.
Ainsi

$$\frac{\partial(XY)}{\partial Z} = \sum_{r,s} \left(E_{r,s}^{p,l} I_l \right) \otimes \left(\frac{\partial X}{\partial Z_{r,s}} Y \right) + \sum_{r,s} \left(I_p E_{r,s}^{p,l} \right) \otimes \left(X \frac{\partial Y}{\partial Z_{r,s}} \right)$$

En utilisant la proposition A.1, on obtient

$$\begin{aligned} \frac{\partial(XY)}{\partial Z} &= \sum_{r,s} \left(E_{r,s}^{p,l} \otimes \frac{\partial X}{\partial Z_{r,s}} \right) (I_l \otimes Y) + \sum_{r,s} (I_p \otimes X) \left(E_{r,s}^{p,l} \otimes \frac{\partial Y}{\partial Z_{r,s}} \right) \\ &= \left(\sum_{r,s} E_{r,s}^{p,l} \otimes \frac{\partial X}{\partial Z_{r,s}} \right) (I_l \otimes Y) + (I_p \otimes X) \left(\sum_{r,s} E_{r,s}^{p,l} \otimes \frac{\partial Y}{\partial Z_{r,s}} \right) \\ &= \frac{\partial X}{\partial Z_{r,s}} Y + X \frac{\partial Y}{\partial Z_{r,s}} \end{aligned}$$

■

On en déduit facilement les deux propositions suivantes :

Proposition A.5 (Dérivation de l'inverse)
Soient A une matrice inversible de $\mathbb{R}^{n \times n}$ dépendant de $X \in \mathbb{R}^{p \times l}$.
Alors

$$\frac{\partial \left(A^{-1} \right)}{\partial X} = - \left(I_p \otimes A^{-1} \right) \frac{\partial A}{\partial X} \left(I_l \otimes A^{-1} \right) \tag{A.13}$$

Démonstration :
Il suffit de remarquer que $AA^{-1} = I_n$ et d'appliquer la règle de la dérivation d'un produit, avec $\frac{\partial I_n}{\partial X} = 0_{p,l}$ ■

A.2.3 Propriétés de dérivation

Théorème A.6
Soient $X \in \mathbb{R}^{p\times l}$ et G et H deux fonctions de transfert à valeur respectivement dans $\mathbb{C}^{m\times p}$ et $\mathbb{C}^{l\times n}$. G et H sont supposées indépendantes de X. On a alors

$$\frac{\partial(GXH)}{\partial X} = (I_p \otimes G)\frac{\partial X}{\partial X}(I_l \otimes H) \quad (A.14)$$
$$= G \circledast H \quad (A.15)$$

avec l'opérateur $\circledast$ défini par

$$G \circledast H \triangleq \mathrm{Vec}(G).\left(\mathrm{Vec}(H^\top)\right)^\top \quad (A.16)$$

Démonstration :

– On peut définir $\frac{\partial X}{\partial X}$ en fonction des $E_{r,s}$

$$\frac{\partial X}{\partial X} = \sum_{r=1}^{p}\sum_{s=1}^{l}\left(E_{r,s}^{p,l} \otimes E_{r,s}^{p,l}\right)$$
$$= \sum_{r=1}^{p}\sum_{s=1}^{l} E^{p^2,l^2}_{(r-1)p+r,(s-1)l+s}$$

– On montre

$$(I_p \otimes A)(B \circledast C) = (AB) \circledast C$$
$$(B \circledast C)(I_q \otimes D) = B \circledast (CD)$$

– On montre facilement aussi que

$$\left(AE_{i,j}^{p^2,l^2}B\right) = A_{\bullet,i}B_{j,\bullet}$$

– La proposition A.4 nous donne

$$\frac{\partial(GXH)}{\partial X} = (I_p \otimes G)\frac{\partial X}{\partial X}(I_l \otimes H)$$

Puis

$$\frac{\partial(GXH)}{\partial X} = \sum_{r,s}(I_p \otimes G)E^{p^2,l^2}_{(r-1)p+r,(s-1)l+s}(I_l \otimes H)$$
$$= \sum_{r,s}(I_p \otimes G)_{\bullet,(r-1)p+r}(I_l \otimes H)_{(s-1)l+s,\bullet}$$
$$= \sum_{r,s}E_{r,s}^{p,l} \otimes (G_{\bullet,r}.H_{s,\bullet})$$

car

$$(I_p \otimes G)^{\top}_{\bullet,(r-1)p+r} = \Big(\boxed{0} \quad \dots \quad \underset{\uparrow\, r^{\text{ième}} \text{ bloc}}{\boxed{G_{\bullet,r}}} \quad \boxed{0} \quad \dots \Big)$$

Et

$$E^{p,l}_{r,s} \otimes (G_{\bullet,r}.H_{s,\bullet}) = \underset{(r,s)^{\text{ème}} \text{ bloc de taille } (m \times n)}{\begin{pmatrix} \boxed{0} & & \\ & \boxed{G_{*,r}.H_{s,*}} & \\ & & \boxed{0} \end{pmatrix}}$$

– Enfin, on a

$$(G_{\bullet,r}.H_{s,\bullet})_{i,j} = G_{i,r}.H_{s,j}$$

et donc

$$\left(\frac{\partial(GXH)}{\partial X}\right)_{(r-1)m+i,(s-1)n+j} = G_{i,r}.H_{s,j}$$

On retrouve ainsi (avec l'équation (A.2))

$$\frac{\partial(GXH)}{\partial X} = \text{Vec}(G).\left(\text{Vec}(H^{\top})\right)^{\top}$$

■

Remarque : la définition de l'opérateur *trace* est

$$tr(AB) = \left(\text{Vec}(A^{\top})\right)^{\top} \text{Vec}(B)$$

alors que celle de l'opérateur $\circledast$ est

$$A \circledast B = \text{Vec}(A).\left(\text{Vec}(B^{\top})\right)^{\top}$$

Proposition A.7
Si l'on reprend les conditions de la proposition précédente, avec $p = l$ et X inversible, on a alors

$$\frac{\partial\left(GX^{-1}H\right)}{\partial X} = -\left(GX^{-1}\right) \circledast \left(X^{-1}H\right) \tag{A.17}$$

Démonstration :
G et H sont indépendantes de X, donc d'après la proposition A.4

$$\frac{\partial \left(GX^{-1}H\right)}{\partial X} = \left(I_p \otimes G\right) \frac{\partial X^{-1}}{\partial X} \left(I_p \otimes H\right)$$

d'où (avec la proposition A.5)

$$\frac{\partial \left(GX^{-1}H\right)}{\partial X} = -\left(I_p \otimes G\right) \left(I_p \otimes X^{-1}\right) \frac{\partial X}{\partial X} \left(I_p \otimes X^{-1}\right) \left(I_p \otimes H\right)$$

et

$$\frac{\partial \left(GX^{-1}H\right)}{\partial X} = -\left(GX^{-1}\right) \circledast \left(X^{-1}H\right) \tag{A.18}$$

■

Les quelques propositions suivantes portent sur l'opérateur binaire $\circledast$:

Proposition A.8
Soient $X \in \mathbb{R}^{p\times l}$, G et H deux fonctions de transfert à valeur respectivement dans $\mathbb{C}^{m\times p}$ et $\mathbb{C}^{l\times n}$, et M une matrice constante.
On obtient les résultats suivants :

$$\|G \circledast H\|_2^2 = \sum_{i,j} \sum_{k,l} \|G_{i,j} H_{k,l}\|_2^2 \tag{A.19}$$

$$\|M \circledast G\|_2^2 = \|M\|_F^2 \, \|G\|_2^2 \tag{A.20}$$

$$\|I_p \circledast G\|_2^2 = l \, \|G\|_2^2 \tag{A.21}$$

$$\|H \circledast I_l\|_2^2 = p \, \|H\|_2^2 \tag{A.22}$$

Démonstration :
La première équation se vérifie par la propriété de la norme 2 :

$$\|G \circledast H\|_2^2 = \sum_{i,j} \|(G \circledast H)_{i,j}\|_2^2$$

La deuxième équation se vérifie par

$$\begin{aligned}
\|M \circledast G\|_2^2 &= \sum_{i,j} \sum_{k,l} \|M_{i,j} G_{k,l}\|_2^2 \\
&= \sum_{i,j} \sum_{k,l} M_{i,j}^2 \, \|G_{k,l}\|_2^2 \\
&= \left(\sum_{i,j} M_{i,j}^2\right) \left(\sum_{k,l} \|G_{k,l}\|_2^2\right) \\
&= \|M\|_F^2 \, \|G\|_2^2
\end{aligned}$$

■

Proposition A.9
Soient A, B et C des matrices de dimensions appropriées.
On a alors

$$\begin{pmatrix} A & B \end{pmatrix} \circledast C = \begin{pmatrix} A \circledast C \\ B \circledast C \end{pmatrix} \tag{A.23}$$

et

$$A \circledast \begin{pmatrix} B \\ C \end{pmatrix} = \begin{pmatrix} A \circledast B & A \circledast C \end{pmatrix} \tag{A.24}$$

Démonstration :

$$\begin{aligned}
\begin{pmatrix} A & B \end{pmatrix} \circledast C &= \mathrm{Vec}\left(\begin{pmatrix} A & B \end{pmatrix}\right) . \left(\mathrm{Vec}(C^\top)\right)^\top \\
&= \begin{pmatrix} \mathrm{Vec}(A) \\ \mathrm{Vec}(B) \end{pmatrix} . \left(\mathrm{Vec}(C^\top)\right)^\top \\
&= \begin{pmatrix} \mathrm{Vec}(A) . \left(\mathrm{Vec}(C^\top)\right)^\top \\ \mathrm{Vec}(B) \left(\mathrm{Vec}(C^\top)\right)^\top \end{pmatrix} \\
&= \begin{pmatrix} A \circledast C \\ B \circledast C \end{pmatrix}
\end{aligned}$$

et

$$\begin{aligned}
A \circledast \begin{pmatrix} B \\ C \end{pmatrix} &= \mathrm{Vec}(A) . \left(\mathrm{Vec}\left(\begin{pmatrix} B^\top & C^\top \end{pmatrix}\right)\right)^\top \\
&= \mathrm{Vec}(A) . \left(\begin{pmatrix} \mathrm{Vec}(B^\top) \\ \mathrm{Vec}(C^\top) \end{pmatrix}\right)^\top \\
&= \mathrm{Vec}(A) . \left(\left(\mathrm{Vec}(B^\top)\right)^\top \quad \left(\mathrm{Vec}(C^\top)\right)^\top\right) \\
&= \left(\mathrm{Vec}(A) . \left(\mathrm{Vec}(B^\top)\right)^\top \quad \mathrm{Vec}(A) . \left(\mathrm{Vec}(C^\top)\right)^\top\right) \\
&= \begin{pmatrix} A \circledast B & A \circledast C \end{pmatrix}
\end{aligned}$$

■

Proposition A.10
Soient G, H, X et Y quatre fonctions de transfert à valeur dans $\mathbb{C}^{m\times p}$, $\mathbb{C}^{l\times n}$ $\mathbb{C}^{p\times p}$ et $\mathbb{C}^{l\times l}$. On a

$$(GX) \circledast (YH) = (I_p \otimes G)\,(X \circledast Y)\,(I_l \otimes H) \tag{A.25}$$

et

$$(GX) \circledast (YH) = \left(X^\top \otimes I_m\right)(G \circledast H)\left(Y^\top \otimes I_n\right) \tag{A.26}$$

Démonstration :

$$\begin{aligned}
(GX) \circledast (YH) &= \mathrm{Vec}\,(GX)\left(\mathrm{Vec}\,(YH)^\top\right)^\top \\
&= \left(X^\top \otimes I_m\right)\mathrm{Vec}(G)\left((Y \otimes I_n)\,\mathrm{Vec}(H^\top)\right)^\top \\
&= \left(X^\top \otimes I_m\right)\mathrm{Vec}(G)\left(\mathrm{Vec}(H^\top)\right)^\top (Y \otimes I_n)^\top \\
&= \left(X^\top \otimes I_m\right)(G \circledast H)\left(Y^\top \otimes I_n\right)
\end{aligned}$$

■

Annexe

B Démonstrations diverses

Proposition B.1 (Fonction de transfert en δ)
On considère une fonction de transfert H telle que

$$H(z) = \frac{\sum_{i=0}^{n} b_i z^{-i}}{1 + \sum_{i=0}^{n} a_i z^{-i}} \qquad \forall z \in \mathbb{C} \tag{B.1}$$

Alors, on peut exprimer cette fonction de transfert avec la variable ρ liée à l'opérateur δ ($\rho = \frac{z-1}{\Delta}$)

$$H(\rho) = \frac{\sum_{i=0}^{n} d_i \rho^{-i}}{1 + \sum_{i=1}^{n} c_i \rho^{-i}} \qquad \forall \rho \in \mathbb{C} \tag{B.2}$$

avec les coefficients $(c_k)_{1 \leqslant k \leqslant n}$ et $(d_k)_{0 \leqslant k \leqslant n}$ tels que

$$c_k = \Delta^{-k} \sum_{l=0}^{k} a_l C_{k-l}^{n-l} \qquad \forall 1 \leqslant k \leqslant n \tag{B.3}$$

$$d_k = \Delta^{-k} \sum_{l=0}^{k} b_l C_{k-l}^{n-l} \qquad \forall 0 \leqslant k \leqslant n \tag{B.4}$$

Démonstration :
Avec $q^{-1} = \frac{\delta^{-1}}{\Delta+\delta^{-1}}$ et $(a_0 = 1)$, on a

$$\begin{aligned} H(z) &= \frac{\sum_{i=0}^{n} b_i z^i}{\sum_{i=0}^{n} a_i z^{-i}} \\ &= \frac{\sum_{i=0}^{n} b_i \rho^{-i} \left(\Delta + \rho^{-1}\right)^{n-i}}{\sum_{i=0}^{n} a_i \rho^{-i} \left(\Delta + \rho^{-1}\right)^{n-i}} \end{aligned}$$

Or

$$\left(\Delta + \rho^{-1}\right)^{n-i} = \sum_{j=0}^{n-i} \rho^{-j} \Delta^{n-i-j} C_j^{n-i}$$

d'où

$$\begin{aligned} \sum_{i=0}^{n} b_i \rho^{-i} \left(\Delta + \rho^{-1}\right)^{n-i} &= \sum_{i=0}^{n} \sum_{j=0}^{n-i} b_i \Delta^{n-(i+j)} C_j^{n-i} \rho^{-(i+j)} \\ &= \sum_{k=0}^{n} \rho^{-k} \Delta^{n-k} \left(\sum_{l=0}^{k} b_l C_{k-l}^{n-l} \right) \end{aligned}$$

Et

$$\sum_{i=0}^{n} a_i \rho^{-i} \left(\Delta + \rho^{-1}\right)^{n-i} = \sum_{k=0}^{n} \rho^{-k} \Delta^{n-k} \left(\sum_{l=0}^{k} a_l C_{k-l}^{n-l} \right)$$

Ainsi

$$\begin{aligned} H(\rho) &= \frac{\sum_{k=0}^{n} \rho^{-k} \Delta^{n-k} \left(\sum_{l=0}^{k} b_l C_{k-l}^{n-l} \right)}{\sum_{k=0}^{n} \rho^{-k} \Delta^{n-k} \left(\sum_{l=0}^{k} a_l C_{k-l}^{n-l} \right)} \\ &= \frac{\sum_{i=0}^{n} d_i \rho^{-i}}{1 + \sum_{i=1}^{n} c_i \rho^{-i}} \end{aligned}$$

avec

$$c_k = \Delta^{-k} \sum_{l=0}^{k} a_l C_{k-l}^{n-l} \qquad 1 \leqslant k \leqslant n \tag{B.5}$$

$$d_k = \Delta^{-k} \sum_{l=0}^{k} b_l C_{k-l}^{n-l} \qquad 0 \leqslant k \leqslant n \tag{B.6}$$

■

Théorème B.2 (Calcul des sensibilités)
La sensibilité de la fonction de transfert H, mise en œuvre par la réalisation $\mathcal{R} := (J, K, L, M, N, P, Q, R, S)$, a pour expression

$$\frac{\partial H}{\partial Z} = \begin{pmatrix} H_3 & H_1 & I_p \end{pmatrix} \circledast \begin{pmatrix} H_4 \\ H_2 \\ I_n \end{pmatrix} \tag{B.7}$$

$$\frac{\partial D}{\partial Z} = \begin{pmatrix} LJ^{-1} & 0 & I_p \end{pmatrix} \circledast \begin{pmatrix} J^{-1}N \\ 0 \\ I_n \end{pmatrix} \tag{B.8}$$

avec

$$H_1 : z \mapsto C(zI_n - A)^{-1} \tag{B.9}$$
$$H_2 : z \mapsto (zI_n - A)^{-1}B \tag{B.10}$$
$$H_3 : z \mapsto C(zI_n - A)^{-1}KJ^{-1} + LJ^{-1} \tag{B.11}$$
$$H_4 : z \mapsto J^{-1}M(zI_n - A)^{-1}B + J^{-1}N \tag{B.12}$$

On a bien entendu

$$\frac{\partial \tilde{H}}{\partial Z} = \frac{\partial H}{\partial Z} - \frac{\partial D}{\partial Z} \tag{B.13}$$

et l'on pourra noter que $\frac{\partial \tilde{H}}{\partial Z}$ peut être calculé à partir de $\frac{\partial H}{\partial Z}$ en rendant cette fonction de transfert strictement propre (en supprimant les termes constants, qui valent $\frac{\partial D}{\partial Z}$).

Démonstration :

- $\dfrac{\partial H(z)}{\partial S} = \dfrac{\partial S}{\partial S} = I_p \circledast I_m$
- $\dfrac{\partial H(z)}{\partial R} = \dfrac{\partial \left(R(zI_n - A)^{-1}B\right)}{\partial R} = I_p \circledast \left((zI_n - A)^{-1}B\right)$
- $\dfrac{\partial H(z)}{\partial Q} = \dfrac{\partial \left(C(zI_n - A)^{-1}Q\right)}{\partial Q} = \left(C(zI_n - A)^{-1}\right) \circledast I_m$
- $\dfrac{\partial H(z)}{\partial P} = -\left(I_n \otimes C(zI_n - A)^{-1}\right) \dfrac{\partial A}{\partial P} \left(I_n \otimes (zI_n - A)^{-1}B\right)$
  $= \left(C(zI_n - A)^{-1}\right) \circledast \left((zI_n - A)^{-1}B\right)$
- $\dfrac{\partial H(z)}{\partial N} = \dfrac{\partial \left(C(zI_n - A)^{-1}KJ^{-1}N\right)}{\partial N} + \dfrac{\partial \left(LJ^{-1}N\right)}{\partial N}$
  $= \left(C(zI_n - A)^{-1}KJ^{-1} + LJ^{-1}\right) \circledast I_m$

$$
\begin{aligned}
\bullet\ \frac{\partial H(z)}{\partial M} &= \left(I_l \otimes LJ^{-1}\right) \frac{\partial M}{\partial M} \left(I_n \otimes (zI_n - A)^{-1}B\right) \\
&\quad + (I_l \otimes C) \frac{\partial \left((zI_n - A)^{-1}\right)}{\partial M} (I_n \otimes B) \\
&= \left(LJ^{-1}\right) \circledast \left((zI_n - A)^{-1}B\right) \\
&\quad + \left(I_l \otimes C(zI_n - A)^{-1}\right) \frac{\partial KJ^{-1}M}{\partial M} \left(I_n \otimes (zI_n - A)^{-1}B\right) \\
&= \left(C(zI_n - A)^{-1}KJ^{-1} + LJ^{-1}\right) \circledast \left((zI_n - A)^{-1}B\right) \\
\bullet\ \frac{\partial H(z)}{\partial L} &= \frac{\partial \left(LJ^{-1}M(zI_n - A)^{-1}B\right)}{\partial L} + \frac{\partial LJ^{-1}N}{\partial L} \\
&= I_p \circledast \left(J^{-1}M(zI_n - A)^{-1}B + LJ^{-1}\right) \\
\bullet\ \frac{\partial H(z)}{\partial K} &= (I_n \otimes C) \frac{\partial \left((zI_n - A)^{-1}\right)}{\partial K} (I_l \otimes B) \\
&\quad + \left(I_n \otimes C(zI_n - A)^{-1}\right) \frac{\partial K}{\partial K} \left(I_l \otimes J^{-1}N\right) \\
&= \left(I_n \otimes C(zI_n - A)^{-1}\right) \frac{\partial \left(KJ^{-1}N\right)}{\partial K} \left(I_l \otimes (zI_n - A)^{-1}B\right) \\
&\quad + \left(C(zI_n - A)^{-1}\right) \circledast \left(J^{-1}N\right) \\
&= \left(C(zI_n - A)^{-1}\right) \circledast \left(J^{-1}M(zI_n - A)^{-1}B + J^{-1}N\right) \\
\bullet\ \frac{\partial H(z)}{\partial J} &= \frac{\partial \left(LJ^{-1}M\right)}{\partial J} \left(I_l \otimes (zI - A)^{-1}B\right) + \frac{\partial \left(LJ^{-1}N\right)}{\partial J} \\
&\quad + \left(I_l \otimes C(zI_n - A)^{-1}\right) \frac{\partial \left(KJ^{-1}N\right)}{\partial J} \\
&\quad + \left(I_l \otimes C(zI_n - A)^{-1}\right) \frac{\partial \left(KJ^{-1}M\right)}{\partial J} \left(I_l \otimes (zI_n - A)^{-1}B\right) \\
&= -\left(LJ^{-1}\right) \circledast \left(J^{-1}M(zI_n - A)^{-1}B\right) - \left(LJ^{-1}\right) \circledast \left(J^{-1}N\right) \\
&\quad - \left(C(zI_n - A)^{-1}KJ^{-1}\right) \circledast \left(J^{-1}N\right) \\
&\quad - \left(C(zI_n - A)^{-1}KJ^{-1}\right) \circledast \left(J^{-1}M(zI_n - A)^{-1}B\right) \\
&= -\left(C(zI_n - A)^{-1}KJ^{-1} + LJ^{-1}\right) \circledast \left(J^{-1}M(zI_n - A)^{-1}B + LJ^{-1}\right)
\end{aligned}
$$

■

Proposition B.3

On considère le système bouclé défini au chapitre 4.3.1. Les sensibilités $\frac{\partial \bar{A}}{\partial Z}$ *et* $\frac{\partial \bar{B}}{\partial Z}$ *sont données par*

$$
\frac{\partial \bar{A}}{\partial Z} = \begin{pmatrix} B_p LJ^{-1} & 0 & B_p \\ KJ^{-1} & I_n & 0 \end{pmatrix} \circledast \begin{pmatrix} J^{-1}N\tilde{C}_p & J^{-1}M \\ 0 & I_n \\ \tilde{C}_p & 0 \end{pmatrix} \tag{B.14}
$$

$$
\frac{\partial \bar{B}}{\partial Z} = \begin{pmatrix} B_p LJ^{-1} & 0 & B_p \\ KJ^{-1} & I_n & 0 \end{pmatrix} \circledast \begin{pmatrix} J^{-1}N\tilde{I} \\ 0 \\ \tilde{I} \end{pmatrix} \tag{B.15}
$$

Démonstration :
Pour évaluer $\frac{\partial \bar{A}}{\partial Z}$ et $\frac{\partial \bar{B}}{\partial Z}$, on exprime tout d'abord $\bar{A}$ et $\bar{B}$ sous la forme $M_1 X M_2 + M_3$, pour $X \in \{J^{-1}, K, L, M, N, P, Q, R, S\}$

$$\bullet\ \bar{A} = \begin{pmatrix} A_p + B_p L J^{-1} N \tilde{C}_p & B_p C \\ B\tilde{C}_p & A \end{pmatrix} + \begin{pmatrix} B_p \\ 0 \end{pmatrix} S \begin{pmatrix} \tilde{C}_p & 0 \end{pmatrix} \quad \text{(B.16)}$$

$$\bullet\ \bar{A} = \begin{pmatrix} A_p + B_p D \tilde{C}_p & B_p L J^{-1} M \\ B\tilde{C}_p & A \end{pmatrix} + \begin{pmatrix} B_p \\ 0 \end{pmatrix} R \begin{pmatrix} 0 & I_n \end{pmatrix} \quad \text{(B.17)}$$

$$\bullet\ \bar{A} = \begin{pmatrix} A_p + B_p D \tilde{C}_p & B_p C \\ B\tilde{C}_p & K J^{-1} M \end{pmatrix} + \begin{pmatrix} 0 \\ I_n \end{pmatrix} P \begin{pmatrix} 0 & I_n \end{pmatrix} \quad \text{(B.18)}$$

$$\bullet\ \bar{A} = \begin{pmatrix} A_p + B_p D \tilde{C}_p & B_p C \\ K J^{-1} N \tilde{C}_p & A \end{pmatrix} + \begin{pmatrix} 0 \\ I_n \end{pmatrix} Q \begin{pmatrix} \tilde{C}_p & 0 \end{pmatrix} \quad \text{(B.19)}$$

$$\bullet\ \bar{A} = \begin{pmatrix} A_p + B_p S \tilde{C}_p & B_p C \\ Q\tilde{C}_p & A \end{pmatrix} + \begin{pmatrix} B_p L J^{-1} \\ K J^{-1} \end{pmatrix} N \begin{pmatrix} \tilde{C}_p & 0 \end{pmatrix} \quad \text{(B.20)}$$

$$\bullet\ \bar{A} = \begin{pmatrix} A_p + B_p D \tilde{C}_p & B_p R \\ B\tilde{C}_p & P \end{pmatrix} + \begin{pmatrix} B_p L J^{-1} \\ K J^{-1} \end{pmatrix} M \begin{pmatrix} 0 & I_n \end{pmatrix} \quad \text{(B.21)}$$

$$\bullet\ \bar{A} = \begin{pmatrix} A_p + B_p D \tilde{C}_p & B_p C \\ Q\tilde{C}_p & P \end{pmatrix} + \begin{pmatrix} 0 \\ I_n \end{pmatrix} K \begin{pmatrix} J^{-1} N \tilde{C}_p & J^{-1} M \end{pmatrix} \quad \text{(B.22)}$$

$$\bullet\ \bar{A} = \begin{pmatrix} A_p + B_p S \tilde{C}_p & B_p R \\ B\tilde{C}_p & A \end{pmatrix} + \begin{pmatrix} B_p \\ 0 \end{pmatrix} L \begin{pmatrix} J^{-1} N \tilde{C}_p & J^{-1} M \end{pmatrix} \quad \text{(B.23)}$$

$$\bullet\ \bar{A} = \begin{pmatrix} A_p + B_p S \tilde{C}_p & B_p R \\ Q\tilde{C}_p & P \end{pmatrix} + \begin{pmatrix} B_p L \\ K \end{pmatrix} J^{-1} \begin{pmatrix} N \tilde{C}_p & M \end{pmatrix} \quad \text{(B.24)}$$

$$\bullet\ \bar{B} = \begin{pmatrix} B_p L J^{-1} N \\ B \end{pmatrix} \tilde{I} + \begin{pmatrix} B_p \\ 0 \end{pmatrix} S\tilde{I} \quad \text{(B.25)}$$

$$\bullet\ \bar{B} = \begin{pmatrix} B_p D \\ K J^{-1} N \end{pmatrix} \tilde{I} + \begin{pmatrix} 0 \\ I \end{pmatrix} Q\tilde{I} \quad \text{(B.26)}$$

$$\bullet\ \bar{B} = \begin{pmatrix} B_p S \\ Q \end{pmatrix} \tilde{I} + \begin{pmatrix} B_p L J^{-1} \\ K J^{-1} \end{pmatrix} N\tilde{I} \quad \text{(B.27)}$$

$$\bullet\ \bar{B} = \begin{pmatrix} B_p D \\ Q \end{pmatrix} \tilde{I} + \begin{pmatrix} 0 \\ I \end{pmatrix} K(J^{-1} N\tilde{I}) \quad \text{(B.28)}$$

$$\bullet\ \bar{B} = \begin{pmatrix} B_p S \\ D \end{pmatrix} \tilde{I} + \begin{pmatrix} B_p \\ 0 \end{pmatrix} L(J^{-1} N\tilde{I}) \quad \text{(B.29)}$$

$$\bullet\ \bar{B} = \begin{pmatrix} B_p S \\ Q \end{pmatrix} \tilde{I} + \begin{pmatrix} B_p L \\ K \end{pmatrix} J^{-1}(N\tilde{I}) \quad \text{(B.30)}$$

On utilise alors la proposition 4.5, ce qui donne

$$\frac{\partial \bar{A}}{\partial S} = \begin{pmatrix} B_p \\ 0 \end{pmatrix} \circledast \begin{pmatrix} \tilde{C}_p & 0 \end{pmatrix} \tag{B.31}$$

$$\frac{\partial \bar{A}}{\partial R} = \begin{pmatrix} B_p \\ 0 \end{pmatrix} \circledast \begin{pmatrix} 0 & I_n \end{pmatrix} \tag{B.32}$$

$$\frac{\partial \bar{A}}{\partial P} = \begin{pmatrix} 0 \\ I_n \end{pmatrix} \circledast \begin{pmatrix} 0 & I_n \end{pmatrix} \tag{B.33}$$

$$\frac{\partial \bar{A}}{\partial Q} = \begin{pmatrix} 0 \\ I_n \end{pmatrix} \circledast \begin{pmatrix} \tilde{C}_p & 0 \end{pmatrix} \tag{B.34}$$

$$\frac{\partial \bar{A}}{\partial N} = \begin{pmatrix} B_p L J^{-1} \\ K J^{-1} \end{pmatrix} \circledast \begin{pmatrix} \tilde{C}_p & 0 \end{pmatrix} \tag{B.35}$$

$$\frac{\partial \bar{A}}{\partial M} = \begin{pmatrix} B_p L J^{-1} \\ K J^{-1} \end{pmatrix} \circledast \begin{pmatrix} 0 & I_n \end{pmatrix} \tag{B.36}$$

$$\frac{\partial \bar{A}}{\partial K} = \begin{pmatrix} 0 \\ I_n \end{pmatrix} \circledast \begin{pmatrix} J^{-1} N \tilde{C}_p & J^{-1} M \end{pmatrix} \tag{B.37}$$

$$\frac{\partial \bar{A}}{\partial L} = \begin{pmatrix} B_p \\ 0 \end{pmatrix} \circledast \begin{pmatrix} J^{-1} N \tilde{C}_p & J^{-1} M \end{pmatrix} \tag{B.38}$$

$$\frac{\partial \bar{A}}{\partial J} = -\begin{pmatrix} B_p L J^{-1} \\ K J^{-1} \end{pmatrix} \circledast \begin{pmatrix} J^{-1} N \tilde{C}_p & J^{-1} M \end{pmatrix} \tag{B.39}$$

$$\frac{\partial \bar{B}}{\partial S} = \begin{pmatrix} B_p \\ 0 \end{pmatrix} \circledast \tilde{I} \tag{B.40}$$

$$\frac{\partial \bar{B}}{\partial Q} = \begin{pmatrix} 0 \\ I \end{pmatrix} \circledast \tilde{I} \tag{B.41}$$

$$\frac{\partial \bar{B}}{\partial N} = \begin{pmatrix} B_p L J^{-1} \\ K J^{-1} \end{pmatrix} \circledast \tilde{I} \tag{B.42}$$

$$\frac{\partial \bar{B}}{\partial K} = \begin{pmatrix} 0 \\ I \end{pmatrix} \circledast (J^{-1} N \tilde{I}) \tag{B.43}$$

$$\frac{\partial \bar{B}}{\partial L} = \begin{pmatrix} B_p \\ 0 \end{pmatrix} \circledast (J^{-1} N \tilde{I}) \tag{B.44}$$

$$\frac{\partial \bar{B}}{\partial J} = -\begin{pmatrix} B_p L J^{-1} \\ K J^{-1} \end{pmatrix} \circledast (J^{-1} N \tilde{I}) \tag{B.45}$$

$$\frac{\partial \bar{B}}{\partial R} = 0 \tag{B.46}$$

$$\frac{\partial \bar{B}}{\partial P} = 0 \tag{B.47}$$

$$\frac{\partial \bar{B}}{\partial M} = 0 \tag{B.48}$$

En regroupant ces équations (grâce à la définition de Z, équation (3.16)), on obtient les équations (B.14) et (B.15). ■

Bibliographie

[1] *Fixed-Point Blockset User's Guide (for use with Simulink)*, 2004. v 1.1 for Matlab R14. 68, 74

[2] Sciences & Vie Hors-Série Automobile, 2004. 26

[3] B. Abou, Y. Chamaillard, G. Corde, G. Gissinger, P. Higelin, N. Le Fort-Piat, and J. Malville. *Contrôle-commande de la voiture*. Traité IC2. G. Gissinger and N. Le Fort-Piat, 2002. 27

[4] D. Alazard, C. Curres, P. Apkarian, M. Gauvrit, and G. Ferreses. *Robustesse et Commande Optimale*. Cepadues Edition, 1999. 41, 42, 160

[5] G. Antoniou. Generalized one-multiplier lattice discrete 2-d filters : Minimal circuit and state-space realization. *IEEE Trans. on Circuits and Systems*, 48(2) :215–219, Februray 2001. 47

[6] J. Aplevich. *Implicit Linear Systems*. Springer-Verlag, 1991. 83

[7] A.D. Back, A.C. Tsoi, B.G. Horne, and C.L. Giles. Alternative discrete time operators and their application to nonlinear models. *IEEE Transactions on Signal Processing, special issue on Applications of Neural Networks to Signal Processing*, 1996. 51

[8] L. Bakule, J. Rodellar, and J. Rossell. Structure of expansion-contraction matrices in the inclusion principle for dynamic systems. *SIAM Matrix Anal. Appl.*, 21(4) :1136–1155, 2000. 97

[9] L. Bakule, J. Rodellar, and J. Rossell. Controllability-observability of expanded composite systems. *ELSEVIER - Linear Algrebra and its Application*, 332-334, 2001. 97

[10] A. Barraud, editor. *Outils d'analyse numérique pour l'Automatique*. Traité IC2. Hermès Science, 2002. 74, 204, 207

[11] M. Bellanger. *Traitement Numérique du Signal - Théorie et Pratique*. Masson, 1994. 48

[12] JL. Beuchat. *Etude et conception d'opérateurs arithmétiques optimisés pour circuits programmables*. PhD thesis, Ecole Polythechnique Fédérale de Lausanne, 2001. 59, 62

[13] A. Booth. A signed binary multiplication technique. *Quarterly Journal of Mechanics and Applied Mathematics*, 4(2) :236–240, June 1951. 68

[14] R. Brent. Algorithms for matrix multiplication. Technical report, Computer Science Departement - Stanford University, 1970. 77

[15] G. Cantor. *Sur les fondements de la théorie des ensembles transfinis*. Mémoires de la Société des Sciences physiques et naturelles de Bordeaux. Edition Jacques Gabay, 1899. 57

[16] GJ Chaitin. On the length of programs for computing finite binary sequences. *Journal of the Association of Computing Machinery*, 13 :547–569, 1966. 112

[17] S. Chen, R.H. Istepanian, J. Wu, and J. Chu. Comparative study on optimizing closed-loop stability bounds of finite-precision controller structures with shift and delta operators. *Systems & Control Letters*, 40(3) :153–163, 2000. 148

[18] S. Chen and B.L. Luk. Adaptive simulated annealing for optimization in signal processing applications. *Signal Processing*, 79 :117–128, 1999. 167

[19] S. Chen, J. Wu, R.H. Istepanian, J. Chu, and J.F. Whidborne. Optimizing stability bounds of finite-precision controller structures for sampled-data systems in the delta operator domain. In *IEE Proc. Control Theory and Applications*, volume 146, 1999. 148, 167

[20] JM. Chesneaux. *Outils d'analyse numérique pour l'Automatique*, chapter Estimation statistique des erreurs d'arrondi, pages 53–82. In Barraud [10], 2002. 70

[21] P. Chevrel. Discrétisation et codage des régulateurs en vue de leur maintenance et de leur codage. In *GDR Auto, Journée thématique "Lisibilité et retouche des correcteurs"*, 2002. 181

[22] R. Cmar, L. Rijnders, P.Schaumont, S.Vernalde, and I. Bolsens. A methodology and design environment for DSP ASIC fixed point refinement. *Design, Automation and Test in Europe (DATE'99)*, page 271, March 1999. 75

[23] G. Constantinides, P. Cheung, and W. Luk. The multiple wordlength paradigm. In *Proc. IEEE Symposium on Field-Programmable Custom Computing Machines*, 2001. 74, 75

[24] L. Dai. *Singular Control Systems*. Springer-verlag, lecture note in control and information sciences edition, 1989. 83

[25] T.O. de Silva, P.G. de Oliveira, J.C. Principe, and B. de Vries. Generalized feedforward filters with complex poles. In *Proceedings of the 1992 IEEE Workshop on Neural Networks for Signal Processing II*, pages 503–510, 1992. 52

[26] D. Defour. *Fonctions élémentaires : algorithmes et implémentations efficaces pour l'arrondi correct en double précision*. PhD thesis, Ecole Normale Supérieure de Lyon, 2003. 73, 74

[27] JJ di Stefano, A. Stubberud, and I. Williams. *Systèmes asservis - Cours et problèmes*. Series Schaum. Mc Graw Hill, 1994. 39

[28] P. Diniz and A. Antoniou. More economical state-space digital-filter structures which are free of constant-input limit cycles. *Acoustics, Speech and Signal Processing*, 34(4) :807–815, August 1986. 38

[29] JP. Elloy. *le Temps réel : rapport établi par le Groupe de Réflexion Temps Réel du département "Sciences Physiques pour l'Ingénieur" du CNRS*, volume 1. CNRS, 1998. 32

[30] F. Fang, T. Chen, and R. Rutenbar. Floating-point bit-width optimization for low-power signal processing applications. *ICASSP*, May 2002. 75

[31] M. Gevers and G. Li. *Parametrizations in Control, Estimation and Filtering Probems*. Springer-Verlag, 1993. 33, 40, 48, 49, 51, 85, 123, 124, 125, 126, 133, 135, 147, 148, 158, 180

[32] D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1) :5–48, 1991. 73, 74

[33] T. Granlund. *The GNU Multiple Precision Arithmetic Library*, 2004. 58

[34] A. Garcia Guerra. Analyse des différentes solutions d'implémentation des filtres dans un calculateur. Technical report, PSA Peugeot Citroën, 2002. 43

[35] A. Garcia Guerra. Implémentation des filtres numériques dans un calculateur. Technical report, PSA Peugeot Citroën, 2002. 43

[36] H. Hanselmann. Implementation of digital controllers - a survey. *Automatica*, 23(1) :7–32, January 1987. 53

[37] T. Hilaire and P. Chevrel. Implémentation des lois de contrôle/commande dans les calculateurs PSA : "etat des lieux, analyse des besoins". Technical report, rapport interne PSA, october 2003. 30, 180, 183

[38] T. Hilaire and P. Chevrel. L'optimisation convexe pour la conception et l'analyse des lois de commande - formalisation grâce aux LMI. Technical report, august 2003. 183

[39] T. Hilaire, P. Chevrel, and J-P. Clauzel. Description macroscopique unifiée de l'implémentation de régulateurs et filtres linéaires. CCT PAF du CNES - Implantation des lois de commande, November 2005. 184

[40] T. Hilaire, P. Chevrel, and J-P. Clauzel. Low parametric sensitivity realization design for fwl implementation of mimo controllers : Theory and application to the active control of vehicle longitudinal oscillations. In *Proc. of Control Applications of Optimisation CAO'O6*, April 2006. 23, 130, 183

[41] T. Hilaire, P. Chevrel, and J-P. Clauzel. Pole sensitivity stability related measure of fwl realization with the implicit state-space formalism. In *to appear in : Proc. ROCOND'06*, July 2006. 23, 144, 183

[42] T. Hilaire, P. Chevrel, and Y. Trinquet. Implémentation des lois de contrôle/commande dans les calculateurs PSA. JDOC : Journée des Doctorants de l'EDSTIM, 2004. 183

[43] T. Hilaire, P. Chevrel, and Y. Trinquet. Designing low parametric sensitivity FWL realizations of LTI controllers/filters within the implicit state-space framework. In *CDC-ECC'05*, December 2005. 23, 85, 127, 183

[44] T. Hilaire, P. Chevrel, and Y. Trinquet. Implicit state-space representation : a unifying framework for FWL implementation of LTI systems. In *IFAC05 World Congress*, July 2005. 23, 81, 183

[45] T. Hilaire, P. Chevrel, Y. Trinquet, and J-P. Clauzel. Implémentation en précision finie : modélisation et recherche de réalisations optimales. GdR MACS : GT MOSAR et GT SAR, June 2005. 184

[46] S.Y. Hwang. Dynamic range constraint in state-space digital filtering. In *IEEE Trans. on Acoustics, Speech and Signal Processing*, volume 23, pages 591–593, 1975. 133

[47] S.Y. Hwang. Minimum uncorrelated unit noise in state-space digital filtering. *IEEE Trans. on Acoust., Speech, and Signal Processing*, 25(4) :273–281, August 1977. 133

[48] G. Ifrah. *Histoire Universelle des Chiffres*, volume 2. 1994. 59

[49] M. Ikeda and D. Šiljak. Overlapping decompositions, expansions, and contractions of dynamic systems. *Large Scale Syst.*, 1 :29–38, 1980. 97, 98

[50] M. Ikeda, D. Šiljak, and D. White. An inclusion principle for dynamic systems. *IEEE Trans. Automatic Control*, 29(3) :244–249, March 1984. 97, 98, 100

[51] L. Ingber. Adaptive simulated annealing (ASA) : Lessons learned. *Control and Cybernetics*, 25(1) :33–54, 1996. 167

[52] R. Istepanian and J.F. Whidborne, editors. *Digital Controller implementation and fragility*. Springer, 2001. 14, 139, 145, 180, 210, 211

[53] R. Istepanian, J.F. Whidborne, and P. Bauer. Stability analysis of block floating point digital controllers. In *Proc. Control 2000*, September 2000. 106, 108

[54] R.H. Istepanian, J. Wu, and S. Chen. Sparse realizations of optimal finite-precision teleoperation controller structures. pages 687–691, June 2000. 171

[55] T. Kailath. *Linear Systems*. Prentice-Hall, 1980. 40, 44

[56] H. Keding, M. Willems, M. Coors, and H. Meyr. FRIDGE : A fixed-point design and simulation environment. *EDAA*, 1998. 68, 75

[57] L. Keel and S. Bhattacharyya. Robust, fragile or optimal ? *IEEE Trans. on Automatic Control*, 42(8) :1098–1105, August 1997. 142

[58] S. Kim, KI. Kum, and W. Sung. Fixed-point optimization utility for C and C++ based digital signal processing programs. *IEEE Transactions on Circuits and Systems*, 45(11) :1455–1464, November 1998. 68, 75

[59] R. Kocik and Y. Sorel. De la modélisation à la réalisation : réduction du cycle de développement des applications temps réels distribuées. *RTS2000 8th conference on Real-time and embedded systems*, 2000. 29

[60] A.N. Kolmogorov. Three approaches to the quantitive definition of information. *Problems of Information Transmission*, 1 :1–17, 1965. 112

[61] KI. Kum, J. Kang, and W. Sung. AUTOSCALER for C : An optimizing floating-point to integer C program converter for fixed-point digital signal processors. *IEEE Transactions on Circuits and Systems*, 47(9) :840–848, september 2000. 75

[62] R. O. Lamaire and J. H. Lang. Performance of digital linear regulators which use logarithmic arithmetic. *IEEE Transactions on Automatic Control*, 31(5) :394–400, 1986. 64

[63] P. Langlois. *Outils d'analyse numérique pour l'Automatique*, chapter Introduction générale, pages 19–52. In Barraud [10], 2002. 73

[64] D. Lefebvre. *Conception de lois de commande avancées, au sein du contrôle du Groupe Moto-Propulseur, en vue de l'optimisation du confort et de la réactivité du véhicule lors des phases transitoires*. PhD thesis, Ecole Centrale de Nantes - Université de Nantes, March 2003. 27, 160, 173, 174

[65] D. Lefebvre, P. Chevrel, and S. Richard. An h-infinity based control design methodology dedicated to active control of longitudinal oscillations. In *Proceedings of the Conference on Decision and Control*, December 2001. 173

[66] D. Lefebvre, P. Chevrel, and S. Richard. An h-infinity based control design methodology dedicated to the active control of longitudinal oscillations. *IEEE Transactions on Control Systems Technology*, 11(6) :948–956, November 2003. 173

[67] V. Lefèvre. Multiplication par une constante entière : minorant de la longueur du code. Technical report, INRIA Lorraine, July 2002. 77

[68] J. Lévine. Analyse et commande des systèmes non linéaires. cours ENPC, March 2004. 44

[69] G. Li. On the structure of digital controllers with finite word length consideration. In *IEEE Transactions on Automatic Control*, volume 43, pages 689–693, May 1998. 126, 145

[70] F. Marafao, S. Deckmann, and A. Lopes. Robust delta operator-based discrete systems for fixed-point dsp implementations. *Applied Power Electronics Conference*, pages 1764–1770, 2004. 50, 148

[71] D. Ménard. *Méthodologie de compilation d'algorithmes de traitement du signal pour les processeurs en virgule fixe sous contrainte de précision*. PhD thesis, ENSSAT, december 2002. 70, 71, 75

[72] D. Ménard. Méthodologies de conversion automatique en virgule fixe pour les applications de traitement du signal. In *Ecole Thematique ARCHI03*, april 2003. 68

[73] D. Ménard, T. Saïdi, D. Chillet, and O. Sentieys. Implantation d'algorithmes spécifiés en virgule flottante dans les DSP virgule fixe. RTSI - TSI - 22/2003 Architecture des ordinateurs, 2003. 75

[74] D. Ménard and O. Sentieys. Automatic evaluation of the accuracy of fixed-point algorithms. In *Proceedings of DATE02 (Design Automation and Test in Europe)*, march 2002. 75

[75] V. Ménissier-Morain. *Arithmétique exacte : Conception, algorithmique et performances d'une implémentation informatique en précision arbitraire*. PhD thesis, Université Paris VII, 1994. 58, 73, 74

[76] R. Middleton and G. Goodwin. *Digital Control and Estimation, a unified approach*. Prentice-Hall International Editions, 1990. 48, 49

[77] R.H. Middleton and G. Goodwin. Improved finite word length characteristics in digital control using delta operators. *IEEE Transactions on Automatic Control*, 31(11) :1015–1021, November 1986. 148

[78] J.M. Muller. *Arithmétique des ordinateurs*. Etudes et Recherches en Informatique. Masson, 1989. 73

[79] J.M. Muller. A few results on table-based methods. In Tibor Csendes, editor, *Developments in Reliable Computing*, pages 279–288. Kluwer, Dordrecht, Netherlands, 1999. 29

[80] JM. Muller. Accelerating floating-point division when the divisor is known in advance. Technical report, INRIA - Arenaire Team, 2002. 77

[81] J.M. Muller. 'Partially rounded' small-order approximations for accurate hardware-oriented, table-based methods. In *Proceedings of the 16th IEEE Symposium on Computer Arithmetic*, pages 114–121, June 2003. 29

[82] C. Mullis and R. Roberts. Synthesis of minimum roundoff noise fixed point digital filters. In *IEEE Transactions on Circuits and Systems*, volume CAS-23, September 1976. 166

[83] H. Neudecker. Some theorems on matrix differentiation with special reference to kronecker matrix products. *Journal of American Statistical Association*, 64 :953–963, 1969. 185

[84] I. Njabeleke, R. Pannett, P. Chawdhry, and C. Burrows. Design of h-infiny loop-shaping controllers for fluid power systems. *IEE Colloquium Robust Control - Theory, Software and Applications*, 1997. 142, 148

[85] C. Chabot Petrault. STD : fonction estimation état de charge batterie plomb 12v (SOC). Technical Report STE 96.492.984.9A, PSA Peugeot Citroën, April 2002. 30

[86] A. Peymandoust, T. Simunic, and G. De Micheli. Low power embedded software optimization using symbolic algebra. 2001. 74, 75

[87] A. Prodić and D. Maksimović. Design of a digital PID regulator based on look-up tables for control of high-frequency DC-DC converters. In *IEEE Workshop on Computers in Power Electronics*, pages 18–22, June 2002. 29

[88] N. Revol. Stabilité des filtres IIR et arithmétique par intervalles. AS Arithmétique virgule fixe, March 2003. 75

[89] E. Rieffel and W. Polak. An introduction to quantum computing for non-physicists. *ACM Computing Surveys*, 32(3) :300–335, September 2000. 55

[90] M. Rombaut, T. Sproesser, P. Higelin, M. Basset, MA. Peltier-Dillies, N. Le Fort-Piat, G. Gissinger, JM. Blosseville, and J. Nouvier. *La voiture intelligente*. Traité IC2. G. Gissinger and N. Le Fort-Piat, 2002. 27

[91] M. Rostgaard, NK. Poulsen, and O. Ravn. A rapprochement between discrete-time operators. In *ECC93*, volume 2, pages 426–431, February 1993. 51

[92] S. Sakata. ASAMIN : a matlab gateway to l. ingber's adaptive simulated annealing (ASA). 167

[93] D. Das Sarma and DW. Matula. Faithful bipartite rom reciprocal tables. In S. Knowles and WH. McAllister, editors, *Proceedings of the 12th IEEE Symposium on Computer Arithmetic*, pages 17–28, 1995. 29

[94] P. Sendur, J. Stein, and H. Peng. Model accuracy & validation. 1999. 170

[95] R. Skelton and D. Wagie. Minimal root sensitivity in linear systems. *Journal of Guidance, Control and Dynamics*, 7(5) :570–574, 1984. 142

[96] C. Sohl. Kronecker tensor products, differentiation theory for matrices and applications. Master's thesis, Lund Institute of Technology, February 2004. 185

[97] T. Song, E. Collins, and R. Istepanian. Improved closed-loop stability for fixed-point controller realizations using the delta operator. *International Journal of Robust and Nonlinear Control*, 11 :41–57, 2001. 148

[98] A. Sripad and D. Snyder. A necessary and sufficient condition for quantization error to be uniform and white. In *IEEE Trans. on Acoustics, Speech and Signal Processing*, volume 5, pages 442–448, 1977. 71

[99] S. Stanković and D. Šiljak. Contractibility of overlapping decentralized control. *System & Control Letters*, 44(3), October 2001. 97

[100] V. Tavşanoğlu and L. Thiele. Optimal design of state-space digital filters by simultaneous minimization of sensibility and roundoff noise. In *IEEE Trans. on Acoustics, Speech and Signal Processing*, volume CAS-31, October 1984. 123, 141, 166

[101] J. Vigne. A survey of the CESTAC method. In *Proceedings of Real Numbers and Computer Conference*, 1996. 71

[102] D. Šiljak. *Decentralized Control of Complex Systems*. Academic Press, 1991. 97

[103] B. Watterson. *On n'arrête pas le progr !s*. Hors Collection, 1991. 5

[104] J. Whidborne and R. Istepanian. *Digital Controller implementation and fragility*, chapter An Evolutionary Algorithm Approach to the Design of Finite Word-Length Controller Structures, pages 143–160. In Istepanian and Whidborne [52], 2001. 170

[105] J.F. Whidborne, R. Istepanian, and J. Wu. Reduction of controller fragility by pole sensitivity minimization. *IEEE Trans. Automatic Control*, 46 :320–325, 2001. 14, 139, 142

[106] J.F. Whidborne and R.H. Istepanian. Finite word length stability issues in an L1 framework. *Int. J. Control*, 73(2) :166–176, 2000. 142, 148

[107] J.F. Whidborne, J. McKernan, and D. Gu. Kolmogorov-Chaitin complexity of linear digital controllers implemented using fixed-point arithmetic. In *8th Int. Conf. Control, Automation, Robotics and Vision*, pages 1587–1592, December 2004. 112

[108] B. Widrow. Statistical analysis of amplitude quantized sampled-data systems. *Trans AIEE*, 2(79) :555–568, 1960. 71

[109] B. Widrow, I. Kollár, and MC. Liu. Statistical analysis of amplitude quantized sampled-data systems. In *IEEE Trans. on Instrumentation and Measurement*, volume 45, pages 353–361, 1995. 71

[110] D. Williamson. Roundoff noise minimization and pole-zero sensibivity in fixed-point digital filters using residue feedback. In *IEEE Trans. on Acoustics, Speech and Signal Processing*, volume ASSP-43, October 1986. 77

[111] D. Williamson. *Digital Control and Implementation, Finite Word-length Considerations*. Prentice-Hall International Editions, 1992. 33, 59, 64, 68, 71, 77, 180

[112] D. Williamson. *Digital Controller implementation and fragility*, chapter Implementation of a Class of Low Complexity, Low Sensitivity Digital Controllers using Adaptive Fixed-Point Arithmetic, pages 43–74. In Istepanian and Whidborne [52], 2001. 181

[113] D. Williamson and S. Sridharan. Residue feedback in digital filters using fractional feedback coefficients. In *IEEE Trans. on Acoustics, Speech and Signal Processing*, volume 33, pages 477–483, April 1985. 77

[114] J. Wu and S. Chen. Design of sparse digital finite-precision controller structures based on an improved closed-loop stability related measure. pages 313–318, February 2002. 171

[115] J. Wu and S. Chen. *Unsolved Problems in Mathematical Systems and Control Theory*, chapter Stable controller coefficient perturbation in floating point implementation, pages 280–284. Princeton University Press, 2004. 143

[116] J. Wu, S. Chen, and J. Chu. Comparative study on finite-precision controller realizations in different representation schemes. *9th Annual Conf. Chinese Automation and Computing Society in UK (Luton, UK)*, September 2003. 108, 112

[117] J. Wu, S. Chen, R. Istepanian, and J. Chu. Shift and delta operator realizations for digital controllers with finite-word-length considerations. *IEE Proc. Control Theory and Applications*, 2000. 48, 147, 148

[118] J. Wu, S. Chen, G. Li, and J. Chu. Constructing sparse realizations of finite-precision digital controllers based on a closed-loop stability rela-

ted measure. *IEE Proc. Control Theory and Applications*, 150(1) :61–68, January 2003. 14, 139, 142, 147, 148, 150, 167, 171

[119] J. Wu, S. Chen, G. Li, R. Istepanian, and J. Chu. An improved closed-loop stability related measure for finite-precision digital controller realizations. *IEEE Trans. Automatic Control*, 46(7) :1162–1166, 2001. 144

[120] J. Wu, S. Chen, J.F. Whidborne, and J. Chu. A unified closed-loop stability measure for finite-precision digital controller realizations implemented in different representation schemes. In *IEEE Trans. Automatic Control*, volume 48, pages 816–823, May 2003. 107, 108, 151

[121] J. Wu, S. Chen, J.F. Whidborne, and J. Chu. Optimal realizations of floating-point implemented digital controllers with finite wordlength considerations. *International Journal of Control*, 77(5) :427–440, 2004. 108

[122] V. Zimpfer. *Amélioration du traitement numérique des signaux dans les systèmes actifs en protection auditive*. PhD thesis, Institut National des Sciences Appliquées de Lyon, december 2000. 77

[123] V. Zimpfer and K. Buck. Procedé de filtrage à large dynamique pour filtre numérique récursif implanté dans un processeur de signal DSP travaillant avec des nombres entiers. Demande de brevet d'invention - INPI, April 2000. 77

Index

Titre
Analyse et synthèse de l'implémentation de lois de contrôle-commande en précision finie — Étude dans le cadre des applications automobiles sur calculateur embarqué.

Résumé
Cette thèse CIFRE, réalisée en collaboration industrielle entre l'IRCCyN et PSA Peugeot-Citroën, s'intéresse à l'aspect numérique de l'implémentation, au sein de calculateurs embarqués, de lois de contrôle/commande (provenant de l'automatique ou du traitement du signal) sous les contraintes de précision finie.
Le processus d'implémentation amène de nombreuses dégradations de la loi et nous nous intéressons plus particulièrement à la quantification des coefficients intervenant dans les calculs.
Pour une loi (filtre ou régulateur) donnée, il existe une infinité de réalisations numériques possibles qui, bien que mathématiquement équivalentes, ne le sont plus en précision finie : de nombreuses réalisations équivalentes existent : forme d'état, réalisations en delta, formes directes, structures retour d'état observateur, décompositions en cascade, en parallèle, etc.
Après avoir présenté ces différentes possibilités, ce mémoire de thèse, propose un formalisme mathématique – la forme implicite spécialisée – qui permet de décrire de manière unifiée un ensemble élargi d'implémentations. Celui-ci, bien que macroscopique, permet d'exprimer précisément les calculs à réaliser et les paramètres réellement mis en jeu. Différentes mesures, appliquées à ce formalisme et qui permettent d'évaluer l'impact de la quantification (en virgule fixe et virgule flottante) et d'analyser la dégradation induite, sont ensuite proposées.
Via un problème d'optimisation, la réalisation qui présente la meilleure robustesse face aux détériorations induites par les processus d'implémentation en précision finie est trouvée.

Title
Analysis and Synthesis of the Finite Word Length Implementation of linear controllers or filters — Application to embedded automotive control.

Abstract
These thesis, done in an industrial context with PSA Peugeot-Citroën and IRCCyN, deals with the numerical aspect of the implementation of filters or controllers in embedded processors.
The implementation of a controller or a filter in a Finite Word Length context may lead to a deterioration of the global performance, due to parametric errors and numerical noises. We focus here on the effect of the quantization of the embedded coefficients.
It exists an infinity of equivalent relalizations of a given controller, and these realizations are no more equivalent in finite precision : classical state-space realizations, delta-realizations, direct forms, observer-state feedback, cascade or parallel decomposition, etc.
After having exhibits theses possibilites, this PhD thesis proposes an unifying framework – the implicit specialized state-space – that encompasses usual realizations (and many others unexplored). This specialized form, but still macroscopic, is directly connected to the in-line calculations to be performed and the involved coefficients. Various analysis tools, applied to our formalism, may be used to determine how the realization will be altered during the FWL process (with floating point and fixed point considerations).
Then, according to these tools, optimal realizations with the best FWL robustness can be found, via an optimization problem.