THESE

présentée par

Remis BALANIUK

pour obtenir le grade de DOCTEUR

de l'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

(Arrête ministériel du 30 mars 1992)

Spécialité : **Informatique**

Identification Structurelle

Date de soutenance : 3 septembre 1996

Composition du jury :

Président et rapporteur :	Philippe CINQUIN
Rapporteur :	Raja CHATILA
Examinateurs :	Daniel BRUN-PICARD
	Bernard ESPIAU
	Emmanuel MAZER

Thèse préparée au sein du laboratoire GRAVIR - INRIA Rhône Alpes
sous la direction de Emmanuel MAZER

Résumé

Dans ce mémoire nous proposons une méthode originale d'acquisition de modèles : **l'identification structurelle**. Nous nous plaçons dans un cadre intermédiaire entre les méthodes de modélisation classiques et les méthodes basées sur l'apprentissage. Nous montrons que pour le cas d'une classe particulière mais assez générale de fonctions il est possible d'inférer automatiquement la forme d'équation qui représente au mieux un certain processus physique, évitant ainsi l'effort de caractérisation du modèle par le concepteur. L'acquisition des modèles est faite suivant un protocole expérimental dans lequel l'identification de paramètres est restreinte à des problèmes n'ayant qu'une seule dimension d'entrée, diminuant ainsi la quantité de données requises. Les modèles générés par la méthode sont facilement différentiables, améliorables et réutilisables. La méthode peut être particulièrement utile en robotique où l'on rencontre souvent le type fonctionnelle considéré.

Abstract

In this thesis we propose an original method to models acquisition : **the structural identification**. Our work can be situated somewhere between classical modelisation methods and learning based methods. We show that in the scope of a particular but quite general functional class it is possible to automatically choose the best equation form to represent a physical process, avoiding the hard work of model caracterisation that the designer should do. Our method uses an experimental protocol where the parameters identification is limited to one entry dimension problems, reducting the amount of data required by the modelisation. The models generated by our method can be easily differentiated, corrected and reused. The method can be particularly useful in robotics where the functional form the method hands can be easily found in many kinds of problems.

Remerciements

Je doit tout d'abord remercier mon pays, le Brésil, qui à travers du financement du CNPQ a rendu possible mon séjour en France et la réalisation de cette thèse.

Je tiens à remercier particulièrement à Emmanuel Mazer, qui a encadré et soutenu mon travail de thèse. La réussite de ce travail est en grande partie due a la liberté de travail qu'il m'a accordé, son enthousiasme, sa patience et son amitié, ainsi qu'a ses compétences scientifiques.

Je remercie aussi le laboratoire GRAVIR (ex-LIFIA) de m'avoir accueilli durant ces quatre ans.

Je remercie également Messieurs Philippe Cinquin, Raja Chatila, Daniel Brun-Picard et Bernard Espiau qui m'ont fait l'honneur de participer au jury de ma thèse, et plus particulièrement aux deux premiers pour avoir accepté d'être les rapporteurs de mon manuscrit.

Je tiens ensuite à exprimer mes remerciements et ma plus vive gratitude à :

- Christian Laugier, responsable de l'équipe SHARP,
- Pierre Bessière, Eric Dedieu et Olivier Lebeltel, de l'équipe Laplace, pour leur amitié et pour l'aide précieuse qu'ils m'ont apporté à la réalisation de ce manuscrit et à la préparation de ma soutenance de thèse,
- toute l'équipe SHARP et son environnement chaleureux, particulièrement à Alberto Munoz, Danièle Herzog, Ammar Joukhadar, Philippe Garnier, Cyrill Novales, Fernando DelaRosa, Anton Deguet et Isabelle Mazon pour les nombreux coups de main,
- Radu Horaud et son équipe pour le système expérimentale d'asservissement visuel.

Enfin, j'adresse un grand merci au nombreux groupe de "amigos brasileiros" à Grenoble avec qui j'ai pu partager autant de bons moments pendant ces années de séjour en France, particulièrement à Alba, Geraldo, Osorio, Alvaro, Jaime, Sueli, Jose Celso, Néia, Javam, Kita, Mari, Silvio, Marcelo, Benhur, Paulo Fernandes, Marilia, Carissimi, Joao, Pida, Marilena et Ana Elisabete.

A mes parents Pedro et Mirsa,
A mon fils Rafael,
A mes sœurs.

Table des matières

Liste des figures

Liste des tables

Chapitre 1

Introduction

Dans ce mémoire nous nous intéressons au problème de la modélisation en robotique. Il s'agit globalement d'établir des programmes d'ordinateur permettant de prédire la valeur de certaines variables d'un système robotique en fonction d'autres variables de ce même système.

Par exemple, on voudra prédire la position cartésienne de l'extrémité d'un bras manipulateur en fonction de la position de chacun de ses axes. Dans une expérience plus complexe où le bras porte une caméra, on établira la relation entre la position des degrès de liberté et la position d'indices visuels dans l'image vidéo. L'obtention de ces modèles a pour objectif le contrôle de ces systèmes, ainsi on cherchera à piloter le bras pour obtenir une position cartésienne donnée ou pour obtenir une configuration particulière d'indices visuels dans l'image.

Nous connaissons aujourd'hui trois grandes approches au problème de la modélisation :

- **La modélisation analytique sans retour d'expérience :** de par ses connaissances préalables, le concepteur possède l'ensemble des informations nécessaires à la construction de son programme. Par exemple, il connait l'ensemble des paramètres et des lois lui permettant de passer de la position des actionneurs à celle de l'extrémité du mécanisme considéré.

- **La modélisation analytique avec retour d'expérience :** dans ce cas on connaît les lois fixant les relations entre les variables mais on choisit d'inférer à partir de mesures expérimentales la valeur de certains des paramètres. Par exemple, dans une opération de calibrage on cherchera à déterminer par un ensemble de mesures la longueur des segments mécaniques d'un robot, les zéros de ses codeurs ou les paramètres intrinsèques d'une caméra.

- **La modélisation par "boîte noire" avec retour d'expérience :** on ne connaît pas les lois du système et on ne s'intéresse qu'aux valeurs des variables. Par une phase d'apprentissage, on cherche à adapter un modèle universel dont on va régler les paramètres.

En fait, l'ensemble de ces approches peuvent être mesurées à l'aune des connaissances préalables du modélisateur : de la connaissance parfaite à l'absence d'information a priori.

Mais il faut bien reconnaître que dès lors que l'on s'intéresse à faire entrer des données expérimentales dans un processus de modélisation, les outils à notre disposition sont ceux de l'identification et en fin de compte ceux de la minimisation. Les connaissances préalables du modélisateur servent à définir la fonction qui sera utilisée pour le modèle. Ansi le modélisateur utilisera une méthode des moindres carrés ou un filtre de Kalman pour déterminer les paramètres d'une équation de mesure ou, s'il ne possède aucune information, il utilisera une procédure de propagation arrière dans un réseau de neurones multi-couche pour déterminer les poids affectés aux arcs de ce réseau. Dans les deux cas il se sert de sa connaissance (ou de son absence de connaissance) pour fixer la fonction dont il cherche à déterminer les paramètres. Nous disons qu'il choisit la structure de son modèle.

Dans ce mémoire nous présentons une alternative à cette approche en définissant une méthode qui présente l'originalité de **ne pas définir explicitement la fonction utilisée dans le processus de modélisation**. Nous appelons cette méthode **l'identification structurelle** car elle permet d'inférer certaines règles de calcul de la fonction à partir des données expérimentales. Ainsi le résultat de notre méthode n'est pas un vecteur dans l'espace des paramètres à estimer mais un programme au sens informatique du terme.

D'un point de vue méthodologique notre contribution est d'avoir montré qu'il existe des connaissances préalables permettant de mettre en oeuvre des procédés de modélisation se situant entre les approches de type "boîtes noires" et les approches utilisant des équations de mesure.

D'un point de vue théorique notre contribution est d'avoir mis en évidence une transformation fonctionnelle originale permettant la décomposition récursive d'une certaine classe de fonctions. Notre méthode est basée sur cette décomposition. Elle fait l'hypothèse que l'appartenance des fonctions de modélisation à cette classe fait partie des connaissances préalables du modélisateur.

D'un point de vue pratique les avantages de notre méthode sont les suivants :

- La classe des fonctions considérée est suffisamment large pour en permettre l'application à nombreux problèmes et notamment en robotique,
- Les modèles peuvent être obtenus à partir d'un nombre restreint d'expériences, ce nombre varie quasiment linéairement avec le nombre de variables d'entrées,

- ces modèles sont exacts dans le cas de données non bruitées,
- ils permettent de calculer l'incertitude,
- ils sont différentiables et permettent le contrôle des systèmes,
- ils peuvent être réutilisés en cas de modification du système. Dans ce cas ils en retiennent la structure et peuvent s'adapter facilement aux nouvelles conditions d'expérimentation.

D'un point de vue expérimental nous avons réalisé une implantation informatique de notre méthode suivie des expérimentations suivantes :

- nous l'avons vérifié sur un cas simulé ou l'on connait la fonction que l'on cherche à identifier (la cinématique directe d'un bras manipulateur à six degrés de liberté);
- nous l'avons utilisé pour obtenir le modèle différentiel reliant les variables articulaires aux positions d'indices visuels dans une image et à partir de cela faire l'asservissement visuel d'un bras à six degrés de liberté.

Plan du mémoire

Nous commençons par le chapitre 2 où nous proposons une étude à propos de la pratique de la modélisation. Le chapitre a trois buts principaux :

- caractériser le domaine de recherche d'acquisition de modèles, son importance comme discipline scientifique, ses concepts principaux et ses limitations;
- présenter une vision générale et structurée du domaine, en introduisant ses principaux axes de recherche et en analysant quelques méthodes qui caractérisent bien chacun de ces axes de recherche;
- mettre en évidence les points communs à la plupart des méthodes de modélisation existantes, même s'ils appartiennent à des axes de recherche différents:
    - ⋆ le rôle central du concepteur dans la modélisation,
    - ⋆ le besoin d'une structure préconçue pour le modèle avant même l'utilisation des méthodes de modélisation,
    - ⋆ l'attention donnée presque exclusivement à l'identification de paramètres au détriment des autres étapes de la modélisation.

Le chapitre 3 présente le concept d'identification structurelle et le situe par rapport aux méthodes de modélisation présentées au chapitre précédent. Nous montrons qu'il est possible, sous certaines conditions, d'inférer automatiquement la structure d'un modèle. Ce chapitre présente aussi la méthode qui fait l'objet de cette thèse. Nous montrons que la méthode peut modéliser toute fonction appartenant à un certain ensemble. Nous abordons encore quelques points pratiques de la méthode, comme l'identification des paramètres de fonctions partielles, et présentons aussi l'algorithme de la méthode.

Le chapitre 4 montre comment utiliser les modèles acquis par notre méthode, et si nécessaire comment les transformer. Un des avantages de notre approche est le fait de construire des modèles ouverts et simples, sur lesquels il est possible d'appliquer une série d'opérations bien définies. On a accès à toutes les données internes du modèle, et cela nous permet de calculer les dérivées partielles, ou les marges d'erreur. Ces informations nous permettent d'inverser le modèle ou de le corriger. Nous proposons aussi une méthode de transformation des modèles existants permettant de réduire le nombre de saisies expérimentales.

Le chapitre 5 illustre l'application de la méthode et montre sa mise en œuvre dans le cadre d'applications à la robotique. Nous avons appliqué la méthode de modélisation à des problèmes simulés sous plusieurs conditions différentes : avec et sans caractérisation idéale, en manipulant des données expérimentales bruitées et non bruitées. Nous avons aussi appliqué la méthode à un problème réel en robotique: la modélisation d'un système intégré "vision-robotique" et son asservissement. Les résultats expérimentaux démontrent l'utilité de la méthode.

Enfin, la conclusion présente un résumé du travail accompli et des résultats obtenus ainsi qu'une vue prospective.

Chapitre 2

Revue des travaux

Ce chapitre a pour objectif de donner une vision générale et structurée du domaine de recherche de l'acquisition de modèles. Le chapitre est organisé de la façon suivante :

- nous commençons en donnant une caractérisation du domaine dans le paragraphe 2.1;
- ensuite, dans les paragraphes 2.2 et 2.3 nous présentons deux des points de vue qui a notre avis caractérisent les façons d'attaquer le problème de l'obtention de modèles;
- nous terminons ce chapitre en faisant une synthèse des divers points abordés dans le paragraphe 2.4.

2.1 Caractérisation du domaine

Richalet [Ric91] définit le but de la modélisation comme suit:

"Le but de la science a toujours été de tenter de plaquer sur une réalité physique mesurable d'un monde extérieur sensible, une représentation rationnelle, en fait logico-mathématique. L'objectif de la modélisation n'est donc pas nouveau et s'il doit être rattaché à celui des Sciences expérimentales (Galilée), la modélisation moderne apporte cependant une contribution spécifique par :

- **La rationalisation de la démarche**, *maintenant dotée de méthodes justifiées théoriquement et implantables dans des calculateurs numériques puissants dont l'efficacité technologique, permet de réaliser des traitements auparavant inaccessibles.*

- **Une réflexion maintenant dépassionnée et plus objective sur la connaissance.** *Qu'est-ce-que connaître ? Que signifie représenter un processus physique ? L'analyse physique telle que nous l'enseigne la démarche analytique cartésienne de la mise en équations à l'aide des "lois de la physique", est-elle le seul mode de représentation du monde ? S'il y en a d'autres, lequel choisir devant un problème donné ? A ces questions sur lequelles se sont heurtées des écoles de pensée, la modélisation apporte des réponses dont l'efficacité heuristique, c'est-à-dire dans l'acception première du terme, qui permet d'apprendre ou de découvrir, est prouvée dans la pratique professionnelle quotidienne" [Ric91].*

Dans ce paragraphe nous analyserons la pratique de la modélisation. Notre objectif ici est d'introduire ses concepts principaux, ses classifications et ses étapes.

Avant de parler de modèles et des différentes façons de les obtenir on doit introduire les concepts de "variable", d'"état" et de "structure". La séparation entre ces trois concepts est à la base de la Théorie des Systèmes ([Ric91]):

- les **VARIABLES** sont des mesures effectuées sur le processus physique.
- l'**ÉTAT** du système : on suppose qu'il existe un ensemble de variables par lequel on peut définir complètement le système. L'ensemble des valeurs prises par ces variables constitue l'**état**.
- la **STRUCTURE** : est le *lien fonctionnel qui met en relation un ensemble d'éléments et qui rend les variables non indépendantes. C'est sur cette structure supposée que l'on va plaquer une représentation logico-mathématique rationnelle. Suivant une démarche scientifique classique, cette représentation va être faite avec des équations de type approprié qui contiennent des paramètres [Ric91].*

La modélisation fait l'hypothèse de l'existence d'une relation mathématique qui permet de calculer les valeurs des variables de sortie du modèle à partir de l'état du système, des valeurs des variables d'entrée et des valeurs des paramètres à trouver. Cette relation devra être capable de simuler pratiquement le processus. On pourra donc à partir d'un ensemble de paramètres choisi et d'un enregistrement de données expérimentales réelles (les entrées), effectuer une simulation du modèle et comparer, conformément en cela à la définition du modèle, les comportements du processus physique objet et de son modèle mathématique.

La modélisation a, en première compréhension, deux extrêmes ([Ric91]):

- les **modèles de connaissance** où la représentation des phénomènes est faite avec l'aide des lois de la physique;

- les **modèles de représentation** où l'on utilise une représentation mathématique du type "boîte noire", un outil mathématique où les paramètres sont identifiés en utilisant des données expérimentales.

Le modèle de connaissance est valable dans un domaine large, et les paramètres qui interviennent dans les équations définissant la structure sont *réifiables*, c'est-à-dire qu'ils ont un "sens physique". Il serait idéal s'il n'était pas lourd, cher et difficile à établir. Le modèle de représentation, par contre, est facile à établir, mais limité. Ses paramètres n'ont pas un sens physique et n'ont qu'une valeur locale. Cependant un modèle de représentation efficace dans un domaine de fonctionnement donné peut être suffisant. Si on considère le rapport *effort de modélisation/capacité de prédire le comportement* un modèle de représentation peut être plus efficace qu'un modèle de connaissance.

Il ne faut pas opposer ces deux types de modélisation et tout l'art du modélisateur va justement être de savoir manipuler les deux concepts et de trouver un compromis. En effet, rares sont les cas où le modèle est purement de connaissance ou purement de représentation. On ne s'interdit pas d'avoir des connaissances scientifiques antérieures sur le processus et la modélisation ne se réduit pas à l'identification d'une formule empirique par une technique d'estimation statistique. Il faut utiliser toute information disponible au préalable pour réduire le processus en sous-systèmes, choisir les variables à identifier sur une base physique, etc [Ric91].

La modélisation apparaît souvent aux non initiés comme se réduisant à un problème d'identification, d'analyse de données ou de "calage de modèle". Les difficultés viennent de la nature non-homogène du problème où se mélangent des considérations structurelles, des méthodes de minimisation, des techniques de recherche de protocoles d'essais.

Il convient de distinguer dans la modélisation trois grandes étapes: la caractérisation, l'identification et la vérification.

- la **caractérisation** est l'étape pendant laquelle on va structurer le modèle mathématique. On doit choisir une classe de modèles pour représenter l'objet. C'est une étape qualitative où la caractérisation de la connaissance préalable va nous aider à utiliser au maximum les informations a-priori disponibles. Le modèle mathématique choisi va être une structure dépendant d'un vecteur de paramètres inconnus.

- l'**identification** du modèle consiste à faire l'adaptation numérique des paramètres, le calage numérique du modèle sur les données expérimentales. Le but est de diminuer par une stratégie quelconque la distance entre les comportements de l'objet et du modèle. Cette distance minimale ne sera en général pas nulle car la caractérisation n'est jamais parfaite, les mesures sont perturbées par les bruits et les traitements numériques se font avec une précision de calcul limitée.

- la **vérification** permet de tester la validité du modèle. Pour cela on choisit une métrique et on mesure la distance entre les comportements de l'objet et du modèle au cours d'un certain nombre d'essais. Ces mesures seront ensuite comparées à une qualité requise.

Il existe bien des façons d'attaquer le problème d'obtention d'un modèle. De nombreuses méthodes vont s'attaquer particulièrement au sous-problème de minimisation de la distance des comportements objet-modèle, et vont le résoudre dans leur domaine de validité spécifique. Cependant, on peut distinguer clairement deux directions dans ce domaine: la modélisation dite "classique", basée sur les connaissances préalables du concepteur (plus proche d'un modèle de connaissance) et la modélisation dite "adaptative", basée sur les données expérimentales (plus proche d'un modèle de représentation). Pour un étude de la théorie et des méthodes du domaine de modélisation clasique voir [Lju87] et [Ric91]. Dans les paragraphe 2.2 nous analyserons la modélisation adaptative.

2.2 Les méthodes adaptatives

L'objectif de ce paragraphe est d'analyser les méthodes de modélisation adaptatives, aussi connues comme les méthodes "boîte noire". Nous voulons aussi mettre en évidence l'utilité de ces méthodes dans les cas où les informations préalables disponibles sur le processus à modéliser sont très limitées ou quand le système évolue dans des conditions incertaines. Dans ces cas il est impossible ou impraticable de dériver d'avance une structure analytique efficace avec propriétés fixes pour le modèle. La structure générale des méthodes adaptatives, dotant le système d'une forme d'apprentissage, permet une solution à ce type de problème.

Voir [SZL+95] pour une revue des méthodes de modélisation du type boîte noire, et [JHB+95] pour un étude des bases mathématiques de ces méthodes.

Les modèles adaptatifs sont le plus souvent des *modèles de pure représentation*. La modélisation avec une méthode adaptative réduit l'étape de caractérisation aux choix et configuration de l'outil adaptatif. Cette configuration peut être assez complexe. Elle ne dépendra pas directement du processus à modéliser, mais plutôt des données qu'il faudra manipuler pendant l'étape d'identification. On sait d'une façon générale que la caractérisation est la phase la plus spécifique et par là même la plus coûteuse de toute modélisation. Cette représentation apparaît donc comme très "économique". Elle va, au prix de plus de calcul, éviter de dépenser du temps de réflexion et être ainsi accessible à tous.

Le paradigme de base des méthodes adaptatives est qu'un système à modéliser peut être vu simplement comme une fonction entre ses entrées et ses sorties [BF92]. Une seule hypothèse préalable est établie à propos de cette fonction : si les entrées et les sorties du système sont représentées par des valeurs continues la fonction doit avoir une forme continue. Cette fonction continue peut être représentée par une hypersurface liant l'hyperespace d'entrée à l'hyperespace de sortie [PG89]. Pour une méthode connexioniste, "apprendre" c'est construire une fonction qui soit une bonne reproduction de la vraie hypersurface définie par le système à modéliser. La construction de cette fonction est faite par la modification d'une fonction initiale au fur et à mesure que les données sont fournies. Cette construction se fait par adaptation automatique des paramètres de la structure.

Le problème de l'apprentissage d'une fonction entre un espace d'entrée et un espace de sortie équivaut au problème de synthèse d'une mémoire associative qui retrouve la bonne sortie quand une entrée connue est présentée, et qui *généralise* quand une entrée inconnue est présentée. Il équivaut aussi au problème d'estimation d'un modèle qui transforme entrées en sorties étant donné un ensemble de paires entrées-sorties comme exemple [PG89]. Un cadre classique pour ce type de problème est la *théorie de l'approximation*.

La théorie de l'approximation traite le problème d'*estimer* ou d'*interpoler* une fonction continue à plusieurs variables $f(X)$ par une fonction $F(W, X)$ ayant un nombre de paramètres fixe W (X et W sont deux vecteurs réels: $X = x_1, x_2, ..., x_n$ et $W = w_1, w_2, ..., w_m$). On choisit une fonction F appartenant une famille de fonctions interpolatrices, et le problème de l'estimation devient alors celui de choisir l'ensemble de paramètres W qui va permettre la meilleure approximation possible de f dans l'ensemble d'exemples. Une famille de fonctions interpolatrices a, en théorie, la propriété de pouvoir approximer toute fonction continue. Cette universalité fait la différence entre les méthodes classiques (où le choix d'une forme fonctionnelle est spécifique au problème traité) et les méthodes adaptatives (où une seule famille interpolatrice peut servir des problèmes de modélisation de nature differentes).

On distingue les méthodes adaptatives *paramétriques* et *non paramétriques*. Une méthode *paramétrique* consiste à déterminer un ensemble fini de paramètres libres d'une fonction mathématique afin que la fonction s'accorde aux données. La fonction est un modèle global qui prend en compte toutes les données en même temps. Parmi les méthodes adaptatives paramétriques, on peut citer les réseaux neuronaux [RMG86], les fonctions radiales de base [PG89], les cartes "topology-conserving" [RMS89], les réseaux "wavelet" [ZB92]. Une méthode *non paramétrique* est aussi une fonction mathématique qui contient un ensemble de paramètres libres. Cependant le nombre et les valeurs des paramètres sont déterminés à chaque utilisation de la méthode à partir d'un sous ensemble local des données. Parmi les méthodes adaptatives non paramétriques, on peut citer les méthodes "memory-based" [Atk90] [SA94], les méthodes de voisinage "n-nearest" [MHJ92].

Dans ce paragraphe nous allons nous limiter à l'analyse du sous domaine des méthodes adaptatives paramétriques. Nous commençons en présentant les réseaux neuronaux dans le paragraphe 2.2.1. Dans le paragraphe 2.2.2 nous analysons la robotique connexionniste. Nous terminons ce paragraphe en analysant les limitations des méthodes adaptatives dans le paragraphe 2.2.3. Cette étude est basée sur le rapport [BMA93] dans lequel nous analysons l'état de l'art de la robotique connexioniste.

2.2.1 Les réseaux neuronaux

Parmi les méthodes adaptatives paramétriques, les réseaux neuronaux (ou systèmes connexionistes) sont devenus les plus populaires. Utilisés comme outil de modélisation les réseaux neuronaux peuvent apporter des résultats positifs rapidement et sans beaucoup d'effort. Les termes et quelques concepts inspirés de la biologie attirent aussi la curiosité, et on a vu naître dans grand nombre de domaines scientifiques un important effort de recherche basé sur la modélisation connexionniste.

En fait, la plupart des techniques d'approximation peuvent être présentées sous la forme d'un réseau qui peut imiter un "réseau neuronal" [1]. Dans le contexte de notre analyse, un réseau est une fonction représentée par la composition d'un grand nombre de fonctions de base. Un réseau, après tout, peut être une bonne représentation graphique pour une grande classe d'algorithmes.

Dans le domaine traditionnel des réseaux neuronaux, on trouve parmi les méthodes les plus populaires: les perceptrons multicouches (et leur algorithme d'apprentissage "back-propagation") [RMG86], les perceptrons multicouches récurrents, les réseaux de Hopfield [Kha90], les cartes de Kohonen [Koh82], la machine de Boltzmann [Kha90]. Ces méthodes ont en commun seulement le fait d'utiliser la fonction sigmoïde comme fonction de base dans des compositions représentées sous la forme de réseau. Chaque méthode a son algorithme pour trouver l'ensemble de paramètres W optimal.

D'autres méthodes "non-neuronales,", comme le GRBF (fonctions radiales de base généralisées) [PG89] ou les réseaux "wavelets" [ZB92], sont représentées aussi sous la forme de réseau mais en utilisant une fonction de base différente.

L'architecture la plus utilisée parmi les réseaux neuronaux est celle des réseaux multicouche (figure 2.1) dont l'apprentissage est effectué à l'aide de l'algorithme

[1] Plusieurs cas parmi les réseaux neuronaux devraient s'appeler réseaux non-neuronaux, car leur lien aux neurones biologiques, dans le meilleur des cas, est faible

de la rétro propagation du gradient (back-propagation) [RMG86], et qui peut être représentée par le schéma suivant :

$$F(X,W) = g(\sum_n w_n.g(\sum_i w_i.g(...g(\sum_j w_j.x_j)...))) \quad \forall X \in \Re^n \tag{2.1}$$

Dans le cas de la fonction ci-dessus, il s'agit d'un réseau à n couches dont $n-2$ couches cachées. Les unités (neurones) de chaque couche calculent leurs entrées comme une combinaison linéaire des sorties des unités de la couche précédente pondérées par les poids des connexions W qui lient les unités des couches successives. Cette entrée est ensuite transformée en sortie par l'intermédiaire d'une fonction d'activation sigmoïde g. Ce type de fonctions a connu un succès considérable dans le domaine des réseaux neuronaux même s'il n'est pas courant dans la littérature relative à la théorie de l'approximation [PG89].

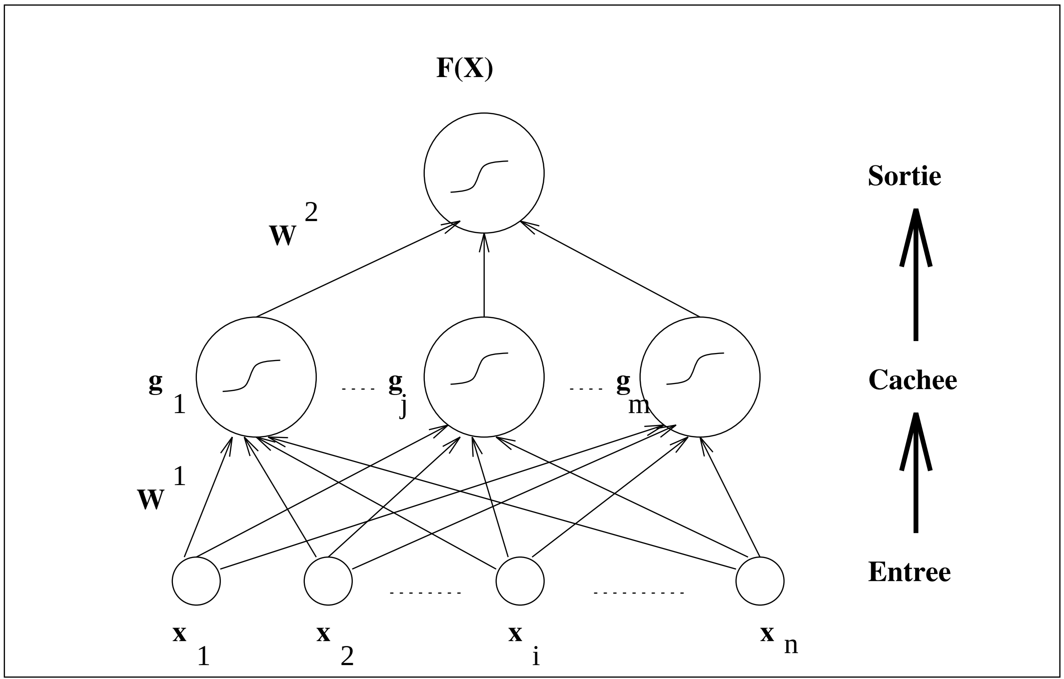

Figure 2.1 : Réseau neuronal à trois couches et à base de sigmoïdes

Les résultats théoriques prouvent que ces réseaux à couches ont la capacité d'estimer avec une erreur arbitraire toute fonction continue, pourvu que le réseau contienne un nombre suffisant de paramètres [HN89] [Fun89].

2.2.2 La robotique connexionniste

Le but de ce paragraphe est d'illustrer et de mieux comprendre les classes d'application pour lesquels les méthodes adaptatives peuvent être utiles. Comme pour le paragraphe précédente nous allons nous limiter aux méthodes connexionistes, et pour pouvoir analyser le sujet un peu plus en détail nous limiterons l'analyse à l'utilisation des réseaux neuronaux à la robotique.

Il reste entendu que lorsque l'on parle dans ce paragraphe de ce que ne peuvent pas faire les réseaux neuronaux, on se réfère aux méthodes actuellement existantes et étudiées. On ne peut préjuger de ce que seront les méthodes futures, ni des résultats à venir des recherches en cours sur certains aspects encore mal compris des réseaux.

Les réseaux neuronaux sont devenus sujet de recherche obligatoire dans la robotique. La quantité importante de problèmes encore sans solution, la complexité de ces problèmes, la nature des données manipulées, définissent un cadre parfait pour que l'appel séduisant des réseaux neuronaux soit irrésistible. La robotique connexionniste s'insère dans le domaine de recherche appelé "machine learning", qui recherche les solutions non-conventionnelles à ces problèmes.

Traditionnellement les roboticiens utilisent les méthodes d'identification de paramètres (optimisation paramétrique). Cependant, pour plusieurs problèmes de robotique, comme par exemple l'obtention d'une matrice d'inertie, la complexité de l'analyse peut rendre la construction d'un modèle analytique impraticable, sinon presque impossible. Dans tous les cas les modèles analytiques s'avèrent être des simplifications d'un monde réel trop complexe, et le roboticien traditionnel se voit obligé de conditionner l'environnement où son robot doit évoluer pour que son modèle soit valable.

De nombreux chercheurs ont été attirés par la possibilité de remplacer les lourds modèles analytiques par les boîtes noires des réseaux neuronaux. Ils imaginaient qu'il suffirait de mettre à l'intérieur les données réelles, bruitées, pour définir ainsi un modèle encore capable de continuer à apprendre devant les imprévus du monde réel.

Ce sont les principales propriétés des réseaux neuronaux qui suggèrent les applications possibles en Robotique [BMA93]:

- les possibilités d'auto-organisation et d'apprentissage propres aux réseaux neuronaux les rendent potentiellement adaptés au problème de l'apprentissage des robots (robot learning). Ce problème est caractérisé par l'obligation de prendre en compte des données sensorielles bruitées, des erreurs de contrôle, des environnements changeant de manière dynamique et des situations où il est nécessaire d'apprendre directement par l'expérience. C'est sans doute là que se portent aujourd'hui les principales recherches dans le domaine de la robotique connexionniste. En fait, comme le signalent plusieurs auteurs, la propriété importante des réseaux est d'être des estimateurs adaptatifs de fonctions non-linéaires. Cette propriété permet de les utiliser comme des classifieurs dotés de capacités d'apprentissage et de généralisation (interpolation et extrapolation), et surtout comme des procédures générales d'apprentissage de fonctions non-linéaires. Cette dernière fonction des réseaux les rend particulièrement adaptés à la résolution d'un problème typique de la Robotique, celui de l'apprentissage de comportements construits sur des mises en correspondance entre les données capteurs et actionneurs.

- les réseaux récurrents permettent de mémoriser des séquences temporelles, et donc des suites discrètes d'actions à exécuter.

- les algorithmes parallèles d'optimisation issus du connexionnisme, tels que les réseaux récurrents pour l'optimisation ou les algorithmes génétiques, peuvent être utilisés pour résoudre les problèmes d'optimisation rencontrés en robotique. C'est le cas dans le domaine de la recherche de trajectoires.

- les propriétés neuromimétiques des réseaux neuronaux permettent de les appliquer à la programmation et au contrôle des robots en utilisant les résultats des recherches menées en psychophysiologie et en neurophysiologie sur les traitements sensori-moteurs mis en œuvre dans les activités cognitives animales.

- la propriété dite d'unicité de l'architecture fonctionnelle des réseaux (dans un réseau, c'est la même structure physique qui apprend, s'organise, mémorise, traite les données d'entrées et généralise aux cas non-appris) ramène à l'idée d'indissociabilité des différentes fonctions d'un système cognitif (perception, décision, action), et donc à l'indissociabilité des études de ces fonctions. L'utilisation de cette propriété paraît être un moyen de résoudre le problème posé par ce que l'on appelle en IA la fragilité des systèmes symboliques (brittleness), et qui est leur incapacité à prendre en compte simultanément tous les aspects du raisonnement d'une manière systématique et à l'intérieur d'un cadre unifié.

- enfin les réseaux neuronaux, parce qu'ils permettent de revenir au concept plus large de réseau d'automates par le biais de ce que l'on appelle aujourd'hui le macro-connexionnisme, conduisent à faire le lien avec les théories de l'émergence comportementale étudiées aujourd'hui en robotique et en intelligence artificielle distribuée (théorie des réseaux d'agents réactifs).

Cependant, malgré le grand nombre d'exemples d'applications prometteuses, il faut reconnaître que l'utilisation des réseaux neuronaux dans la robotique vit encore dans son moyen-âge [Ber93]. Il n'existe pas encore de systèmes robotiques neuronaux, seulement des solutions satisfaisantes pour quelques problèmes spécifiques. D'une façon générale les réalisations expérimentales sont faites pour être démonstratives d'un concept mais n'indiquent jamais un niveau de maturité "préindustrielle" [BMA93].

La recherche dans le domaine de la robotique connexioniste est partagée. Du côté de l'expérimentateur l'utilité d'utiliser des réseaux "en aveugle" pour démontrer la validité des outils employés est mise en cause. Du coté du théoricien les efforts se concentrent sur l'utilisation des réseaux neuronaux pour l'estimation fonctionnelle, et il devient alors difficile de justifier particulièrement cette approche face à d'autres approches (ondelettes, fonctions radiales de base, estimations paramétrique et non paramétrique).

De notre point de vue, l'utilisation des réseaux neuronaux en robotique possède au moins le grand mérite de remettre en question un certain nombre de "dogmes" concernant la planification, la modélisation et l'utilisation des capteurs en robotique "classique".

2.2.3 Limitations des méthodes adaptatives

Utiliser une méthode adaptative, basée seulement sur les données expérimentales, apporte l'énorme avantage de dispenser d'un modèle analytique. Mais bien sûr il y a un prix à payer pour cela: le caractère approximatif des réponses du modèle. La qualité de cette approximation dépend essentiellement de trois conditions: des données expérimentales utilisées (leurs nombre et leurs marges d'erreur), de l'effort dispensé au processus convergent d'adaptation de leurs paramètres et de la configuration de l'outil adaptatif.

Même si une méthode adaptative peut prouver formellement être capable de bien estimer une fonction lisse, dans la pratique il restera toujours une question fondamentale à laquelle il faut répondre: *combien d'exemples sont ils nécessaires pour obtenir un degré de précision requis* [Sto82] ? Il est bien connu que la réponse dépend de la dimension d et du degré de lissage (smoothness) p de la classe de fonctions à estimer [Sto82] [Sto85] [Lor86]. Ces études prouvent que le nombre d'exemples nécessaires pour bien estimer une fonction croit énormément avec la dimension de l'espace dans lequel elle est définie, bien que cet effet soit atténué par un haut degré de lissage de la fonction [PG89].

Pour une illustration quantitative prenons les considérations de Stone [Sto82]. Il a étudié une classe de problèmes d'estimation non-paramétrique et il a calculé le taux de convergence optimal ϵ_n, c'est à dire, une mesure de la précision qu'on peut espérer d'une approximation à partir d'un nombre donné d'exemples n. Il a montré qu'en utilisant la regression polynomiale on peut atteindre le taux de convergence optimal donné par l'équation 2.2.

$$\epsilon_n = n^{-\frac{p}{2*p+d}} \tag{2.2}$$

où p indique le nombre de variables du problème et d indique combien de fois la fonction est différentiable.

Pour une fonction deux fois différentiable avec deux variables, par exemple, il faut 8000 exemples pour obtenir $\epsilon_n = 0.05$, mais si cette fonction dépend de 10 variables le nombre d'exemples nécessaires pour obtenir le même taux de convergence augmente à 10^9. Par contre, si cette fonction de 10 variables était 10 fois différentiable 8000 exemples suffiraient pour obtenir $\epsilon_n = 0.05$.

Les résultats montrés ci-dessus sont optimaux, mais on ne doit pas espérer qu'ils seront atteints par les méthodes adaptatives présentées, même si une configuration optimale de la méthode (nombre et valeurs des paramètres) est choisie. Dans les cas pratiques ces quantités auront une tendance à s'élargir.

Cette caractéristique des méthodes adaptatives, d'avoir besoin d'un grand nombre d'exemples, surtout pour les problèmes à plusieurs variables, peut limiter leur utilité pratique. Pour beaucoup d'applications potentielles, le besoin d'obtenir une base d'exemples d'un ordre de grandeur élevé peut devenir un obstacle insurmontable. Le nombre d'exemples détermine aussi l'effort de calcul nécessaire pour trouver les bonnes valeurs des paramètres de la structure, et dans la pratique les essais d'"apprentissage", ou d'adaptation des paramètres, peuvent demander une grande puissance de calcul.

Le problème de la configuration de l'outil adaptatif est aussi très complexe. Déjà, il faut choisir la méthode la plus adaptée à un problème de modélisation particulier. Puis, dans la méthode choisie il y aura un certain nombre de choix à faire pour sa configuration avant de réaliser l'identification des valeurs des paramètres. Le comportement de la méthode dépend beaucoup de cette configuration. Le choix non averti de cette configuration peut rendre impossible une modélisation de bonne qualité. MacKay présente dans [Mac92] une étude approfondie de la comparaison et régularisation de modèles adaptatifs dans un cadre bayesien. Cependant, même s'il existe un support théorique pour le choix de la configuration des méthodes, dans la pratique on peut avoir beaucoup de difficulté à s'en servir. Cette difficulté limite l'intérêt initial d'utiliser les méthodes adaptatives comme outil rapide et simple de modélisation.

2.3 Une approche probabiliste de la modélisation

Dans ce paragraphe nous analyserons la modélisation dans le cadre des systèmes autonomes. Dans le paragraphe 2.3.1 nous proposons une réflexion à propos de la modélisation traditionnelle présentée jusqu'ici dans ce chapitre. Dans le paragraphe 2.3.2 nous présentons l'approche probabiliste de modélisation.

2.3.1 La modélisation des phénomènes complexes

Dans les deux paragraphes précédents nous avons étudié deux points de vue de la modélisation par l'identification de paramétres: la modélisation classique et les méthodes adaptatives. Ces deux points de vue divergent entre eux par la façon de déterminer la structure fonctionnelle du modèle. La modélisation classique est basée sur la caractérisation des structures analytiques mieux adaptées à un problème donné, alors que les méthodes adaptatives sont basées sur les outils généraux d'interpolation (éliminant ainsi le besoin d'une caractérisation).

Toutes les méthodes de modélisation présentées ont pourtant quelques points en commun:

- Ces méthodes supposent une complète dépendance du modèle au concepteur. Le rôle de ce concepteur varie beaucoup d'une méthode à l'autre, mais il doit au moins choisir les variables du système à mettre en relation dans le modèle ainsi que la structure du modèle.
- L'automatisme dans les méthodes se limite principalement à l'identification des paramètres de la structure (dans certaines méthodes il existe aussi une sélection automatique des données qui maximisent un critère de gain d'information [AO90], [PW93]).
- Les méthodes manipulent des données experimentales. Toute variation des valeurs contenues dans ces données, en dehors d'une relation fonctionnelle supposée par le concepteur, est vue comme du bruit. Aucune information n'est retenue de ces variations.
- Si le "bruit" contenu dans les données entraîne une erreur d'estimation (et une incertitude) des paramètres au-delà d'un seuil acceptable (iso-D minimum élevée) le modèle n'a aucune utilité.

Il est clair que la pratique de la modélisation basée sur les outils présentés dans ce chapitre est d'une grande utilité pour une grande classe de problèmes spécifiques. On note cependant le rôle préponderant du concepteur dans ces méthodes, ainsi que la difficulté à manipuler les incertitudes. Ceci peut dans certains cas rendre la modélisation impraticable.

Il faut chercher une nouvelle dimension pour la modélisation, où l'on envisage l'automatisation de toutes les étapes de la conception d'un modèle, et non plus seulement l'identification des paramètres. L'étude des systèmes autonomes fait partie de cette démarche.

Un système autonome doit être capable de construire lui même les représentations calculables de son interaction avec l'environnement, basé sur les connaissances préalables, l'expérience et l'observation de ces interactions. Il doit être capable d'établir des relations entre ses variables sensorimotrices, de trouver une structure pour représenter ces relations, de faire le calage objet-modèle, de vérifier la validité de son modèle, d'utiliser ses erreurs d'estimation comme information soit pour re-estimer ses paramètres, soit revoir la structure de son modèle, ou encore rétablir les relations entre ses variables. Il doit remplacer le modelisateur.

L'étude des systèmes autonomes est à présent seulement une direction de recherche à suivre. Plusieurs domaines scientifiques s'y intéressent (sciences cognitives, intelligence artificielle, robotique, etc.), mais il n'existe pas encore de méthodes générales ou d'outils disponibles.

Dans la suite de ce paragraphe on analysera une approche qui suggère une direction nouvelle dans l'étude de l'acquisition autonome de modèles.

2.3.2 Description de l'approche

Dans ce paragraphe nous allons présenter une approche de modélisation qui propose l'utilisation des probabilités comme outil pour acquérir, représenter et exploiter les dépendances entre variables d'un système autonome (voir [LBM94], [Ded95], [BDML94]).

La force de cette approche, par rapport à la modélisation classique, vient du fait d'utiliser un seul paradigme (les distributions de probabilités) pour représenter à la fois les connaissances préalables du concepteur, les connaissances acquises a posteriori, la structure, l'état du modèle et les incertitudes.

Un autre avantage de l'approche est sa structuration. Les connaissances sont décrites toutes de la même façon, mais contrairement aux méthodes “boîtes noires” adaptatives, on garde une vue structurée du modèle et de l'origine de toute forme paramétrique qui mette les variables du système en relation. Les changements du modèle par le concepteur peuvent donc se faire facilement et n'invalident pas les données acquises dans le passé.

Un modèle dans cette approche ne décrit pas les aspects d'une réalité physique, mais des *états de connaissance* du système. L'approche n'entretient jamais l'ambiguïté entre le modèle et la realité. Un état de connaissance est déterminé par toutes les informations disponibles et il évolue avec l'arrivée de nouvelles informations. Un état de connaissance prend en compte ses connaissances mais aussi son ignorance.

Les modèles classiques ne peuvent rien faire avec les variations des données en dehors des valeurs prévues par une structure préconçue. Un modèle basé sur les probabilités peut se servir des données bruitées ou des informations incomplètes, en associant aux valeurs calculées une probabilité.

Les deux propriétés de cette approche: l'existence d'un paradigme unique pour la manipulation des données et des connaissances de plus haut niveau et la structuration des connaissances, définissent un cadre favorable à l'automatisation de toutes les étapes de la modélisation.

On peut envisager une classe de processus d'auto analyse pour faire évoluer la structure même du modèle. Un processus d'auto analyse devra prendre en compte les erreurs d'évaluation du modèle, les incertitudes et les imprévus, et analyser la structure du modèle en manipulant les connaissances décrites par le concepteur de la même façon qu'il manipule les autres propositions.

Dans [Ded95] Dedieu propose une démarche pour la gestion de l'imprévu par les systèmes autonomes fondée sur cette approche probabiliste.

Représentation des connaissances

Cette approche est basée sur une vision probabiliste de la logique. La théorie des probabilités interprétée comme une extension de la logique a été proposée par Jaynes [Jay95].

En utilisant la logique, on peut partir d'un ensemble de propositions vraies (axiomes) et produire de nouvelles propositions vraies (théorèmes).

On peut aussi prouver qu'une proposition donnée est vraie ou fausse. Les "probabilités logiques" (PaL - de l'anglais "Probability as Logic") associent une *probabilité* d'être vraie ou fausse aux propositions. A la place de produire de nouveaux théorèmes par déduction, PaL calcule la probabilité d'une proposition d'être vraie étant donnée la probabilité d'autres propositions. Un "état de connaissance" d'un système d'inférence probabiliste est une distribution de probabilités sur l'ensemble des propositions du système.

Les probabilités des propositions sont combinées les unes avec les autres en suivant les règles suivantes:

$$P(AB|X) = P(A|X)P(B|AX) = P(B|X)P(A|BX) \tag{2.3}$$

$$P(A|X) + P(\overline{A}|X) = 1 \tag{2.4}$$

où A et B sont des propositions, X est la connaissance préalable, "AB" équivaut à "A et B", "$\overline{A}$" équivaut à "non A" et "$P(A|B)$" signifie "la probabilité de A être vraie étant donnée que B soit vraie".

Cox ([Cox79]) a montré que la logique est un cas particulier de la théorie PaL où les propositions ont soit la probabilité 1 ou 0 d'être vraies. PaL est donc une théorie plus puissante que la logique, une fois que ses calculs demeurent valables même quand les propositions n'ont pas 1 ou 0 comme valeurs, c'est-à-dire, même en utilisant des données incertaines, ou bruitées.

Un deuxième atout important de cette approche est la possibilité de réaliser le calcul des propositions même à partir d'un ensemble incomplet d'informations. Il est possible de prouver que la distribution qui respect le *maximum d'entropie* est la "meilleure" dans un sens mathématique très précis. Le maximum d'entropie permet de traiter l'ignorance, c'est-à-dire de déterminer un critère général de calcul qui comptabilise dans une distribution les points pour lesquels on dispose de données avec ceux qui ne sont pas encore connus. Dans la pratique la maximisation de l'entropie n'est possible que pour certains cas spécifiques.

Exploitation des connaissances

Pour illustrer cette façon de représenter la connaissance, supposons une proposition A qui concerne une action (par exemple "bouger un axe jusqu'à la position 10"), une proposition S qui concerne la lecture d'un capteur (par exemple "la distance du bout du manipulateur au mur") et une hypothèse H à propos de l'environnement dans lequel l'expérience se passe, on peut calculer:

- $P(S|AHX)$ la probabilité d'observer sur le capteur une valeur donnée sachant que l'action A a été prise dans le contexte H
- $P(A|SHX)$ la probabilité d'avoir pris (ou de devoir prendre) l'action A étant donné qu'on a observé (ou qu'on veut observer) S du capteur.
- $P(H|ASX)$ la probabilité d'être dans l'environnement H sachant que l'action A a été prise et que la valeur S a été observée.

Afin de pouvoir transformer les données expérimentales en distributions de probabilités manipulables par le calcul, on peut par exemple adopter l'hypothèse que la moyenne et l'écart type des valeurs obtenues sont des informations suffisantes pour décrire nos connaissances. Un état de connaissance va être décrit par un ensemble de gaussiennes. Pour le cas de notre illustration, pour une position donnée du manipulateur, les probabilités des distances possibles au mur seront la loi normale ayant la moyenne et l'écart type observés au cours des expériences:

$$P(S|AHDX) = \frac{1}{\sqrt{2\pi\sigma(A,H,D)^2}} e^{-\frac{(S-\mu(A,H,D))^2}{2\sigma(A,H,D)^2}} \tag{2.5}$$

où $\sigma(A,H,D)$ est l'écart type et $\mu(A,H,D)$ la moyenne sur les valeurs des actions A contenues dans l'ensemble D des données expérimentales prises dans l'environnement H.

Le problème cognitif direct

Dans le cadre de l'approche, et en utilisant toujours l'illustration du manipulateur, le problème cognitif direct consiste à prédire, à partir d'une consigne de position, les valeurs probables des variables sensorielles qui seront obtenues. Autrement dit: "quelles sensations vais je avoir si j'effectue tel mouvement ? "

Ce problème est résolu de manière triviale puisque la réponse est donnée par les distributions $P(S|AHDX)$ calculées directement d'après les données.

Le problème cognitif inverse (première version)

Le problème cognitif inverse consiste à prédire l'action permettant la perception d'une sensation désirée. On s'intéresse donc à la distribution $P(A|SHDX)$. Les distributions de probabilités ont un énorme avantage sur les autres modèles fonctionnels: elles sont *toujours* inversibles si on utilise le "théorème de Bayes" (dû en fait à Laplace). Pour notre cas d'illustration on aura:

$$P(A|SHDX) = P(A|HDX)\frac{P(S|AHDX)}{P(S|HDX)} \tag{2.6}$$

$P(A|HDX)$ nous est donné par la connaissance préalable sur l'expérimentation. On peut faire en sorte que toutes les N positions du manipulateur seront explorées suivant une loi uniforme:

$$P(A|HDX) = \frac{1}{N}$$

$P(S|AHDX)$ sont les gaussiennes obtenues d'après les données.

Enfin, $P(S|HDX)$ est obtenue en normalisant $P(A|SHDX)$:

$$\begin{aligned}\sum_A P(A|SHDX) &= 1 \\ P(S|HDX) &= \sum_A P(A|HDX)P(S|AHDX)\end{aligned}$$

Le problème cognitif inverse (deuxième version) ou la reconnaissance de situation

Le système a jusqu'ici une représentation interne, quantitative, non analytique et non symbolique, d'un certain environnement où se trouve le manipulateur. Cette représentation a été acquise par apprentissage comme le résultat d'interactions répétées avec cet environnement.

Maintenant, on suppose que notre système a appris par expérience un certain nombre de nouveaux environnements, avec des différentes configurations d'objets autour du manipulateur. Le manipulateur est placé dans l'un des environnements qu'il a déjà explorés (évidemment sans savoir lequel). Sa tâche consiste à sélectionner parmi les modèles disponibles celui qui correspond à l'environnement courant. Il peut pour cela faire des expériences c'est-à-dire aller à une certaine position et mesurer la distance à l'obstacle le plus proche. Nous sommes donc intéressés par $P(H|SADX)$ la probabilité de chacun des modèles:

$$\begin{aligned}P(H|SADX) &= P(H|DX)\frac{P(SA|HDX)}{P(SA|DX)} \\ &= P(H|DX)\frac{P(A|HDX)P(S|AHDX)}{P(SA|DX)P(S|DX)}\end{aligned}$$

L'absence de connaissance a-priori sur les probabilités respectives des modèles nous donne:

$$P(H|DX) = \frac{1}{M}$$

où M est le nombre d'environnements explorés.

Le dénominateur est obtenu par normalisation, et après quelques simplifications on obtient:

$$P(H|SADX) = \frac{P(S|AHDX)}{\sum_H P(S|AHDX)} \tag{2.7}$$

2.4 Synthèse

Dans ce chapitre, nous avons tracé un panorama du domaine de recherche de la modélisation. Nous avons vu les concepts d'état et de structure, les classifications des modèles et les étapes de la modélisation. Nous avons montré les grandes directions de la recherche:

- la modélisation "réaliste", qui automatise la manipulation des paramètres et des données, mais qui laisse au concepteur le rôle de gérant de la structure, et des circonstances en dehors du modèle (conditions d'expérimentation, erreurs et imprévus);
- l'idéal de la modélisation "autonome", comme nouvelle dimension de l'acquisition des modèles.

Dans la modélisation "réaliste" nous avons analysé :

- les méthodes "classiques", basées sur l'étape de caractérisation du modèle, où le concepteur élabore une structure adaptée au processus physique modélisé, pour faire ensuite l'identification des paramètres;
- les méthodes "adaptatives", basées sur les structures générales des interpolateurs. Ces "modèles de représentation pure" dispensent d'une caractérisation élaborée du modèle et dépendent surtout des données expérimentales.

Dans la modélisation "autonome" nous avons décrit une approche probabiliste de modélisation.

La phrase suivante, due à Richalet, définit bien le stade actuel du domaine:

"Modéliser est un acte prétentieux, ou inversement les modélisateurs expérimentés ont appris à être modestes. En tentant de plaquer une représentation mathématique pure et abstraite sur une réalité bruitée, non linéaire et non stationnaire, on peut espérer au mieux une similitude locale de comportement entre l'objet et le modèle, mais certainement pas un isomorphisme" [Ric91].

Chapitre 3

Une méthode d'identification structurelle

Ce chapitre a pour objectif de présenter la méthode qui fait l'objet de cette thèse, en voici le plan détaillé:

- Nous introduisons le chapitre par la description des objectifs de la méthode. Dans ce même paragraphe nous présentons le concept d'*identification structurelle*.
- Au paragraphe **3.2** nous présentons une illustration de la méthode pour matérialiser l'idée d'acquisition d'un modèle par identification structurelle.
- Au paragraphe **3.3** nous présentons la formulation mathématique de la méthode pour l'acquisition de *modèles directs*.
- Au paragraphe **3.4** nous présentons la méthode sous forme algorithmique.
- Au paragraphe **3.5** nous allons analyser le problème d'identification des paramètres des fonctions élémentaires qui composent nos modèles (les *fonctions de forme*).

3.1 Identification structurelle

Au chapitre 2 nous avons présenté les différentes façons d'aborder le problème d'acquisition des modèles. Nous y avons vu que les méthodes existantes d'acquisition automatique de modèles font en fait de l'identification paramétrique sur une forme structurelle définie par le concepteur. Cette forme peut-être un ensemble de fonctions mathématiques simples composées dans un outil d'interpolation générique (méthodes adaptatives) ou une équation de mesure bien adaptée à la description d'une certaine classe de processus physique (méthodes classiques).

Nous avons vu les avantages et aussi les inconvénients de chaque classe de méthodes. Fondamentalement nous avons distingué une forte dépendance des résultats des méthodes adaptatives aux données expérimentales, et à l'autre extrême, une forte dépendance des résultats des méthodes classiques à l'étape de caractérisation.

Dans ce mémoire nous voulons proposer une méthode d'acquisition de modèles qui va dans une nouvelle direction. Nous allons montrer que pour certaines formes générales d'équation il est possible d'inférer automatiquement l'équation de mesure qui s'adapte le mieux aux données expérimentales. Nous appelons cette inférence *l'identification structurelle*. L'identification structurelle va être faite en même temps que l'identification des paramètres, c'est-à-dire que la méthode estime les valeurs de certains paramètres en même temps qu'elle fait évoluer la structure du modèle.

Dans notre approche l'identification de paramètres est restreinte à un ensemble de sous-problèmes à une seule dimension d'entrée (les fonctions de forme). Pour résoudre ces sous-problèmes on peut utiliser une méthode d'identification de paramètres quelconque, adaptative ou classique.

L'avantage d'une méthode structurelle par rapport aux méthodes adaptatives est *la réduction de la complexité de l'identification de paramètres due à la réduction de dimension*. Il y a aussi une forte réduction du nombre de données nécessaires à l'acquisition du modèle.

L'avantage d'une méthode structurelle par rapport aux méthodes classiques est la *simplification de l'étape de caractérisation*. Le concepteur choisira une forme paramétrique (une courbe) adaptée à chaque sous-problème. L'équation de mesure qui combine toutes les variables sera choisie automatiquement.

La principale limitation de l'identification structurelle est sa forte dépendance à une forme d'équation particulière. Notre méthode propose une procédure qui fait l'identification structurelle automatique des fonctions pouvant s'écrire sous la forme:

$$f(x_0, ..., x_P) = \sum_{i=0}^{N} (\prod_{j=0}^{P} \varphi_{ij}(x_j)) \tag{3.1}$$

où les φ_{ij} sont des fonctions quelconques à une seule variable.

Au cours de ce mémoire les références génériques aux “formes polynômiales”, ou simplement aux “polynômes”, veulent exprimer cette forme d'équation.

En [BMB95] nous présentons cette même méthode d'acquisition de modèles comme un outil de "reconstruction d'hypersurfaces", dans un cadre d'approximation de fonctions. Dans ce mémoire nous proposons le concept d'identification structurelle pour situer plus clairement la méthode dans le domaine de recherche de l'acquisition de modèles.

Dans les paragraphes suivants nous allons présenter notre méthode. Nous introduisons la présentation par une illustration: le problème de la cinématique directe d'un bras planaire à deux dégrés de liberté 3.2. Ensuite nous passons à la formulation de la méthode.

3.2 Le problème de la cinématique directe

Dans ce paragraphe nous allons présenter un exemple d'acquisition de modèle pour un problème simple. Avec cet exemple nous voulons montrer comment il est possible d'acquérir un modèle complet d'un système à partir d'un ensemble réduit de données experimentales. Nous présentons un algorithme simplifié qui introduit notre méthode d'identification structurelle. La méthode sera présentée et justifiée formellement dans le paragraphe 3.3.

Considerons un problème simple: le calcul de la cinématique directe d'un bras planaire à deux degrés de liberté (figure 3.1).

La position du bras est définie par quatre variables: θ_1 et θ_2 (les angles des articulations), l_1 et l_2 (les longueurs des segments). Le problème de la cinématique directe consiste à calculer x et y (les coordonnées cartésiennes du bout du bras) à partir de la position angulaire du bras.

$$x = f_x(\theta_1, \theta_2) \tag{3.2}$$

$$y = f_y(\theta_1, \theta_2) \tag{3.3}$$

Pour illustrer notre méthode nous allons montrer que nous sommes capables de reconstituer automatiquement l'équation 3.2 qui permet de déduire x de θ_1 et θ_2. Notre reconstitution sera faite à partir d'un ensemble de fonctions dites de "forme". Pour simplifier l'illustration analytique on va considérer les longueurs

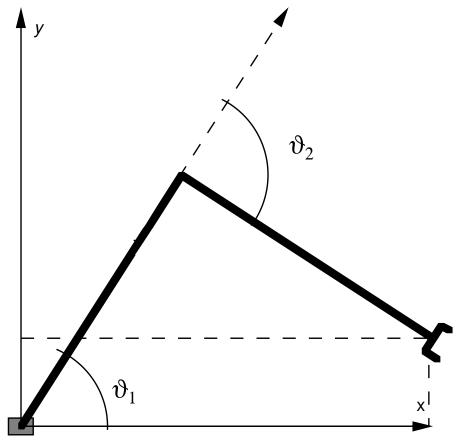

Figure 3.1 : Bras planaire à deux dégrés de liberté

des segments l_1 et l_2 égales à 1.

La fonction 3.2 utilisée dans notre illustration détermine une *hypersurface* dans un *hyperespace* d'entrée à deux dimensions. La figure 3.2 montre cette hypersurface.

La procédure de modélisation de la méthode commence par l'obtention d'un ensemble de fonctions élémentaires qu'on appelle les **fonctions de forme**. Une fonction de forme est mono-variable (une entrée). La variable d'entrée d'une fonction de forme appartient à l'ensemble des variables d'entrée du problème.

La figure 3.3 montre une fonction de forme de l'hypersurface du bras planaire pour l'axe θ_2 (trait continu).

Pour déterminer une fonction de forme, $f(\theta_1, 0)$ par exemple, les données seront obtenues (voir figure 3.4) en bougeant seulement une des articulations (articulation 1) pendant que l'autre articulation (articulation 2) reste fixe à une position angulaire donnée (0 radians).

Pour cette illustration nous supposerons que les représentations analytiques des fonctions de forme nous sont données par un "deus ex machina". Bien entendu l'idée de la méthode est d'obtenir une représentation de ces fonctions de forme par une des méthodes d'identification de paramètres expliquées au chapitre

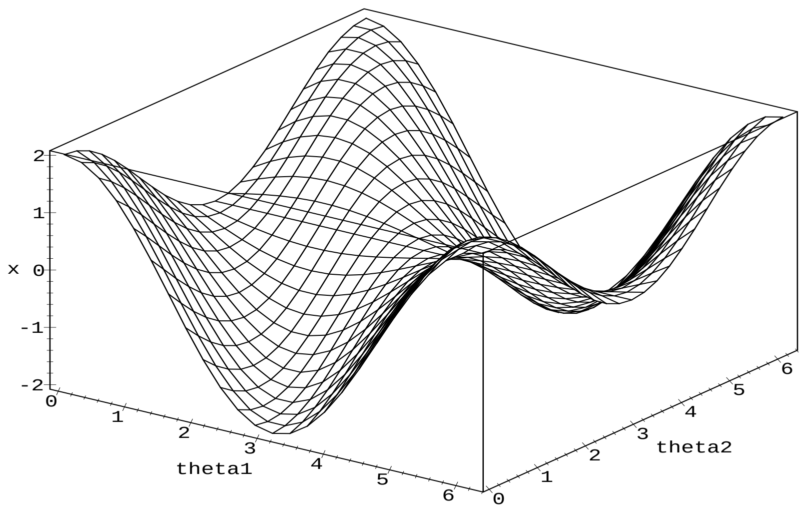

Figure 3.2 : Hypersurface déterminée par le bras planaire

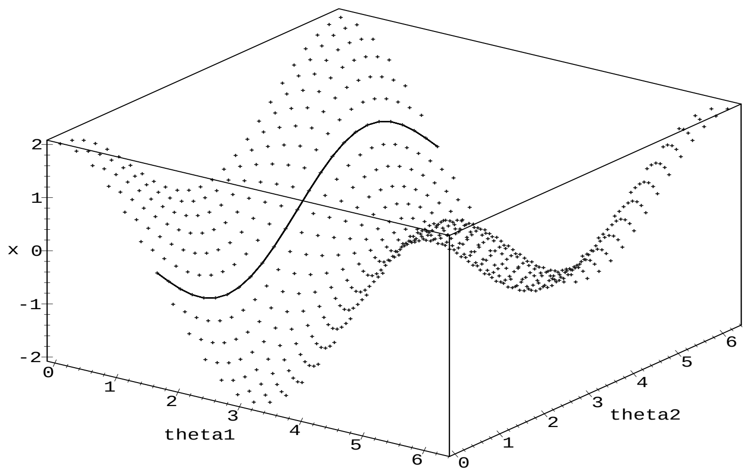

Figure 3.3 : Fonction de forme

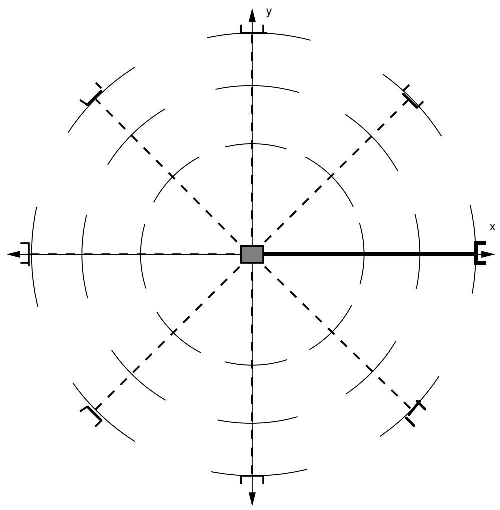

Figure 3.4 : Mouvement du premier axe du bras planaire

2, en ne perdant pas de vue que la dimension des entrées est réduite à 1.

Le mouvement de l'articulation 1 représenté dans la figure 3.4 détermine une fonction ayant la forme de la figure 3.5 et qui peut être décrite par l'équation 3.4.

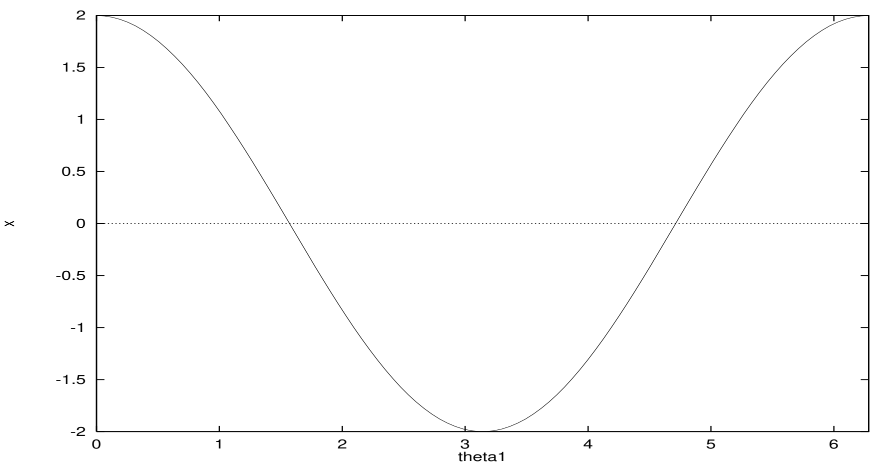

Figure 3.5 : Représentation du mouvement du premier axe du bras planaire

$$f_1^1(\theta_1) = f(\theta_1, 0) = 2.cos(\theta_1) \tag{3.4}$$

La méthode utilise les fonctions de forme dans un algorithme simple. Le premier pas de l'algorithme consiste à obtenir une fonction de forme par variable d'entrée. Pour notre illustration nous allons utiliser $f_1^1(\theta_1)$ et une deuxième fonction de forme obtenue cette fois en bougeant θ_2 et ayant θ_1 fixée. Pour notre ilustration θ_1 sera fixée à 0 radians (figure 3.6).

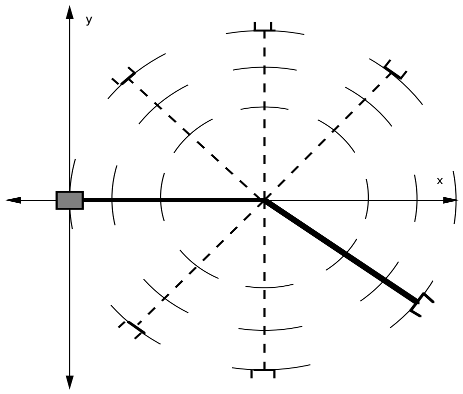

Figure 3.6 : Mouvement du deuxième axe du bras planaire

Dans ce cas on va obtenir une fonction ayant la forme de la figure 3.7 et qui peut être décrite par l'équation 3.5.

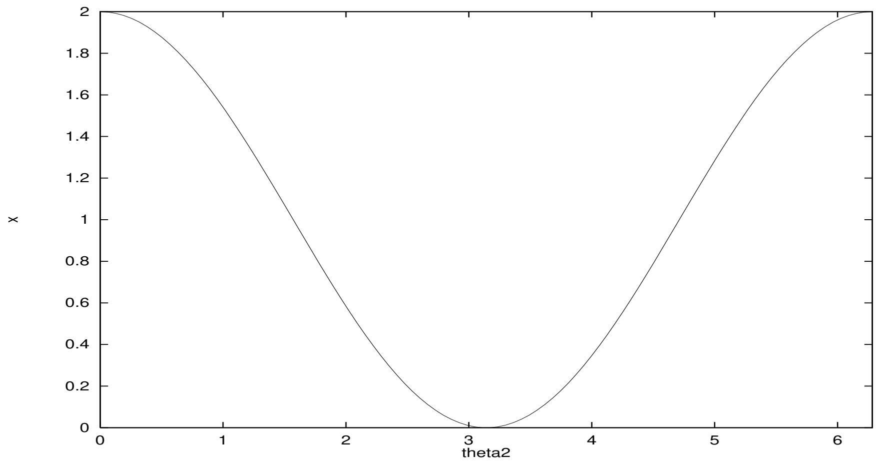

Figure 3.7 : Representation du mouvement du deuxième axe du bras planaire

$$f_2^1(\theta_2) = f(0, \theta_2) = 1 + cos(\theta_2) \tag{3.5}$$

Ensuite, le deuxième pas de l'algorithme fait le produit des deux fonctions de forme, $f_1^1(\theta_1)$ et $f_2^1(\theta_2)$. Ce produit doit être normalisé par la valeur de f au point de croisement des deux fonctions. Ce produit normalisé va définir une nouvelle fonction $f_1(\theta_1, \theta_2)$ (3.6).

$$f_1(\theta_1, \theta_2) = \frac{f_1(\theta_1).f_2(\theta_2)}{f(0,0)} \tag{3.6}$$

Le troisième pas de l'algorithme fait la comparaison entre la fonction f_1 obtenue et f. C'est le test d'arrêt de l'algorithme. Numériquement cette comparaison peut être faite par une série de tests sur des points non contenus par les fonctions de forme.

Dans notre illustration analytique f_1 aura la forme décrite par 3.7.

$$f_1(\theta_1, \theta_2) = cos(\theta_1) + cos(\theta_1).cos(\theta_2) \tag{3.7}$$

La fonction f_1 n'est pas encore l'équation recherchée, on continue à suivre l'algorithme. Dans la suite de l'algorithme (quatrième pas) on va utiliser deux nouvelles fonctions de forme de f obtenues par expérimentation. Il faut choisir deux autres positions fixes ($\theta_1 = \pi/2$ et $\theta_2 = \pi/2$ par exemple).

$$\begin{aligned} f_2^1(\theta_1) = f(\theta_1, \pi/2) &= cos(\theta_1) - sin(\theta_1) \\ f_2^2(\theta_2) = f(\pi/2, \theta_2) &= -sin(\theta_2) \end{aligned}$$

Et on va utiliser aussi deux fonctions de forme cette fois ci calculées à partir de f_1.

$$\begin{aligned} f_3^1(\theta_1) = f_1(\theta_1, \pi/2) &= cos(\theta_1) + cos(\theta_1).cos(\pi/2) = cos(\theta_1) \\ f_3^2(\theta_2) = f_1(\pi/2, \theta_2) &= cos(\pi/2) + cos(\pi/2).cos(\theta_2) = 0 \end{aligned} \tag{3.8}$$

Le cinquième pas consiste à calculer le produit de deux différences normalisé par une troisième différence. Chaque différence est faite entre une fonction de forme de f et une fonction de forme de f_1 pour la même variable (3.9).

$$\begin{aligned} f_2(\theta_1,\theta_2) &= \frac{(f_2^2(\theta_2) - f_3^2(\theta_2)).(f_2^1(\theta_1) - f_3^1(\theta_1))}{(f(\pi/2,\pi/2) - f_1(\pi/2,\pi/2))} \\ &= \frac{(-sin(\theta_2) - 0).(cos(\theta_1) - sin(\theta_1) - cos(\theta_1))}{-1-0} \\ &= -sin(\theta_1).sin(\theta_2) \end{aligned} \tag{3.9}$$

Le sixième pas de l'algorithme consiste à faire l'addition de l'équation utilisée comme entrée du troisième pas de l'algorithme (la fonction f_1) et l'équation obtenue à la sortie du sixième pas (la fonction f_2).

$$\begin{aligned} f_3(\theta_1,\theta_2) &= f_1(\theta_1,\theta_2) + f_2(\theta_1,\theta_2) \\ &= cos(\theta_1) + cos(\theta_1).cos(\theta_2) - sin(\theta_1).sin(\theta_2) \end{aligned}$$

À ce point on retourne au troisième pas de l'algorithme en utilisant f_3 à la place de f_1. Cette boucle a sa fin quand le test du trosième pas aura été satisfaisant. Dans notre illustration la fonction f_3 a déjà la forme recherchée pour l'équation 3.2, qui nous permet de déduire x de θ_1 et θ_2.

Nous avons trouvé à partir des 4 fonctions de forme $f(0,\theta_2)$, $f(\theta_1,0)$, $f(\pi/2,\theta_2)$ et $f(\theta_1,\pi/2)$ un modèle pour la cinématique directe du bras planaire.

Il faut remarquer que l'identification des paramètres se résume à l'obtention des fonctions de forme. Chaque fonction de forme étant une fonction d'une seule entrée cette identification est considérablement simplifiée.

Nous avons choisi les valeurs $\theta_1 = 0$, $\theta_2 = 0$, $\theta_1 = \pi/2$ et $\theta_2 = \pi/2$ pour simplifier les équations intermédiaires de l'illustration. La méthode **reste valable pour n'importe quel choix des valeurs utilisées comme référence des fonctions de forme**, pourvu que la valeur de f et de la différence $f - f_1$ utilisées pour la normalisation soient différentes de 0. Dans l'annexe A nous présentons cette même illustration (l'application de la méthode au bras planaire) sous forme symbolique dans le format Maple (logiciel mathématique). La présentation symbolique montre que le choix des valeurs de référence des fonctions de forme ne change pas le modèle obtenu.

3.3 Formulation de la méthode

Au paragraphe précédent nous avons présenté une illustration analytique de notre méthode d'identification structurelle. Dans ce paragraphe nous allons présenter la formulation complète de la méthode, et prouver qu'elle peut faire l'identification structurelle de toute fonction appartenant à l'ensemble des produits scalaires de fonctions quelconques d'une seule variable (équation 3.1).

Nous commençons par la formulation pour les cas à deux variables d'entrée.

Soit $\mathcal{F}$ l'ensemble de toutes les fonctions. Notre méthode s'intéresse à l'ensemble $\mathcal{F}_n$ décrit dans 3.10.

$$\begin{aligned} \mathcal{F}_n = \{ \quad f \in \mathcal{F} \text{ telle que } f(\theta_1, \theta_2) = & \quad \varphi_0(\theta_1).\psi_0(\theta_2) + \varphi_1(\theta_1).\psi_1(\theta_2) + ... \\ & + \varphi_n(\theta_1).\psi_n(\theta_2) \\ \text{où } \varphi_0, \psi_0, \varphi_1, \psi_1, ..., \varphi_n, \psi_n & \quad \text{sont des fonctions quelconques } \} \end{aligned} \tag{3.10}$$

La forme de $\mathcal{F}_n$ est assez générale et renferme plusieurs classes intéressantes de fonctions, comme par exemple les polynômes algébriques et les polynômes trigonométriques.

La base de la méthode d'identification structurelle pour les fonctions contenues dans l'ensemble $\mathcal{F}_n$ est la transformation fonctionnelle DPf proposée dans 3.11.

$$DP^{(\theta'_1, \theta'_2)} f(\theta_1, \theta_2) = f(\theta_1, \theta_2) - \frac{f(\theta_1, \theta'_2).f(\theta'_1, \theta_2)}{f(\theta'_1, \theta'_2)} \tag{3.11}$$

avec θ'_1 et θ'_2 tels que $f(\theta'_1, \theta'_2) \neq 0$.

Dans cette équation θ'_1 et θ'_2 représentent des valeurs constantes pour les variables d'entrée θ_1 et θ_2. $f(\theta'_1, \theta_2)$ et $f(\theta_1, \theta'_2)$ sont les fonctions de forme. $f(\theta'_1, \theta'_2)$ est la valeur de f au point de croisement de deux fonctions de forme.

Pour simplifier les notations nous allons utiliser $DP'f$ à la place de $DP^{(\theta'_1, \theta'_2)} f$ pour répresenter la transformation DP et ses constantes de normalisation.

Nous voulons montrer que la transformation DPf utilisée de façon récursive peut avec l'aide des fonctions de forme imiter exactement toute fonction appartenant à l'ensemble $\mathcal{F}_n$.

Pour montrer cela on va définir un ensemble d'ensembles Γ (3.12).

$$\left\{\begin{array}{l}\Gamma_0 = \{f \in \mathcal{F} \text{ et } \exists(\theta_1^0, \theta_2^0) \text{ tel que } \exists DP^0 f \text{ et } DP^0 f = 0\} \\ \Gamma_1 = \{f \in \mathcal{F} \text{ et } \exists(\theta_1^1, \theta_2^1) \text{ tel que } DP^1 f \text{ existe et appartient à } \Gamma_0\} \\ \Gamma_2 = \{f \in \mathcal{F} \text{ et } \exists(\theta_1^2, \theta_2^2) \text{ tel que } DP^2 f \text{ existe et appartient à } \Gamma_1\} \\ \vdots \\ \Gamma_n = \{f \in \mathcal{F} \text{ et } \exists(\theta_1^n, \theta_2^n) \text{ tel que } DP^n f \text{ existe et appartient à } \Gamma_{n-1}\}\end{array}\right\} \tag{3.12}$$

Le premier des ensembles de 3.12, Γ_0, contient toutes les fonctions de $\mathcal{F}$ pour lesquelles la transformation DPf est identiquement nulle. De façon récursive, Γ_1 contient toutes les fonctions de $\mathcal{F}$ pour lesquelles la transformation DPf définit une fonction qui à son tour appartient a l'ensemble antérieur Γ_0. Par conséquent $DP^1 DP^0$ des fonctions de Γ_1 est identiquement nulle.

Nous voulons prouver le théorème suivant:

Théorème: Pour toute fonction f et tout n, s'il existe une suite $(\theta_1^0, \theta_2^0), ..., (\theta_1^n, \theta_2^n))$ de couples distincts de R^2 telle que:

- $f(\theta_1^0, \theta_2^0) \neq 0$,
- $DP^0 f(\theta_1^1, \theta_2^1) \neq 0, \dots$,
- $DP^{n-1} DP^{n-2} ... DP^0 f(\theta_1^n, \theta_2^n) \neq 0$

alors:

$$f \in \Gamma_n \Leftrightarrow f \in \mathcal{F}_n$$

Nous devons prouver les deux sens de l'implication, c'est-à-dire, premièrement que toute fonction qui appartient à Γ_n et qui respecte les conditions de non nullité appartient aussi à $\mathcal{F}_n$ et deuxièmement que toute fonction qui appartient à $\mathcal{F}_n$ et qui respecte les conditions de non nullité appartient aussi à Γ_n. Pour cette preuve nous allons utiliser deux preuves par récurrence.

Il faut dire que les conditions de non nullité du théorème sont très peu contraignantes. Elles signifient simplement que la fonction $DP^i...DP^0$ n'est pas identiquement nulle.

3.3.1 Première preuve: $\Gamma_n \subset \mathcal{F}_n$

Pour la démonstration de la première partie prenons d'abord une fonction g appartenant à Γ_0 et démontrons que $g \in \mathcal{F}_0$. Par définition les fonctions de Γ_0 sont telles que DP^0 est identiquement nulle. Nous allons réécrire 3.11 pour cette fonction (3.13).

$$\begin{aligned} DP^0 g &= 0 \\ g(\theta_1, \theta_2) &= \frac{g(\theta_1, \theta_2^0).g(\theta_1^0, \theta_2)}{g(\theta_1^0, \theta_2^0)} \end{aligned} \tag{3.13}$$

Nous pouvons réécrire 3.13 en utilisant:

$$\begin{aligned} \varphi_0(\theta_1) &= \frac{g(\theta_1, \theta_2^0)}{g(\theta_1^0, \theta_2^0)} \\ \psi_0(\theta_2) &= g(\theta_1^0, \theta_2) \end{aligned}$$

et nous allons obtenir:

$$g(\theta_1, \theta_2) = \varphi_0(\theta_1).\psi_0(\theta_2) \tag{3.14}$$

Nous avons prouvé que g appartient à $\mathcal{F}_0$.

Supposons maintenant que:

$$\Gamma_{n-1} \subset \mathcal{F}_{n-1} \tag{3.15}$$

Pour la preuve par induction nous devons prendre une fonction k appartenant à Γ_n et réécrire 3.11 pour cette fonction (équation 3.16). La définition récursive de l'ensemble Γ nous dit que DP^0k est une fonction qu'appartient à Γ_{n-1}. Par hypothèse une fonction appartenant à Γ_{n-1} appartient aussi à $\mathcal{F}_{n-1}$.

$$\begin{aligned} DP^0k(\theta_1,\theta_2) &= k(\theta_1,\theta_2) - \frac{k(\theta_1,\theta_2^0).k(\theta_1^0,\theta_2)}{k(\theta_1^0,\theta_2^0)} \\ &= \varphi_0(\theta_1).\psi_0(\theta_2) + ... + \varphi_{n-1}(\theta_1).\psi_{n-1}(\theta_2) \end{aligned} \tag{3.16}$$

Nous pouvons réécrire 3.16 en utilisant:

$$\begin{aligned} \varphi_n(\theta_1) &= \frac{k(\theta_1,\theta_2^n)}{k(\theta_1^n,\theta_2^n)} \\ \psi_n(\theta_2) &= k(\theta_1^n,\theta_2) \end{aligned}$$

et nous allons obtenir:

$$k(\theta_1,\theta_2) = \varphi_0(\theta_1).\psi_0(\theta_2) + ... + \varphi_{n-1}(\theta_1).\psi_{n-1}(\theta_2) + \varphi_n(\theta_1).\psi_n(\theta_2) \tag{3.17}$$

CQFD.

3.3.2 Deuxième preuve: $\mathcal{F}_n \subset \Gamma_n$

Dans la deuxième partie de la preuve nous voulons prouver que toute fonction appartenant à l'ensemble $\mathcal{F}_n$ appartient aussi à Γ_n.

Nous commençons par montrer que les fonctions de $\mathcal{F}_0$ appartiennent aussi à Γ_0. Par définition les fonctions de $\mathcal{F}_0$ s'écrivent:

$$g(\theta_1,\theta_2) = \varphi_0(\theta_1).\psi_0(\theta_2) \in \mathcal{F}_0 \tag{3.18}$$

d'où:

$$DP^0 g(\theta_1,\theta_2) = \varphi_0(\theta_1).\psi_0(\theta_2) - \frac{\varphi_0(\theta_1^0).\psi_0(\theta_2).\varphi_0(\theta_1).\psi_0(\theta_2^0)}{\varphi_0(\theta_1^0).\psi_0(\theta_2^0)} \tag{3.19}$$

On voit bien que les termes en θ_1^0 et θ_2^0 de g s'annulent et que la différence résulte en 0. De cette façon on prouve que les fonctions de $\mathcal{F}_0$ appartiennent aussi à Γ_0.

Nous allons maintenant supposer comme hypothèse de récurrence que:

$$f \in \mathcal{F}_{n-1} \Rightarrow f \in \Gamma_{n-1} \tag{3.20}$$

Nous allons utiliser une fonction générale f de $\mathcal{F}_{n-1}$ (équation 3.21).

$$f(\theta_1,\theta_2) = \sum_{i=0}^{n-1} \varphi_i(\theta_1).\psi_i(\theta_2) \tag{3.21}$$

Nous allons utiliser encore deux autres fonctions :

$$g(\theta_1,\theta_2) = \varphi_n(\theta_1).\psi_n(\theta_2) \tag{3.22}$$

$$h(\theta_1,\theta_2) = f(\theta_1,\theta_2) + g(\theta_1,\theta_2) \tag{3.23}$$

h est une fonction de $\mathcal{F}_n$.

$$DP^0 h(\theta_1,\theta_2) = ((\sum_{i=0}^{n-1} \varphi_i(\theta_1).\psi_i(\theta_2)) + \varphi_n(\theta_1).\psi_n(\theta_2)) -$$
$$\frac{(\varphi_n(\theta_1).\psi_n(\theta_2^0) + \sum_{i=0}^{n-1} \varphi_i(\theta_1).\psi_i(\theta_2^0)).(\varphi_n(\theta_1^0).\psi_n(\theta_2) + \sum_{i=0}^{n-1} \varphi_i(\theta_1^0).\psi_i(\theta_2))}{((\sum_{i=0}^{n-1} \varphi_i(\theta_1^0).\psi_i(\theta_2^0)) + \varphi_n(\theta_1^0).\psi_n(\theta_2^0))} \tag{3.24}$$

Nous changeons maintenant les notations de 3.24 pour avoir une forme d'équation plus simple. Notons:

- $\alpha = \sum_{i=0}^{n-1} \varphi_i(\theta_1).\psi_i(\theta_2^0)$,
- $\alpha_n = \varphi_n(\theta_1).\psi_n(\theta_2^0)$,
- $\beta = \sum_{i=0}^{n-1} \varphi_i(\theta_1^0).\psi_i(\theta_2)$,
- $\beta_n = \varphi_n(\theta_1^0).\psi_n(\theta_2)$,
- $p = \sum_{i=0}^{n-1} \varphi_i(\theta_1^0).\psi_i(\theta_2^0)$,
- $p_n = \varphi_n(\theta_1^0).\psi_n(\theta_2^0)$

.

En utilisant notre nouvelle notation DP^0h se présente de la forme suivante:

$$DP^0h(\theta_1, \theta_2) = \sum_{i=0}^{n-1} \varphi_i(\theta_1).\psi_i(\theta_2) + \varphi_n(\theta_1).\psi_n(\theta_2) - \frac{(\alpha + \alpha_n)(\beta + \beta_n)}{(p + p_n)} \tag{3.25}$$

En développant 3.25 on obtient:

$$DP^0h(\theta_1, \theta_2) = \sum_{i=0}^{n-1} \varphi_i(\theta_1).\psi_i(\theta_2) + \varphi_n(\theta_1).\psi_n(\theta_2) - \frac{(\alpha.\beta + \alpha.\beta_n + \alpha_n.\beta + \alpha_n.\beta_n)}{p + p_n} \tag{3.26}$$

On sépare les termes:

$$DP^0h(\theta_1, \theta_2) = \sum_{i=0}^{n-1} \varphi_i(\theta_1).\psi_i(\theta_2) + \varphi_n(\theta_1).\psi_n(\theta_2) - (\frac{\alpha.\beta}{p + p_n} + \frac{\alpha.\beta_n + \alpha_n.\beta}{p + p_n} + \frac{\alpha_n.\beta_n}{p + p_n}) \tag{3.27}$$

Nous allons utiliser la transformation proposée dans l'équation 3.28 pour réécrire deux termes de l'équation 3.27:

$$\frac{a}{b + c} = \frac{a}{c} - \frac{a.b}{(b + c).c} \tag{3.28}$$

$$\frac{\alpha.\beta}{p_n + p} = \frac{\alpha.\beta}{p} - \frac{\alpha.\beta.p_n}{(p_n + p).p}$$
$$\frac{\alpha_n.\beta_n}{p + p_n} = \frac{\alpha_n.\beta_n}{p_n} - \frac{\alpha_n.\beta_n.p}{(p + p_n).p_n}$$

Après ces transformations on obtient:

$$\begin{aligned} DP^0 h(\theta_1, \theta_2) = & \quad \textstyle\sum_{i=0}^{n-1} \varphi_i(\theta_1).\psi_i(\theta_2)) + \varphi_n(\theta_1).\psi_n(\theta_2) \\ & -\tfrac{\alpha.\beta}{p} + \tfrac{\alpha.\beta.p_n}{(p_n+p).p} - \tfrac{\alpha.\beta_n+\alpha_n.\beta}{p+p_n} - \tfrac{\alpha_n.\beta_n}{p_n} + \tfrac{\alpha_n.\beta_n.p}{(p+p_n).p_n} \end{aligned} \tag{3.29}$$

Parmi les termes de 3.29 nous retrouvons $\frac{\alpha.\beta}{p}$. Si on réécrit ce terme avec la notation originale:

$$\frac{\alpha.\beta}{p} = \frac{(\sum_{i=0}^{n-1} \varphi_i(\theta_1).\psi_i(\theta_2^0)).(\sum_{i=0}^{n-1} \varphi_i(\theta_1^0).\psi_i(\theta_2))}{(\sum_{i=0}^{n-1} \varphi_i(\theta_1^0).\psi_i(\theta_2^0))}$$

Si on regroupe ce terme avec le premier terme de 3.29 on obtient:

$$\sum_{i=0}^{n-1} \varphi_i(\theta_1).\psi_i(\theta_2) - \frac{(\sum_{i=0}^{n-1} \varphi_i(\theta_1).\psi_i(\theta_2^0)).(\sum_{i=0}^{n-1} \varphi_i(\theta_1^0).\psi_i(\theta_2))}{(\sum_{i=0}^{n-1} \varphi_i(\theta_1^0).\psi_i(\theta_2^0))} = DP^0 f(\theta_1, \theta_2) \tag{3.30}$$

Le terme $\frac{\alpha_n.\beta_n}{p_n}$ nous intéresse aussi:

$$\frac{\alpha_n.\beta_n}{p_n} = \frac{\varphi_n(\theta_1).\psi_n(\theta_2^0).\varphi_n(\theta_1^0).\psi_n(\theta_2)}{\varphi_n(\theta_1^0).\psi_n(\theta_2^0)}$$

Si on regroupe ce terme et $\varphi_n(\theta_1).\psi_n(\theta_2)$ on obtient:

$$\varphi_n(\theta_1).\psi_n(\theta_2) - \frac{\varphi_n(\theta_1).\psi_n(\theta_2^0).\varphi_n(\theta_1^0).\psi_n(\theta_2)}{\varphi_n(\theta_1^0).\psi_n(\theta_2^0)} = DP^0 g(\theta_1, \theta_2) = 0$$

Après ces simplifications $DP^0 h$ se présente comme:

$$DP^0 h(\theta_1, \theta_2) = DP^0 f(\theta_1, \theta_2) + \frac{\alpha.\beta.p_n}{(p_n + p).p} - \frac{\alpha.\beta_n + \alpha_n.\beta}{p + p_n} + \frac{\alpha_n.\beta_n.p}{(p + p_n).p_n}) \tag{3.31}$$

Si on regroupe les trois derniers termes:

$$DP^0 h(\theta_1, \theta_2) = DP^0 f(\theta_1, \theta_2) + \frac{\alpha.\beta.p_n^2 + p^2.\alpha_n.\beta_n - p.p_n.\alpha.\beta_n - p.p_n.\beta.\alpha_n}{p^2.p_n + p.p_n^2} \tag{3.32}$$

On peut réécrire le nouveau terme comme un produit de deux termes divisés par un troisième terme constant:

$$DP^0 h(\theta_1, \theta_2) = DP^0 f(\theta_1, \theta_2) + \frac{(\alpha.p_n - p.\alpha_n).(\beta.p_n - p.\beta_n)}{p^2.p_n + p.p_n^2} \tag{3.33}$$

Nous allons réécrire 3.33 en utilisant deux nouvelles fonctions:

$$\begin{aligned} \sigma(\theta_1) &= \frac{\alpha.p_n - p.\alpha_n}{p^2.p_n + p.p_n^2} \\ &= \frac{(\sum_{i=0}^{n-1} \varphi_i(\theta_1).\psi_i(\theta_2^0)).\varphi_n(\theta_1^0).\psi_n(\theta_2^0) - (\sum_{i=0}^{n-1} \varphi_i(\theta_1^0).\psi_i(\theta_2^0)).\varphi_n(\theta_1).\psi_n(\theta_2^0)}{(\sum_{i=0}^{n-1} \varphi_i(\theta_1^0).\psi_i(\theta_2^0))^2.\varphi_n(\theta_1^0).\psi_n(\theta_2^0) + \sum_{i=0}^{n-1} \varphi_i(\theta_1^0).\psi_i(\theta_2^0).\varphi_n(\theta_1^0)^2.\psi_n(\theta_2^0)^2} \end{aligned}$$

$$\begin{aligned} \rho(\theta_2) &= \beta.p_n - p.\beta_n \\ &= (\sum_{i=0}^{n-1} \varphi_i(\theta_1^0).\psi_i(\theta_2)).\varphi_n(\theta_1^0).\psi_n(\theta_2^0) - (\sum_{i=0}^{n-1} \varphi_i(\theta_1^0).\psi_i(\theta_2^0)).\varphi_n(\theta_1^0).\psi_n(\theta_2) \end{aligned}$$

L'équation 3.33 devient donc:

$$DP^0 h(\theta_1, \theta_2) = DP^0 f(\theta_1, \theta_2) + \sigma(\theta_1).\rho(\theta_2) \tag{3.34}$$

Sachant que:

- d'après notre hypothèse de récurrence, la fonction f appartient à $\mathcal{F}_{n-1}$ et donc à Γ_{n-1}.
- par définition la transformation DP des fonctions de Γ_{n-1} définit une fonction qui à son tour appartient à Γ_{n-2}.
- nous avons prouvé auparavant que les fonctions de Γ_{n-2} appartiennent à $\mathcal{F}_{n-2}$ (paragraphe 3.3.1).

Nous pouvons donc conclure que le terme $DP^0 f(\theta_1, \theta_2)$ dans 3.34 est une fonction de $\mathcal{F}_{n-2}$.

Dans l'équation 3.34 nous avons en conséquence une fonction de $\mathcal{F}_{n-2}$ plus une fonction de $\mathcal{F}_0$, ce qui nous fait conclure que $DP^0 h(\theta_1, \theta_2)$ est une fonction de $\mathcal{F}_{n-1}$.

Depuis notre hypothèse de récurrence nous pouvons conclure aussi que si $DP^0 h(\theta_1, \theta_2) \in \mathcal{F}_{n-1}$ alors $DP^0 h(\theta_1, \theta_2) \in \Gamma_{n-1}$, et en conséquence que $h \in \Gamma_n$. Donc par induction nous pouvons conclure que effectivement $h \in \mathcal{F}_n \Rightarrow h \in \Gamma_n$.

CQFD.

3.3.3 Généralisation de la méthode

L'illustration du paragraphe 3.2 et la formulation presentée jusqu'ici dans ce paragraphe présentent la méthode pour l'identification structurelle des produits scalaires ayant deux variables d'entrée. Cependant, la méthode peut être généralisée pour les cas à un nombre quelconque de variables en rajoutant un deuxième niveau de récurrence.

Exemple: pour le cas des fonctions à trois variables d'entrée la transformation DP va avoir la forme suivante:

$$DP^0 f(x, y, z) = f(x, y, z) - \frac{f(x, y, z_0).f(x_0, y_0, z)}{f(x_0, y_0, z_0)} \tag{3.35}$$

Toute la formulation proposée dans ce paragraphe pour le cas à deux variables demeure valable dans le cas a trois variables, avec les ensembles récurrents de fonctions et l'application de la transformation DP à chaque niveau. La seule différence dans la transformation 3.35 est l'existence d'une fonction à deux variables d'entrée $f(x, y, z_0)$.

Cette fonction peut être elle même réduite de façon récursive en utilisant la méthode à deux variables d'entrée que nous venons de présenter.

Pour le cas général à n variables d'entrées, on aura alors l'application récurrente de la méthode pour le cas à $n-1$ variables, et ainsi de suite, jusqu'à la réduction complète de la fonction utilisant seulement des fonctions de forme.

Dans l'annexe B nous présentons, dans le format Maple, la méthode appliquée symboliquement au cas d'un polynôme à trois variables d'entrée et deux termes.

3.4 Algorithme de la méthode

Dans ce paragraphe nous allons présenter une version de notre algorithme récurrent. Pour l'utiliser nous devons nous placer sous les hypothèses suivantes:

- La fonction que nous cherchons à modéliser est de la forme:

$$f(x_0, ..., x_P) = \sum_{i=0}^{N} (\prod_{j=0}^{P} \varphi_{ij}(x_j)) \tag{3.36}$$

- Nous disposons d'une technique d'approximation fonctionnelle pour les fonctions à un paramètre et nous pouvons la mettre en oeuvre en ne faisant varier qu'une des variables d'entrée de notre système à la fois pour obtenir les fonctions de forme.
- Nous disposons d'un test d'arrêt nous permettant de décider si notre approximateur est bien celui de la fonction f.

Nous pouvons maintenant en décrire le principe: considérons une fonction $f(x, y)$.

Supposons que $f(x_0, y_0) \neq 0$ et que nous ayons obtenu expérimentalement deux approximateurs pour les fonctions de forme $f(x_0, y)$ et $f(x, y_0)$.

En posant :

- $r_0 = f(x_0, y).f(x, y_0)/f(x_0, y_0)$

- $DP^0 f(x,y) = f(x,y) - r_0(x,y)$
- on remarque que l'on sait calculer $r_0(x,y)$ en tout point.

S'il existe un couple x_1, y_1 tel que $f(x_1,y_1) - r_0(x_1,y_1) \neq 0$ alors on peut appliquer encore une fois DP:

$$DP^1 DP^0 f(x,y) = f(x,y) - r_0(x,y) - \frac{((f(x,y_1) - r_0(x,y_1)).(f(x_1,y) - r_0(x_1,y))}{(f(x_1,y_1) - r_0(x_1,y_1)))}$$

ou bien encore:

$$r'_0(x,y) = \frac{((f(x,y_1) - r_0(x,y_1)).(f(x_1,y) - r_0(x_1,y))}{(f(x_1,y_1) - r_0(x_1,y_1)))} \tag{3.37}$$

$$DP^1 DP^0 f(x,y) = f(x,y) - r_0(x,y) - r'_0(x,y)$$

On remarque que l'on peut calculer r'_0 en tout point dès lorsque l'on a obtenu expérimentalement des approximateurs des fonctions de forme $f(x,y_1)$ et $f(x_1,y)$. On peut alors calculer $r_1(x,y) = r_0(x,y) + r'_0(x,y)$ en tout point.

Ce processus nous permet de calculer la fonction r'_{n-1} en tout point. En effet si $f(x,y) - r_{n-1}(x,y)$ est égale à zéro en tout point alors $f(x,y) = r_{n-1}(x,y)$ et notre algorithme est terminé. Sinon, on peut choisir un couple (x_n, y_n) et obtenir expérimentalement les approximateurs des fonctions de forme $f(x,y_n)$ et $f(x_n,y)$ puis appliquer la formule:

$$r'_{n-1} = \frac{(f(x,y_n) - r_{n-1}(x,y_n)).(f(x_n,y) - r_{n-1}(x_n,y))}{f(x_n,y_n) - r_{n-1}(x_n,y_n)}$$

on obtient une nouvelle fonction r_n:

$$r_n(x,y) = r_{n-1}(x,y) + r'_{n-1}(x,y) \tag{3.38}$$

Notre première hypothèse à propos de la forme des fonctions (équation 3.36), et le théorème démontré aux paragraphes 3.3.1 et 3.3.2 assurent la convergence de l'algorithme.

La programmation reste cependant délicate car chaque pas de l'algorithme ne produit pas des valeurs mais une fonction dont l'évaluation sera nécéssaire à l'étape suivante. Nous tentons maintenant de donner une version en "pseudo lisp" de cet algorithme en précisant que si l'on est hermétique a Lisp il faut retenir que notre algorthime produit une fonction plutôt que des valeurs.

Nous appelons programme la forme:

```
`(lambda (x y) ....)
```

Nous rappelons que Lisp nous permet d'appliquer dynamiquement un programme à des paramètres. Ansi si r_n est un programme Lisp à deux paramètres alors

```
(rn x y)
```

fait l'appel de r_n avec les paramètres x et y. Il faut garder en mémoire que le signe "`" bloque l'évaluation alors que le signe "," la force.

Nous allons construire un algorithme récursif ayant en entrée un programme r_n et en sortie le même programme si $f = r_n$ ou un programme r_p si $f = r_p$.

```
(defun structure (s)
  (let ((pivot (experimental_non_null_f_minus_s(s))))
; permet de déterminer expérimentalement un couple de valeur
;pour lesquel f-s n'est pas nulle.
    (cond ((not pivot) s)

;; il n'existe pas de valeur pour lesquelles  $f-s$ n'est pas nulle
;; donc f=s. (s x y) permet d'évaluer s en tout point: nous avons
;; notre estimateur il va naturellement remonter à la surface des
;; appels ! sinon la suite:
         ( t (let* ((xn  (car pivot))
                    (yn  (cadr pivot))
; xn et yn ont été choisi tel que
         f(xn,yn) - ( s xn yn) <> 0
                    (fnxy (experimental_eval))
; prise de la valeur experimental de f(xn,yn)
                    (fnx  (experimental_form_function_x yn))
; définition d'une fonction de forme de x avec y=yn fixé dans le
; process
                    (fny  (experimental_form_function_y xn))
; définition d'une fonction de forme de y avec x=xn fixé dans le
; process
; ------------------------------------------------------------
; déclaration du programme générique:
                    (s1 `(lambda (x y)
                           (+ ; rn + rn'
                            (,s x y) ;  rn
                            (/ (* (- (,fnx x) (,s x ,yn)) ;rn'
                                  (- (,fny y) (,s ,xn y)))
                               ,fnxy)))))
               (structure s1)))))); appel recursif;
; ------------------------------------------------------------
```

Le premier appel se faisant avec le programme s_0:

```
(lambda (x y) (/ (* (,f0x x)(,f0y y)) ,f0xy)))
```

En supposant que $DP^1DP0f = 0$ le programme obtenu à la forme:

```
(lambda (x y) (+ (/ (* (-
                         (f1x x)
                         ((lambda (x y)
                            (/ (* (f0x x) (f0y y)) f0xy))
                          x y1))
                        (-
                         (f1y y)
                         ((lambda (x y)(/ (* (f0x x) (f0y y)) f0xy))
                          x1 y))
                     (- f1xy
                        ((lambda (x y) (/ (* (f0x x) (f0y y)) f0xy))
                         x1 y1))))
                  ((lambda (x y)(/ (* (f0x x) (f0y y)) f0xy))
                   x y)))
```

Ce programme peut être représenté sous la forme d'un arbre (voir figure 3.8) où I1, I2, I3 et I4 sont les fonctions de forme f0x, f0y, f1x et f1y.

Généralisation de la méthode pour les fonctions à p paramètres

Dans le paragraphe 3.3.3 nous avons donné la définition de la transformation *DP* pour une fonction à trois paramètres (équation 3.35).

L'algorithme pour ces fonctions se déduit du précédent de la façon suivante. Le terme r_0 dépend maintenant d'une fonction à deux paramètres $f(x, y, z_0)$ et d'une fonction de forme $f(x_0, y_0, z)$. Il suffit alors de déterminer expérimentalement la fonction de forme et d'appliquer l'algorithme précédent pour la fonction à deux variables pour obtenir un modèle de cette fonction et pouvoir calculer r_0 en tout point.

Si l'on applique de nouveau la transformation *DP* on obtient:

$$DP^1DP^0f(x, y, z) = f(x, y, z) - r_0 - r'_0$$

avec:

$$r'_0 = \frac{(f(x, y, z_1) - r_0(x, y, z_1)).(f(x_1, y_1, z) - r_0(x_1, y_1, z))}{f(x_1, y_1, z_1) - r_0(x_1, y_1, z_1)}$$

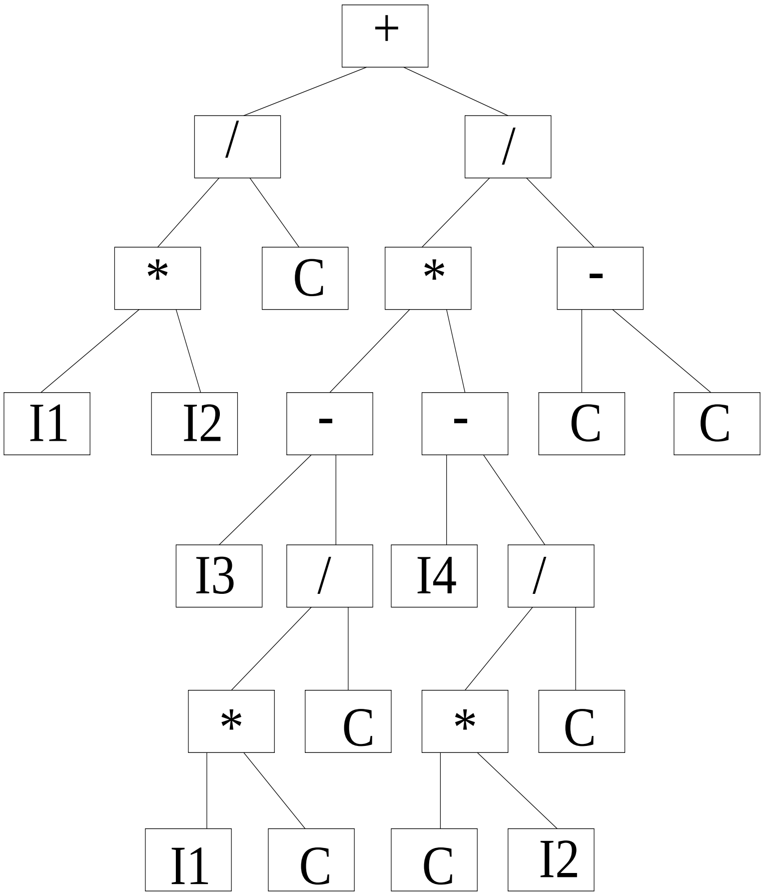

Figure 3.8 : Structure du modèle du bras planaire

On voit bien que r'_0 peut se calculer par le même procédé: en construisant expérimentalement une fonction de forme $f(x_1, y_1, z)$ et en utilisant l'algorithme précédent pour le modèle de $f(x, y, z_1)$. On peut alors définir $r1 = r_0 + r'_0$.

De cette facon on peut construire $r_n = r_{n-1} + r'_{n-1}$ et terminer l'algorithme quand on vérifie expérimentalement que $f = r_n$.

De manière plus génerale si f à p paramètres alors r'_{n-1} peut s'obtenir à partir d'une fonction de forme de f et d'un modèle d'une fonction à $p-1$ paramètres.

On mesure qu'en utilisant cet algorithme sans précaution particulière on va faire croître de façon exponentielle le nombre de fonctions de forme a acquérir. Comme nous allons le montrer dans le paragraphe 4.2 il est cependant possible d'obtenir des modèles partiels de f sans avoir recour à de nouvelles expériences.

Encore une dernière remarque: il est important de noter que les termes $r'_i(x, y)$ (équations 3.37 et 3.38) qui composent la transformation DPf aux différents niveaux de récurrence ne sont pas équivalents aux termes $\prod_{j=0}^{P} \varphi_{ij}(x_j)$ de la forme analytique supposée de f (équation 3.1). La fonction finale determininée par DPf est équivalente à f mais ses termes intermédiaires n'ont rien à voir avec les termes de f.

3.5 Les fonctions de forme

Dans ce paragraphe nous allons analyser le problème de la représentation des fonctions de forme.

Un des avantages qu'apporte la méthode d'identification structurelle par rapport à l'acquisition de modèles par identification de paramétres simple est justement la réduction de la dimension du problème d'identification des paramètres. Nous allons manipuler des fonctions d'une seule variable au lieu de fonctions à plusieurs dimensions. On évitera ainsi l'effet exponentiel d'augmentation de la complexité du problème d'identification de paramétres que la croissance du nombre de dimensions du problème entraîne.

Nous retrouvons ici la même problématique mise en évidence au chapitre 2. Pour représenter une fonction de forme on a le choix entre construire un modèle de représentation en utilisant une technique générique d'approximation fonctionnelle, ou construire un modèle plus proche d'un modèle de connaissance déterminé par une étape de caractérisation.

3.5.1 Les modèles de représentation des fonctions de forme

Pour construire un modèle de représentation d'une fonction de forme nous avons en gros le choix entre utiliser un outil interpolateur du type "boîte noire" ou estimer la fonction de forme par une fonction interpolatrice explicite.

Si on veut choisir un outil interpolateur il existe bien des options: les réseaux de neurones, l'interpolation linéaire, les "memory-based models", ou encore les méthodes d'ajustement (*fitting*) de courbes par régression locale. Les méthodes adaptatives d'interpolation qui ont été analysées dans 2.2 sont toutes applicables dans ce cas.

Il est important de remarquer que faire de l'interpolation de points sur un ensemble de sous-problèmes à une seule dimension au lieu d'avoir un seul grand problème d'interpolation à plusieurs dimensions entraîne une forte réduction du nombre de points nécessaires pour obtenir une précision donnée, et permet aussi la parallélisation de l'identification de paramètres.

Nous n'allons pas analyser en détail l'utilisation des outils interpolateurs pour la représentation des fonctions de forme. Le choix de les utiliser effectivement existe, mais il nous semble que l'utilisation des fonctions interpolatrices explicites, qu'on analysera ensuite, est plus commode, plus économique, et en conséquence plus adaptée aux problèmes d'interpolation à une dimension d'entrée.

Les polynômes algébriques et les polynômes trigonométriques constituent deux classes très importantes de fonctions d'approximation, qui sous certaines conditions connues, ont prouvé être le moyen naturel d'estimation d'autres fonctions plus au moins arbitraires [AKL86].

A la fin du dernier siècle, le mathématicien allemand Weirstrass a prouvé la possibilité théorique d'estimer une fonction continue arbitraire par un polynôme algébrique (3.39) avec un niveau donné d'exactitude [AKL86], ce qui fait des polynômes algébriques des interpolateurs universels. Les polynômes trigonométriques (3.40) constituent une deuxième classe très importante de fonctions d'approximation.

$$P(x) = a_0 + a_1 x + ... + a_n x^n \tag{3.39}$$

$$u_n(x) = \alpha_0 + \sum_{k=1}^{n} (\alpha_k cos(kx) + \beta_k sin(kx)) \tag{3.40}$$

Dans l'annexe D nous analysons ces deux classes de polynômes interpolateurs, et nous montrons comment est faite l'identification des paramètres de ces polynômes.

3.5.2 Caractérisation des fonctions de forme

Jusqu'à ce point nous avons traité le problème de la représentation des fonctions de forme en supposant que le concepteur du modèle ne dispose pas d'informations préalables à propos de ces fonctions, et qu'une méthode générale d'approximation (ou d'interpolation) va être utilisée pour construire les représentations des fonctions. C'est-à-dire qu'il n'y aura pas une phase de caractérisation des fonctions de forme, seulement un choix entre utiliser un outil d'approximation ou utiliser une fonction interpolatrice, et pour chaque cas un choix d'outil ou de classe de fonction à utiliser.

Dans certains cas le concepteur peut être capable de caractériser son modèle, et de choisir la (ou les) forme(s) fonctionnelle(s) idéale(s) à la répresentation des fonctions de forme. La caractérisation est idéale dans ces cas, et une identification de paramètres idéale (sans influence de bruit) nous amènerait à un modèle parfait du système.

Un cas illustrant bien cette caractérisation idéale dans la robotique est le cas des bras manipulateurs. Généralement l'état d'un manipulateur peut être déterminé par l'ensemble des valeurs de ses articulations. Pour une articulation qui réalise des mouvements de rotations, par exemple, les fonctions de forme vont mettre en relation ses valeurs angulaires et les coordonnées cartésiennes. On sait bien que la forme fonctionnelle la plus adéquate de représentation pour ces fonctions de forme seront les sinusoïdes.

Une sinusoïde quelconque peut être représentée par le polynôme trigonométrique élémentaire de l'équation 3.41, qui est un cas particulier de la forme générale D.6 (annexe D) ayant $n = 1$.

$$f(x) = a_1 + a_2 cosx + a_3 sinx \tag{3.41}$$

On peut facilement identifier les paramètres a_1, a_2, a_3 en utilisant la méthode classique présentée au paragraphe 3.5.1. C'est-à-dire que pour les cas des manipulateurs, les fonctions de forme n'ont besoin que de trois points pour être déterminées.

A l'exemple des manipulateurs, s'il est possible de caractériser idéalement la fonction de forme, l'exactitude du modèle construit par la méthode va dépendre seulement de l'exactitude de l'identification paramétrique, qui à son tour dépend des points d'interpolation.

Chapitre 4

Utilisation et transformation des modèles

Les modèles acquis par la méthode d'identification structurelle ont une architecture ouverte et simple. Cela permet d'envisager plusieurs types d'opération sur la structure des modèles visant à mieux les exploiter et même les transformer a-posteriori s'il le faut.

Dans ce chapitre nous proposons une série d'opérations de base sur la structure des modèles. En voici son plan.

- Les modèles directs acquis par la méthode ont une structure qui nous permet le calcul des dérivées partielles du système. Au paragraphe **4.1** nous présentons les *modèles différentiels directs* pour le calcul de ces dérivées. Nous y présentons aussi une approche d'*inversion* des modèles. Cette inversion va nous permettre le contrôle du système.

- Au paragraphe **4.2** nous proposons une variation de la méthode récursive d'identification structurelle pour faire des transformations de modèles. Cette méthode va permettre l'acquisition de nouveaux modèles et sous modèles à partir de modèles et sous modèles existants.

- Dans des conditions réalistes d'expérimentation les modèles acquis par la méthode auront un écart de structure par rapport au système, c'est-à-dire, ses sorties vont toujours contenir une certaine erreur. Au paragraphe **4.3** nous proposons une approche d'estimation des marges d'erreur du modèle dues au bruit contenu dans les données expérimentales.

- L'erreur du modèle direct peut être importante quand il y a une erreur d'estimation des paramètres. Cette erreur est une conséquence de l'utilisation des données bruitées. Au paragraphe **4.4** nous proposons une technique d'adaptation du modèle pour réduire l'erreur d'estimation des paramètres.

Les paramètres vont être corrigés et l'effet du bruit contenu dans le données va être minimisé.

- **4.5**. Dans ce dernier paragraphe nous allons faire une analyse quantitative des données nécessaires à l'acquisition des modèles par notre méthode.

4.1 Calcul des dérivées partielles et inversion du modèle

Dans les termes de la Théorie des Systèmes, les modèles acquis par notre méthode d'identification structurelle sont dits des *modèles directs*, car ils peuvent estimer les valeurs des sorties du système modélisé à partir des valeurs de ses entrées. Dans ce paragraphe nous allons montrer comment obtenir un *modèle différentiel direct* du système à partir de son modèle direct acquis par la méthode. Le modèle différentiel direct fournit la matrice jacobienne des dérivées partielles du système, avec laquelle on pourra inverser le modèle afin de faire son asservissement.

La méthode va déterminer une structure complète pour le modèle. Cette structure sera composée par un ensemble de fonctions de forme, un ensemble d'opérations algébriques et un ensemble de paramètres identifiés.

Dans la suite nous allons considérer la représentation arborescente du modèle engendré par notre algorithme (figure 3.8).

Nous rappelons que les symboles $+-*/$ représentent les opérations algébriques, les C représentent les constantes de normalisation de la transformation DP, les In représentent les fonctions de formes. Les paramètres identifiés de la structure sont les constantes de normalisation et les paramètres des équations choisies pour modéliser les fonctions de formes.

Les fonctions de forme à la base de l'arbre et l'enchaînement d'opérations algébriques nous permettent de calculer les dérivées partielles des sorties par rapport à chaque entrée. Pour le calcul des dérivées partielles du premier ordre on utilise les règles de différentiation classiques que nous allons rappeler.

- Pour une fonction de forme dépendant d'une variable q, sa dérivée peut être obtenue formellement par la différentiation de sa forme paramétrique, ou alors la dérivée peut être estimée approximativement par : $\frac{\partial f(q)}{\partial q} = \frac{\Delta f(q)}{\Delta q}$ en utilisant deux valeurs calculées de la fonction de forme. À l'exception de la variable q, les autres variables du système gardent des valeurs constantes pendant l'acquisition de la fonction de forme. En conséquence les dérivées partielles de cette fonction de forme par rapport à ces autres variables sont nulles.

- Pour les opérations algébriques, le calcul des dérivées de la sortie de l'opération par rapport à chaque variable d'entrée est fait en utilisant les règles de différentiation rappelées en 4.1. Pour ce calcul on utilise les valeurs des entrées de l'opération et des dérivées partielles déjà calculées pour ces entrées.

$$
\begin{aligned}
\frac{\partial(u+v)}{\partial q} &= \frac{\partial u}{\partial q} + \frac{\partial v}{\partial q} \\
\frac{\partial(u-v)}{\partial q} &= \frac{\partial u}{\partial q} - \frac{\partial v}{\partial q} \\
\frac{\partial(u*v)}{\partial q} &= u*\frac{\partial v}{\partial q} + v*\frac{\partial u}{\partial q} \\
\frac{\partial(\frac{u}{v})}{\partial q} &= \frac{v*\frac{\partial u}{\partial q} - u*\frac{\partial v}{\partial q}}{v^2}
\end{aligned}
\tag{4.1}
$$

La différentiation au premier ordre du modèle direct $x = f(q)$ pour toutes les x_i sorties par rapport à toutes les q_j entrées nous donne la *matrice jacobienne*:

$$J_{ij} = \frac{\partial f_i(q)}{\partial q_j} \; (i = 1, ..., m; \; j = 1, ..., n)$$

où: J_{ij} définit l'élément de la i-nième ligne et j-ième colonne de la matrice, m étant le nombre de variables de sortie du système et n le nombre de variables d'entrée du système.

Pour notre illustration nous pouvons écrire:

$$f(q) = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \tag{4.2}$$

Après la différentiation nous allons avoir:

$$\begin{bmatrix} dx \\ dy \end{bmatrix} = \mathbf{J} \begin{bmatrix} d\theta_1 \\ d\theta_2 \end{bmatrix} = \begin{bmatrix} \frac{\partial f_x}{\partial \theta_1} & \frac{\partial f_x}{\partial \theta_2} \\ \frac{\partial f_y}{\partial \theta_1} & \frac{\partial f_y}{\partial \theta_2} \end{bmatrix} \begin{bmatrix} d\theta_1 \\ d\theta_2 \end{bmatrix}$$

On obtient alors un ensemble d'équations algébriques linéaires:

$$\begin{cases} dx = & \sum_j \frac{\partial f_x}{\partial \theta_j} * d\theta_j \\ dy = & \sum_j \frac{\partial f_y}{\partial \theta_j} * d\theta_j \end{cases} \tag{4.3}$$

L'intérêt principal d'obtenir la matrice jacobienne du système modélisé est de pouvoir calculer l'**inversion** du modèle. Un modèle inverse est un modèle interne qui produit une action en fonction d'un état du système et d'une sensation désirée [JR91]. Il s'agit, effectivement, d'inverser l'ordre causal mis en jeu par le système physique. Dans le cas de notre illustration, le bras planaire, si le bout du bras exécute un mouvement, l'origine est bien l'ensemble des commandes motrices appliquées au bras. Un roboticien doit alors concevoir un module, un algorithme, ou d'une manière plus générale une *représentation fonctionnelle* permettant le passage automatique des effets voulus aux causes, de la trajectoire du bout du bras aux commandes motrices.

Les modèles inverses sont très utiles dans plusieurs domaines. Notre intérêt est particulièrement centré sur l'utilisation des modèles inverses pour l'asservissement des systèmes, où un contrôleur reçoit une sensation désirée comme entrée et doit trouver une action qui puisse engendrer une sensation aussi proche que possible de la sensation désirée; c'est-à-dire, le contrôleur doit inverser la transformation d'actions en sensations (le modèle direct).

Une façon de faire l'asservissement est d'utiliser un modèle inverse explicite comme contrôleur. Cependant, tandis que les modèles directs sont déterminés de façon unique, les modèles inverses généralement ne le sont pas. Si le système est caractérisé par un "mapping many-to-one" des actions aux sensations, alors il existe généralement un nombre infini de modèles inverses possibles. En Robotique ces systèmes sont définis comme ayant des *degrés de liberté en excès.*

Il faut aussi remarquer que l'inversion n'existe pas toujours - il n'est pas toujours possible d'atteindre n'importe quelle sensation désirée en partant de n'importe quel état.

Le problème de l'inversion des modèles est assez complexe. Une solution analytique, qui nous donne un modèle inverse explicite, est plus désirable, car l'asservissement doit être fait généralement en temps réel et le calcul à partir d'un modèle explicite est normalement plus rapide. Dans la pratique les concepteurs cherchent à établir des heuristiques ou des algorithmes d'inversion pour trouver un modèle explicite. Chacune de ces techniques n'est valable que pour certaines catégories de système [SV89].

Une solution alternative pour l'inversion des modèles est l'utilisation des procédés itératifs, qui sont basés sur un modèle inverse différentiel du système et qui réduisent progressivement l'écart résiduel dû à l'erreur du modèle inverse de façon itérative. La matrice jacobienne est largement utilisée par les méthodes itératives.

La formulation différentielle de premier ordre du problème, qu'on voit dans le système d'équations 4.3, permet l'utilisation des techniques mathématiques pour la résolution des systèmes d'équations algébriques linéaires.

Dans le système d'équations 4.3 on veut calculer $d\theta_j$ étant donnés dx et dy. La jacobienne $\mathbf{J}$ est la matrice des coefficients du système d'équations linéaires, et la solution de ce système pour le vecteur d'inconnues θ_j équivaut au calcul de l'inverse de la matrice jacobienne $\mathbf{J^{-1}}$ au point où se trouve le système [Pre92]. L'inversion du notre modèle peut être donc représentée par l'équation 4.4.

$$\begin{bmatrix} d\theta_1 \\ d\theta_2 \end{bmatrix} = \mathbf{J^{-1}} \begin{bmatrix} dx \\ dy \end{bmatrix} \tag{4.4}$$

L'inversion numérique des matrices carrées ne pose pas de difficultés particulières. Pour plusieurs problèmes, pourtant, la matrice jacobienne ne sera pas carrée. Pour pouvoir inverser une jacobienne non carrée il faut utiliser une technique de pseudo-inverse ($(\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T$).

4.2 Une méthode de transformation de modèles

Dans ce paragraphe nous présentons une méthode de transformation de modèles qui permet l'acquisition d'un nouveau modèle à partir d'autres modèles similaires existants.

En fait cette méthode de transformation est équivalente à la méthode d'acquisition de modèles proposée au chapitre 3. La seule différence est que notre méthode initiale était destinée à modéliser les fonctions de l'ensemble $\mathcal{F}_n$ décrit par l'équation 3.10 et cette deuxième méthode sert à modéliser les fonctions contenues d'ensemble $\mathcal{F}_n^*$ (équation 4.5).

$$\begin{aligned} \mathcal{F}_n^* = \{ \quad & f \in \mathcal{F} \text{ telle que } f(x) = \varphi_0(x).\psi_0 + \varphi_1(x).\psi_1 + ... + \varphi_n(x).\psi_n \\ & \text{où } \varphi_0, \varphi_1, ..., \varphi_n \text{ sont des fonctions quelconques et} \\ & \psi_0, \psi_1, ..., \psi_n \text{ sont des scalaires}\} \end{aligned} \tag{4.5}$$

x représente une variable ou un vecteur ayant un nombre quelconque de variables.

Les cas qui nous intéressent ici sont ceux où à partir d'un seul ensemble donné de fonctions $\{\varphi_0(x), \varphi_1(x), ..., \varphi_n(x)\}$ on peut obtenir un ensemble de fonctions de $\mathcal{F}_n^*$: $\{f_0(x), ..., f_m(x)\}$ en utilisant des ensembles distincts de scalaires: $\{\{\psi_0^0, ..., \psi_n^0\}, ..., \{\psi_0^m, ..., \psi_n^m\}\}$, ou autrement dit où $\{\varphi_0(x), \varphi_1(x), ..., \varphi_n(x)\}$ forme la base d'un espace vectoriel de fonctions.

Le but de la méthode est d'acquérir un modèle d'une fonction:

$$f_m(x) = \sum_{i=0}^{n} \varphi_i(x).\psi_i^m$$

à partir de:

- un ensemble de fonctions $\{f_{i_0}(x), ..., f_{i_n}(x)\}$ déjà modélisées,
- plus un ensemble de points (données expérimentales) $\{f_m(x_0), ..., f_m(x_n)\}$.

sans connaître pour autant les termes de la base φ_i.

4.2.1 La transformation DC

Nous allons définir une transformation DC:

$$DC^{(x_a, i_a)} f_m(x) = f_m(x) - \frac{f_{i_a}(x).f_m(x_a)}{f_{i_a}(x_a)} \tag{4.6}$$

avec x_a et i_a tels que $f_{i_a}(x_a) \neq 0$.

Pour simplifier les notations nous allons utiliser $DC^a f$ à la place de $DC^{(x_a, i_a)} f$.

Comme pour la transformation DP nous voulons montrer que la transformation DC utilisée de façon récursive peut imiter exactement toute fonction appartenant à l'ensemble $\mathcal{F}_n^*$.

Pour montrer cela on va définir un ensemble d'ensembles Γ^* (4.7).

$$\left\{ \begin{array}{l} \Gamma_0^* = \{f_m \in \mathcal{F} \text{ et } \exists(x_0, i_0) \text{ tel que } \exists DC^0 f_m \text{ et } DC^0 f_m = 0\} \\ \Gamma_1^* = \{f_m \in \mathcal{F} \text{ et } \exists(x_1, i_1) \text{ tel que } DC^1 f_m \text{ existe et appartient à } \Gamma_0^*\} \\ \Gamma_2^* = \{f_m \in \mathcal{F} \text{ et } \exists(x_2, i_2) \text{ tel que } DC^2 f_m \text{ existe et appartient à } \Gamma_1^*\} \\ \vdots \\ \Gamma_n^* = \{f_m \in \mathcal{F} \text{ et } \exists(x_n, i_n) \text{ tel que } DC^n f_m \text{ existe et appartient à } \Gamma_{n-1}^*\} \end{array} \right\} \tag{4.7}$$

Le premier des ensembles de 4.7, Γ_0^*, contient toutes les fonctions de $\mathcal{F}$ pour lesquelles la transformation $DC^0 f$ est identiquement nulle. De façon récursive, Γ_1^* contient toutes les fonctions de $\mathcal{F}$ pour lesquelles la transformation $DC^0 f$ définit une fonction qui à son tour appartient a l'ensemble antérieur Γ_0^*.

Nous voulons prouver le théorème suivant:

Théorème: Pour toute fonction f_m et tout n, s'il existe une suite $(x_0, i_0), ..., (x_n, i_n)$ de couples distincts telle que:

- $f_{i_0}(x_0) \neq 0$,
- $DC^0 f_{i_1}(x_1) \neq 0, \ldots$,
- $DC^n DC^{n-1} ... DC^0 f_{i_n}(x_n) \neq 0$

alors:

$$f_m \in \Gamma_n^* \Leftrightarrow f_m \in \mathcal{F}_n^*$$

Il faut remarquer que dans cette nouvelle formulation la récurrence joue sur le choix des modèles connus qui vont être utilisés. Ainsi:

$$DC^1 DC^0 f_m(x) = DC^0 f_m(x) - \frac{DC^0 f_{i_1}(x).DC^0 f_m(x_1)}{DC^0 f_{i_1}(x_1)} \tag{4.8}$$

où $DC^0 f_{i_1}(x)$ peut s'écrire comme:

$$DC^0 f_{i_1}(x) = f_{i_1}(x) - \frac{f_{i_0}(x).f_{i_1}(x_0)}{f_{i_0}(x_0)} \tag{4.9}$$

On voit que ce terme dépend des modèles des fonctions f_{i_0} et f_{i_1}.

La preuve de ce théorème est similaire à celle décrite aux paragraphes 3.3.1 et 3.3.2. Elle est donnée dans l'annexe E. La transformation DC est en fait la même transformation DP, qui dans cette deuxième version n'utilise plus les fonctions de forme mais un ensemble de modèles connus de fonctions similaires plus un ensemble de données expérimentales de la nouvelle fonction qu'on veut modéliser.

4.2.2 Utilisation de la méthode de transformation

Nous appelons la méthode presentée dans ce paragraphe notre *méthode de transformation* parce qu'elle permet l'acquisition d'un nouveau modèle à partir d'un ensemble de modèles "similaires" existants. La "similitude" de modèles a un sens très précis dans ce context, qui est déterminée par l'existence d'un ensemble commun de fonctions $\{\varphi_0, \varphi_1, ..., \varphi_n\}$. Si cet ensemble de modèles similaires existe, le modèle d'une nouvelle fonction peut être acquis à partir d'un ensemble de n points (données expérimentales) de cette nouvelle fonction (n étant le nombre de termes des fonctions similaires).

La méthode de transformation va être très utile dans les cas suivants:

- **les caractéristiques d'un système déjà modélisé changent**: la méthode de transformation va permettre une espèce de "calibrage" des modèles acquis par la méthode d'identification structurelle. Tout changement des caractéristiques d'un système modélisé qui preserve la dépendance entre les variables d'entrée (qui garde le nombre de termes des polynômes du modèle et la structure des fonctions des forme) peut être modélisé par une transformation à partir d'un nombre de "versions" initiales du modèle et d'un nombre de "points de calibrage";

- **on doit modéliser plusieurs objets ayant les mêmes caractéristiques**: on fait l'acquisition par identification structurelle d'un nombre initial de modèles et on obtient les suivants par transformation à partir d'un nombre beaucoup plus réduit de données expérimentales;

- **on fait l'acquisition d'un modèle par identification structurelle d'une fonction ayant plus de trois variables d'entrée**: pendant l'acquisition d'un modèle par la méthode d'identification structurelle il est possible d'utiliser la méthode de transformation pour limiter le nombre de fois qu'un "sous modèle" doit être acquis.

La transformation de sous modèles

Soit une fonction $f(x,y,z,w)$. Nous allons appeler *sous-modèle* de f acquis par la méthode toute fonction intermédiaire construite du type de $f(x,y,z,w_0)$. Un sous-modèle est donc déterminé par un sous-ensemble de l'ensemble de variables du modèle. Une fonction de forme, par exemple, est un sous-modèle à une variable d'entrée.

Si on écrit la transformation DPf on obtient:

$$DP^0 f(x,y,z,w) = f(x,y,z,w) - \frac{f(x,y,z,w_0).f(x_0,y_0,z_0,w)}{f(x_0,y_0,z_0,w_0)} \tag{4.10}$$

Le sous-modèle $f(x,y,z,w_0)$ peut être vu comme une fonction $g(x,y,z)$, qu'on peut modéliser en utilisant une DPg:

$$DP^0 g(x,y,z) = f(x,y,z,w_0) - \frac{f(x,y,z_0,w_0).f(x_0,y_0,z,w_0)}{f(x_0,y_0,z_0,w_0)} \tag{4.11}$$

La DP^1DP^0g va être:

$$DP^1DP^0 g(x,y,z) = DP^0 g(x,y,z) - \frac{DP^0 g(x,y,z_1).DP^0 g(x_1,y_1,z)}{DP^0 g(x_1,y_1,z_1)} \tag{4.12}$$

Le terme $DP^0 g(x,y,z_1)$ va être:

$$DP^0 g(x,y,z_1) = f(x,y,z_1,w_0) - \frac{f(x,y,z_0,w_0).f(x_0,y_0,z_1,w_0)}{f(x_0,y_0,z_0,w_0)} \tag{4.13}$$

Si on revient à f, et on écrit DP^1DP^0f:

$$DP^1DP^0 f(x,y,z,w) = DP^0 f(x,y,z,w) - \frac{DP^0 f(x,y,z,w_1).DP^0 f(x_1,y_1,z_1,w)}{DP^0 g(x_1,y_1,z_1,w_1)} \tag{4.14}$$

Où le terme $DP^0 f(x, y, z, w_1)$ va être:

$$DP^0 f(x, y, z, w_1) = f(x, y, z, w_1) - \frac{f(x, y, z_0, w_1).f(x_0, y_0, z, w_1)}{f(x_0, y_0, z_0, w_1)} \tag{4.15}$$

Cette suite contient quelques-uns des pas à suivre dans l'acquisition du modèle de f. Si dans ces équations on fait attention aux sous-modèles qui ont x, y comme hyperespace d'entrée on trouvera: $f(x, y, z_0, w_0)$ (équation 4.11), $f(x, y, z_1, w_0)$ (équation 4.13) et $f(x, y, z_0, w_1)$ (équation 4.15). Ce sont les "versions" des sous-modèles en x, y qui vont se multiplier pendant l'acquisition de f. Le nombre total de versions de ces sous-modèles dependera des niveaux de recurrence de la transformation. Le nombre total de variables du modèle détermine aussi les nombres de versions nécessaires de chaque sous-espace.

La sortie des sous modèles en x, y peut être décrite par l'équation 4.16:

$$h(x, y) = \sum_{i=0}^{n} \varphi_i(x).\psi_i(y).\rho_i(z_j).\sigma_i(w_k) \tag{4.16}$$

où les indices j et k sont determinés par l'algorithme de la méthode.

Si on note: $\lambda_i(x, y) = \varphi_i(x).\psi_i(y)$ et $\phi_i^{jk} = \rho_i(z_j).\sigma_i(w_k)$ on peut écrire h comme:

$$h(x, y) = \sum_{i=0}^{n} \lambda_i(x, y).\phi_i^{jk} \tag{4.17}$$

on constate que $h \in \mathcal{F}^*$.

Il faut remarquer que l'équation 4.17 est une description analytique des sous modèles (en particulier de ceux déterminés par deux variables) mais qu'en pratique on n'a pas accès aux termes $\lambda_i(x, y)$. On dispose seulement des valeurs de x, y et de la valeur de $h(x, y)$ pour certains points (données expérimentales). Si on connaisait les valeurs des $\lambda_i(x, y)$ l'estimation des ϕ_i^{jk} d'un nouveau sous-modèle serait la solution d'un système d'équations simple. Nous avons déjà fait cette remarque en parlant de l'algorithme de la méthode. Les pas intermédiaires de l'acquisition ne correspondent pas aux termes du polynôme qu'on modélise.

De façon générale, un sous modèle a son ensemble de variables (son hyperespace), et dehors il y a l'ensemble des variables restantes (encore non rajoutées) du modèle. Plus grand est cet ensemble restant, plus nombreuses seront les versions du sous modèle. Chaque version est déterminée par un vecteur distinct de valeurs constantes attribuées aux variables restantes. Mais à la fin toutes ces versions peuvent s'écrire sous la forme de l'équation 4.17.

Nous pouvons, donc, utiliser la méthode de transformation aux sous-modèles. On doit faire l'acquisition d'un ensemble initial de n versions (avec DP), où n est le niveau de recurrence nécessaire pour acquerir le sous-modèle, et on obtient les versions suivantes par transformation.

En pratique, si on considère le rapport *nombre de données expérimentales nécessaires à l'acquisition d'un modèle / nombre de variables du modèle*, **la méthode de transformation linéarise l'évolution de ce rapport.** Une analyse quantitative detaillée est présentée au paragraphe 4.5.

L'adaptation des modèles existants

Pour montrer comment fonctionne l'adaptation des modèles existants en utilisant la méthode de transformation de modèles nous allons reprendre l'illustration du bras planaire. Comme pour la présentation de la méthode nous allons considérer qu'un "Deus ex machina" nous donne les formes analytiques des modèles et des fonctions de forme. Cela nous facilite l'illustration, mais ces informations ne sont pas nécessaires dans l'application pratique de la méthode.

Supposons qu'on dispose de deux modèles du bras planaire:

- modèle 0: (longueur des segments = 1 et 2) $f_0(\theta_1, \theta_2) = cos(\theta_1) + 2 * cos(\theta_1 + \theta_2)$
- modèle 1: (longueur des segments = 2 et 3) $f_1(\theta_1, \theta_2) = 2 * cos(\theta_1) + 3 * cos(\theta_1 + \theta_2)$

Nous voulons obtenir un modèle f_2 d'un bras planaire similaire aux bras modèlisés par f_0 et f_1, mais qui a la longueur de ses segments = 3 et 4.

La transformation $DC^0 f_2$ aura la forme:

$$DC^0 f_2(\theta_1, \theta_2) = f_2(\theta_1, \theta_2) - \frac{f_0(\theta_1, \theta_2).f_2(\theta_1^0, \theta_2^0)}{f_0(\theta_1^0, \theta_2^0)} \qquad (4.18)$$

On peut aussi écrire la transformation $DC^1DC^0f_2$:

$$DC^1DC^0f_2(\theta_1,\theta_2) = DC^0f_2(\theta_1,\theta_2) - \frac{DC^0f_1(\theta_1,\theta_2).DC^0f_2(\theta_1^1,\theta_2^1)}{DC^0f_1(\theta_1^1,\theta_2^1)} \tag{4.19}$$

où $DC^0f_1(\theta_1,\theta_2)$ peut s'écrire comme:

$$DC^0f_1(\theta_1,\theta_2) = f_1(\theta_1,\theta_2) - \frac{f_0(\theta_1,\theta_2).f_1(\theta_1^0,\theta_2^0)}{f_0(\theta_1^0,\theta_2^0)} \tag{4.20}$$

Nous connaissons les modèles f_0 et f_1, et nous savons qu'ils ont deux termes (ils apartiennent à $\mathcal{F}_1$). Nous modélisons un bras similaire, donc nous pouvons conclure que f_2 appartient aussi à $\mathcal{F}_1$ et à $\mathcal{F}_1^*$. En conséquence $DC^1DC^0f_2(\theta_1,\theta_2) = 0$.

Nous allons remplacer dans l'équation 4.19 les termes décrits par les équations 4.18 et 4.20:

$$\begin{aligned} & f_2(\theta_1,\theta_2) - \frac{f_0(\theta_1,\theta_2).f_2(\theta_1^0,\theta_2^0)}{f_0(\theta_1^0,\theta_2^0)} \\ & -\frac{(f_1(\theta_1,\theta_2) - \frac{f_0(\theta_1,\theta_2).f_1(\theta_1^0,\theta_2^0)}{f_0(\theta_1^0,\theta_2^0)}).(f_2(\theta_1^1,\theta_2^1) - \frac{f_0(\theta_1^1,\theta_2^1).f_2(\theta_1^0,\theta_2^0)}{f_0(\theta_1^0,\theta_2^0)})}{f_1(\theta_1^1,\theta_2^1) - \frac{f_0(\theta_1^1,\theta_2^1).f_1(\theta_1^0,\theta_2^0)}{f_0(\theta_1^0,\theta_2^0)}} = 0 \end{aligned} \tag{4.21}$$

Si on choisit $\theta_1^1 = 0$,$\theta_1^1 = \pi/2$, $\theta_2^1 = 0$ et $\theta_2^1 = \pi/2$:

- on a besoin de deux données expérimentales du nouveau bras (nous considérons que ces deux valeurs peuvent être obtenues):
  - ⋆ $f_2(\theta_1^0,\theta_2^0) = 7$
  - ⋆ $f_2(\theta_1^1,\theta_2^1) = -4$
- on a besoin de deux valeurs calculées par le premier modèle :
  - ⋆ $f_0(\theta_1^0,\theta_2^0) = 3$

- $\star$ $f_0(\theta_1^1, \theta_2^1) = -2$

- et on a besoin de deux valeurs calculées par le deuxième modèle :
  - $\star$ $f_1(\theta_1^0, \theta_2^0) = 5$
  - $\star$ $f_1(\theta_1^1, \theta_2^1) = -3$

L'équation 4.21 devient alors:

$$f_2(\theta_1, \theta_2) = \frac{f_0(\theta_1, \theta_2).7}{3} + \frac{(f_1(\theta_1, \theta_2) - \frac{f_0(\theta_1,\theta_2).5}{3}).(-4 - \frac{-2.7}{3})}{-3 - \frac{-2.5}{3}} \tag{4.22}$$

Nous allons faire les substitutions des équations connues des modèles f_1 et f_2:

$$f_2(\theta_1, \theta_2) = \frac{7.(cos(\theta_1) + 2.cos(\theta_1 + \theta_2))}{3} + \\ \frac{(2.cos(\theta_1) + 3.cos(\theta_1 + \theta_2) - \frac{5.(cos(\theta_1)+2.cos(\theta_1+\theta_2))}{3}).(2/3)}{1/3}$$

Après quelques simplifications on trouve:

$$\begin{aligned} f_2(\theta_1, \theta_2) = \quad & \tfrac{7}{3}.cos(\theta_1) + \tfrac{14}{3}.cos(\theta_1 + \theta_2) + 4.cos(\theta_1) + \\ & 6.cos(\theta_1 + \theta_2) - \tfrac{10}{3}.cos(\theta_1) - \tfrac{20}{3}.cos(\theta_1 + \theta_2) \end{aligned}$$

Et finalement:

$$f_2(\theta_1, \theta_2) = 3.cos(\theta_1) + 4.cos(\theta_1 + \theta_2) \tag{4.23}$$

Nous avons trouvé le modèle f_2 du bras de longueurs de segments égales à 3 et 4, en utilisant seulement deux données expérimentales ($f_2(0,0)$ et $f_2(\pi/2, \pi/2)$) et deux modèles connus auparavant (f_0 et f_1). Bien sur la transformation donne les mêmes résultats pour n'importe quel choix de θ_1^0,θ_1^1,θ_2^0 et θ_2^1 (si ces valeurs respectent les conditions de non nullité de la transformation).

4.3 Analyse de l'erreur d'estimation d'un modèle

Dans ce paragraphe nous allons présenter une approche qui va nous permettre d'estimer les marges de l'erreur contenue dans les sorties des modèles acquis par notre méthode.

L'estimation, même approximative, des marges de l'erreur des sorties calculées par le modèle direct peut être utile comme indice de la fiabilité du modèle. Cependant, quand le modèle est utilisé pour asservir le système, la manque de précision du modèle direct ne compromet pas nécessairement la précision de l'inversion du modèle.

Les erreurs d'un modèle sont produites soit par une caractérisation imparfaite, soit par une identification imparfaite des paramètres. Dans le contexte de notre méthode, la caractérisation faite par le concepteur est restreinte aux fonctions de forme. Pendant le processus automatique d'identification structurelle un seul choix peut être considéré comme étant une caractérisation du modèle: le choix du nombre de récurrences de la transformation DP. Nous allons considérer ce choix comme étant parfait. Nous allons aussi considérer que la caractérisation des fonctions de forme est adéquate, c'est-à-dire, que le concepteur a choisi une forme paramétrique qui est idéale (voir 3.5.2) ou au moins il a choisi un outil ou une classe de fonctions interpolatrices assez puissante pour réprésenter la fonction de forme avec une précision donnée.

Dans ces conditions nous allons analyser les erreurs des modèles comme étant conséquence d'une identification imparfaite des paramètres des fonctions de forme. L'identification paramétrique des fonctions de forme va dépendre fortement des données expérimentales, obtenues directement du système modélisé. S'il s'agit de la modélisation d'un système physique (non simulé) ces données expérimentales vont contenir toujours un certain niveau de bruit, qui peut être plus au moins important en fonction des caractéristiques du processus d'acquisition des données. Ce bruit va entraîner un écart sur l'estimation paramétrique des fonctions de forme. Les erreurs d'estimation vont être propagées à travers le modèle jusqu'aux sorties calculées.

L'estimation des marges d'erreur du modèle comporte deux étapes: l'estimation des marges d'erreur des sorties des fonctions de forme, que nous allons analyser au paragraphe 4.3.1, et la propagation de ces valeurs jusqu'à la sortie du modèle, qui nous allons analyser au paragraphe 4.3.3. Au paragraphe 4.3.2 nous analysons les fonctions de sensibilité des polynômes interpolateurs.

4.3.1 Estimation des marges d'erreur des fonctions de forme

Pour chaque fonction de forme, un ensemble de données expérimentales va être utilisé pour réaliser une identification des valeurs d'un vecteur de paramètres. La fonction de forme va donc être représentée par un point O (nominal) dans un espace paramétrique. L'existence du bruit d'état dans les données empêche en pratique l'identification du point dans cette espace paramétrique relatif à l'objet. Cependant, si une identification idéale était possible, pour un point n de l'espace d'entrée, l'écart w entre les sorties s_O de la fonction de forme et les sorties s_M de son modèle ne pourrait représenter que du bruit d'état.

$$s_O(n) = s_M(n, p) + w(n) \tag{4.24}$$

On peut considérer qu'un autre modèle $s_M(n, p + \Delta p)$, Δp étant une variation des paramètres du modèle, aurait pu donner la sortie bruitée de s_O en n.

$$s_O(n) = s_M(n, p + \Delta p)$$

Un développement au premier ordre permet d'écrire:

$$s_O(n) = s_M(n, p) + \sum_{k=1}^{K} \Delta p_k \sigma_k(n) \tag{4.25}$$

$\{\sigma_k\}_{k=1,K}$ étant des *fonctions de sensibilité*. Dans le cas d'un modèle qui contient un vecteur de paramètres p, une fonction de sensibilité σ_k relative au paramètre p_k aura la forme:

$$\sigma_k = \frac{\partial s_M}{\partial p_k} \tag{4.26}$$

Une fonction de sensibilité peut être donnée par la différentiation de la forme paramétrique de la fonction de forme, ou calculé numériquement pour un point n donnée par:

$$\sigma_k(n) = \frac{s_M(n, p_k + \delta p_k) - s_M(n, p_k)}{\delta p_k} \; (k = 1, ..., K)$$

les paramètres $p_i : i \neq k$ restants inchangés.

D'après 4.24 et 4.25 on peut écrire:

$$w(n) = \sum_{k=1}^{K} \Delta p_k \sigma_k(n) \tag{4.27}$$

soit:

$$w(n) = \Delta p^T \xi(n) \tag{4.28}$$

où:

$$\begin{aligned} \Delta p &= (\Delta p_1, \Delta p_2, ..., \Delta p_K)^T \\ \xi(n) &= [\sigma_1(n), \sigma_2(n), ..., \sigma_K(n)]^T \end{aligned}$$

On ne connaît toujours pas le modèle idéal, pas plus que le vecteur Δp. Une méthode d'identification paramétrique peut calculer un point qu'appartient à un domaine iso-distance déterminé par le bruit. Supposant que nous nous intéressons à une estimation même approximative du bruit contenu dans les sorties des fonctions de forme, nous allons supposer que le point relatif à l'objet sera quelque part proche du centre de l'iso-domaine déterminé par le bruit, et qu'une estimation au sens des moindres carrés d'un ensemble d'essais d'identification des paramètres serait une bonne approximation de ce centre.

Pour estimer les marges d'erreur des sorties d'une fonction de forme nous allons utiliser le bruit calculé par 4.28. A la place du vecteur Δp on utilise un vecteur Δp_{max} avec les écarts maximaux entre la valeur estimée de p_O et les valeurs des paramètres de l'ensemble d'essais d'identification.

$$\Delta p_{max_k} = max \| p_{O_k} - p_{M_{ki}} \| \forall i \ (k = 1, ..., K)$$

L'erreur maximale e_{max} de la fonction de forme au point n de l'espace d'entrée sera alors estimée par:

$$e_{max}(n) = \| \Delta p_{max}^T \xi(n) \| \tag{4.29}$$

étant donnée qu'on est capable d'estimer l'erreur maximale d'une fonction de forme, on veut maintenant calculer l'erreur maximale e'_{max} du calcul de sa dérivée

$\frac{\partial s_M}{\partial x}$ par rapport à sa seule entrée x (à ne pas confondre avec la dérivée de son erreur maximale).

$$e'_{max}(n) = \frac{\partial(s_M(n) + e_{max}(n))}{\partial x} - \frac{\partial s_M(n)}{\partial x} \tag{4.30}$$

D'après 4.29 et 4.30 on obtient:

$$\begin{aligned} e'_{max}(n) &= \frac{\partial(s_M(n) + \|\Delta p_{max}^T \xi(n)\|)}{\partial x} - \frac{\partial s_M(n)}{\partial x} \\ &= \frac{\partial(\|\Delta p_{max}^T \xi(n)\|)}{\partial x} \\ &= \|\Delta p_{max}^T \xi'(n)\| \end{aligned}$$

où:

$$\xi'(n) = [\sigma'_1(n), \sigma'_2(n), ..., \sigma'_K(n)]^T$$

et

$$\sigma'_k(n) = \frac{\partial \sigma_k(n)}{\partial x} \tag{4.31}$$

Encore une fois σ'_k peut être obtenue formellement par differentiation de la fonction de sensibilité σ_k ou numériquement par:

$$\sigma'_k(n, x) = \frac{\sigma_k(n, x + \delta x) - \sigma_k(n, x)}{\delta x}$$

4.3.2 Les fonctions de sensibilité des polynômes interpolateurs

Les polynômes interpolateurs ont été présentés dans 3.5. Si les fonctions de forme sont estimées par ces polynômes alors les fonctions de sensibilité peuvent être analytiquement déterminées.

Les fonctions de sensibilité des polynômes algébriques Les polynômes algébriques ont la forme générale:

$$P(x) = \sum_{k=0}^{N} a_k x^k \tag{4.32}$$

où N est le dégré du polynôme.

Pour les modèles directs la fonction de sensibilité σ_k relative au k-ième paramètre d'une fonction de forme (équation 4.26) va être déterminée par:

$$\begin{aligned} \sigma_k &= x^k \ (k = 1..N) \\ \sigma_0 &= 1 \end{aligned}$$

Pour les modèles différentiels directs la dérivée de la fonction de sensibilité σ'_k relative au k-ième paramètre d'une fonction de forme (équation 4.31) va être déterminée par:

$$\begin{aligned} \sigma'_k &= kx^{k-1} \ (k = 1..N) \\ \sigma'_0 &= 0 \end{aligned}$$

Les fonctions de sensibilité des polynômes trigonométriques Les polynômes trigonométriques ont la forme générale :

$$P(x) = a_1 + \sum_{i=1}^{N} (a_{2i} cos(ix) + a_{2i+1} sin(ix)) \tag{4.33}$$

Pour les modèles directs la fonction de sensibilité σ_k relative au paramètre a_k d'une fonction de forme va être déterminée par:

$$\begin{aligned} \sigma_k &= cos(ix) \ (k = 2, 4, ..., 2.N; i = k/2) \\ \sigma_k &= sin(ix) \ (k = 3, 5, ..., 2.N + 1; i = (k-1)/2) \\ \sigma_1 &= 1 \end{aligned}$$

Pour les modèles différentiels directs la dérivée de la fonction de sensibilité σ'_k relative au paramètre a_k d'une fonction de forme va être déterminée par:

$$\begin{aligned} \sigma_k &= -sin(ix)i \ (k = 2, 4, ..., 2.N; i = k/2) \\ \sigma_k &= cos(ix)i \ (k = 3, 5, ..., 2.N + 1; i = (k-1)/2) \\ \sigma_1 &= 0 \end{aligned}$$

4.3.3 Propagation des marges d'erreur

Les marges d'erreur estimées pour les fonctions de forme doivent être propagées à travers la structure du modèle jusqu'aux sorties. Nous allons maintenant dériver pour chaque opération algébrique le calcul des marges d'erreur en fonction des marges d'erreur de ses entrées.

Pour la somme:

$$\begin{aligned} s^{+} &= a+b \\ s^{+}_{max} &= (a+e^{a}_{max})+(b+e^{b}_{max}) \\ e^{+}_{max} &= s^{+}_{max}-s^{+}e^{+}_{max}=e^{a}_{max}+e^{b}_{max} \end{aligned}$$

Pour la différence:

$$\begin{aligned} s^{-} &= a-b \\ s^{-}_{max} &= (a+e^{a}_{max})-(b-e^{b}_{max}) \\ e^{-}_{max} &= s^{-}_{max}-s^{-}e^{-}_{max}=e^{a}_{max}+e^{b}_{max} \end{aligned}$$

Pour le produit:

$$\begin{aligned} s^{*} &= a*b \\ s^{*}_{max} &= (\|a\|+e^{a}_{max})*(\|b\|+e^{b}_{max})*signe(a)*signe(b) \\ e^{*}_{max} &= s^{*}_{max}-s^{*} \\ e^{*}_{max} &= signe(a)*b*e^{a}_{max}+signe(b)*a*e^{b}_{max}+e^{a}_{max}*e^{b}_{max} \end{aligned}$$

où $signe(x)=-1$ si $x<0$ et $signe(x)=1$ au cas contraire. ($signe(x)*\|x\|=x$).

Pour la division:

$$\begin{aligned} s^{\div} &= a/b \\ s^{\div}_{max} &= (signe(a)*(\|a\|+e^{a}_{max}))/(signe(b)*(\|b\|-e^{b}_{max})) \\ e^{\div}_{max} &= s^{\div}_{max}-s^{\div} \\ e^{\div}_{max} &= \frac{signe(a)*b*e^{a}_{max}+signe(b)*a*e^{b}_{max}}{b^{2}-\|b\|*e^{b}_{max}} \end{aligned}$$

Les mêmes formes de propagation sont utilisées pour les marges d'erreur du modèle direct et du modèle différentiel direct.

4.4 Réduction de l'erreur d'estimation d'un modèle

Les équations du paragraphe 4.3.3 laissent entrevoir une expansion très rapide de l'erreur dans le modèle. L'arbre du modèle acquis peut contenir centaines d'opérations, et effectivement les erreurs d'estimation des fonctions de forme à la base du modèle augmentent de façon presque exponentielle par la suite des calculs.

Cette caractéristique négative des modèles s'explique par le fait d'avoir un espace paramètrique réduit (nombre réduit de paramètres) par rapport au nombre d'opérations de calcul faites à partir des valeurs de ces paramètres. Un écart, même réduit, du point estimé de l'espace paramétrique par rapport au point idéal peut entraîner un erreur importante sur les sorties du modèle.

Pour minimiser ce problème, et pour pouvoir acquérir un modèle d'une précision désirée à partir des données bruitées, nous proposons une technique d'adaptation des modèles, qui va permettre la réduction de l'écart dans l'espace paramétrique du modèle.

Cette technique consiste à étendre le concept des fonctions de sensibilité, qui ont été présentés en 4.3.1. Dans ce paragraphe nous avons utilisé les fonctions de sensibilité pour estimer les marges d'erreur des fonctions de forme. Les fonctions de sensibilité d'une fonction de forme sont en fait ses dérivés partielles par rapport à chacun de ses paramètres.

Nous pouvons utiliser le même procédé que celui présenté au paragraphe 4.1 pour propager les dérivés partielles des fonctions de forme jusqu'aux sorties du modèle. On obtient ainsi les fonction de sensibilité du modèle entier, ou plus précisément, on dispose des dérivés partielles de chaque sortie du modèle par rapport à chaque paramètre qui détermine cette sortie [1].

Nous pouvons maintenant définir un Jacobien de l'espace paramétrique. Ce Jacobien va mettre en relation un ensemble de points de l'hyperespace d'entrée du modèle et l'ensemble des paramètres du modèle. Prenons une sortie isolée x du modèle direct: $x = h(q, p)$, q étant le vecteur des entrées et p le vecteur des paramètres du modèle h. Le Jacobien de l'espace paramétrique va être défini par:

$$J_{ij} = \frac{\partial h(q_i, p)}{\partial p_j} \; (i = 1, ..., m; \; j = 1, ..., n)$$

[1] Nous construisons un modèle pour chaque sortie du modèle, donc chaque sortie est définie par un ensemble distinct de paramètres

où: J_{ij} définit l'élément de la i-nième ligne et j-ième colonne de la matrice jacobienne, m étant le nombre de points considérés et n le nombre de paramètres du système.

Pour chaque point pris en considération dans le Jacobien de l'espace paramétrique il existe une différence entre sa valeur calculée par le modèle et sa valeur réelle (pour la sortie en question). Nous appelons D le vecteur des différences. Nous suivons le même procédé utilisé pour l'asservissement du système. La matrice jacobienne J est inversée (par une technique de pseudo inversion) et en suite $J^{\dagger}$ est multiplié par D^T.

$$P = J^{\dagger} D^T$$

Le vecteur P sera multiplié par un gain d'asservissement g. $P.g$ nous donnera les valeurs des corrections qu'on va appliquer sur le vecteur p des paramètres du modèle. Les corrections doivent réduire l'écart dans l'espace paramétrique et donc réduire les différences entre les valeurs calculées et réelles pour les points choisis. Le procedé peut être répeté dans une boucle jusqu'à que les différences soient dans des limites acceptables.

La technique d'adaptation doit être appliquée sur un ensemble des points où se trouvent:

- les points d'interpolation (pour lesquels l'erreur d'estimation est nulle)
- plus un nouvel ensemble de points de l'hyperespace du modèle. L'estimation faite par le modèle pour ces points contient normalement un erreur non nulle de calcul qu'on veut réduire.

De cette façon on peut être sûr que le modèle adapté est meilleur que le modèle initial.

La technique d'adaptation peut être lourde et instable si elle est appliquée pour un modèle ayant un grand nombre de paramètres, ou de points, ou encore s'il existe un grand écart initial dans l'espace paramétrique.

Nous avons utilisé la technique d'adaptation de modèles dans le procédé d'acquisition. Après l'acquisition de chaque sous modèle nous faisons l'adaptation des paramètres spécifiques au sous modèle. L'avantage est qu'il s'agit à chaque adaptation d'un ensemble réduit de paramètres à corriger et de points à considérer. Les écarts à corriger sont aussi réduits. Pour chaque variable d'entrée rajoutée au modèle on corrige seulement les paramètres des fonctions de forme de cette variable. Les points considerés sont les points d'interpolation spécifiques à la variable rajoutée plus un nombre de points d'adaptation choisis dans l'hyperespace du sous modèle.

Dans le chapitre 5 nous allons montrer les résultats expérimentaux obtenus en utilisant la technique d'adaptation.

4.5 Analyse quantitative de la méthode

Dans ce paragraphe nous analyserons la quantité de données nécessaires à l'acquisition des modèles. Le nombre de points que l'on doit obtenir du système à modéliser est un facteur déterminant de l'utilité d'une méthode, surtout quand il s'agit des systèmes physiques. L'acquisition des données d'un système physique peut être coûteuse.

Le nombre de points à obtenir pendant l'acquisition de nos modèles sera déterminé par quatre facteurs:

- le nombre de points nécessaires à l'identification des paramètres des fonctions de forme,
- le nombre de variables d'entrée,
- l'utilisation des transformations de sous modèles
- et les niveaux de récurrence du modèle et de ses sous modèles

.

Le nombre de points nécessaires à l'acquisition d'un modèle ne dépend pas directement du nombre de variables de sortie du modèle. Un modèle est acquis pour chaque variable de sortie, mais tous les modèles utilisent les mêmes points d'interpolation (même point dans l'hyperespace d'entrée). Si les niveaux de récurrence ou la forme paramétrique diffèrent entre les modèles on doit considérer les valeurs maximales de chaque facteur pour le calcul du nombre de points d'interpolation à obtenir.

Comme nous l'avons déjà dit en 3.5, le nombre de points d'interpolation nécessaires à l'identification paramétrique d'une fonction de forme dépend du type d'approximation choisie (interpolation de points, fonction interpolatrice). Pour chaque choix de type d'approximation suivra un certain nombre de choix plus spécifiques (configuration de l'outil d'interpolation, dégré du polynôme interpolateur, etc.). Les choix peuvent être faits cas par cas (variable par variable). Nous n'allons pas entrer dans les détails, et considérer N_i comme le nombre de points nécessaires à l'identification des paramètres d'une fonction de forme pour la variable d'entrée i.

Pour l'acquisition d'un modèle (ou sous modèle) à deux variables d'entrée nous avons besoin de $2 * M$ fonctions de forme, où M est le niveau de récurrence du modèle. C'est-à-dire $M * (N_1 + N_2)$ points d'interpolation au maximum car en pratique un seul point de l'hyperespace d'entrée peut servir à l'identification des paramètres de plusieurs fonctions de forme (pour de variables d'entrées différentes).

Pour l'acquisition d'un modèle (ou sous modèle) à i variables ($i > 2$) on doit disposer de M_i fonctions de forme pour la i-ième variable et M_i sous modèles à $i - 1$ variables. (M_i étant le niveau de récurrence du modèle (ou sous modèle) à i variables). Le nombre de points d'interpolation nécessaires à l'acquisition du modèle est donnée par 4.34.

$$\begin{aligned} p_i &= M_i.(p_{i-1} + N_i) \ (i > 2) \\ p_2 &= M_2.(N_1 + N_2) \end{aligned} \tag{4.34}$$

Si on considère M le niveau de récurrence du modèle et de tous les sous modèles, N le nombre de points d'interpolation de toutes les fonctions de forme et I le nombre de variables d'entrée du modèle, on obtient une expression plus claire pour le calcul du nombre maximum de points d'interpolation.

$$p = N.(2M^{I-1} + \sum_{i=1}^{I-2} M^i) \tag{4.35}$$

Pour l'acquisition des sous modèles on peut utiliser la méthode de transformation proposée au paragraphe 4.2. L'analyse du nombre de points d'interpolation nécessaires à l'acquisition des modèles aura donc deux quantités à considérer. La première est le nombre de sous modèles nécessaires à l'acquisition du modèle et la deuxième est le nombre de points nécessaires à l'acquisition de l'ensemble des sous modèles. Les sous modèles peuvent être regroupés par leur nombre de variables d'entrée, et le nombre total de sous modèles par groupe peut être décrit par l'équation 4.36.

$$\begin{aligned} s_i &= M_{i+1}^2 \ (i = 1..I - 2) \\ s_{I-1} &= M_I \\ s_I &= 1 \end{aligned} \tag{4.36}$$

s_i étant le nombre de sous modèles à i variables d'entrée nécessaires à l'acquisition du modèle à I variables d'entrée.

L'équation 4.36 est déterminée par la transformation de modèles. Un modèle à $i+1$ variables doit être acquis un nombre M_{i+1} de fois, pour pouvoir être calculé par transformation par la suite. Chaque acquisition va utiliser M_{i+1} sous modèles à i variables.

Le nombre p de points d'interpolation sera donnée par:

$$p = \sum_{i=1}^{I} n_i \tag{4.37}$$

n_i étant le nombre total de points d'interpolation utilisés spécifiquement pour l'acquisition des sous modéles du groupe s_i (c'est-à-dire sans considérer à chaque niveau de récurrence les points utilisés pour le niveau antérieur). n_i est donné par l'équation 4.38.

$$\begin{aligned} n_I &= M_I.N_I \\ n_i &= M_i^2.N_i + (s_i - M_i).M_i \ (i = 2..I-1) \\ n_1 &= s_1.N_1 \end{aligned} \tag{4.38}$$

Si on considère la même simplification utilisée pour l'équation 4.35, l'équation 4.37 devient:

$$p = M.N + 2.M^2.N + (I-3).(M^2.N + (M^2 - M).M) \tag{4.39}$$

On peut remarquer que le nombre de points d'interpolation nécessaires à l'acquisition d'un modèle n'est plus exponentiellement dépendent du nombre de variables d'entrée du modèle.

Considérons comme exemple un système defini par 6 variables d'entrée (I=6), un niveau de recursivité égal à 3 à tous les niveaux (M=3) et toutes les fonctions de forme réprésentées par une forme paramétrique à 3 paramètres (N=3)

D'aprés l'équation 4.35 le nombre maximum de points d'interpolation nécessaires à l'acquisition du modèle sans l'utilisation des transformation de sous modèles sera de **1818** points. D'aprés l'équation 4.39 le même modèle peut être acquis avec un maximum de **198** points d'interpolation en utilisant les transformations [2]. Rappelons que couvrir cet hyperespace à 6 dimensions avec des points d'interpolation pour permettre une interpolation linéaire, par exemple, implique en avoir n^6 points, où $n = 30$ entraine une erreur résiduelle maximale de $\approx 1\%$, et $n = 10$ entraine une erreur résiduelle maximale de $\approx 3\%$.

[2]Les valeurs de l'exemple correspondent au modèle d'un bras robot à six degrés de liberté

Chapitre 5

Expérimentations

Dans ce chapitre nous allons présenter une série d'expérimentations faites en utilisant la méthode décrite au chapitre 3. La méthode est assez générale, et permet la modélisation d'une large classe de systèmes, soit par caractérisation idéale, soit par approximation. Pour cette raison, dans le cadre de cette thèse il serait impossible de faire le tour des domaines où la méthode peut être utile. Notre but, dans ce chapitre, est d'illustrer l'application de la méthode et montrer sa mise en œuvre dans le cadre d'applications à la robotique.

Nous avons eu le souci de ne pas rester seulement dans le cadre des expérimentations simulées. Nous pensons que les vrais défis de la modélisation se trouvent dans les domaines réels, et que les simulations informatiques peuvent facilement cacher la vraie complexité des problèmes.

Ce chapitre est divisé en trois paragraphes:

- **5.1**. Dans ce paragraphe nous présentons notre schéma général d'expérimentation. Nous présentons aussi le logiciel général qui a été développé pour mettre en œuvre la méthode.

- **5.2**. Ce paragraphe présente l'application de la méthode à un problème simulé pour lequel existe une caractérisation idéale.

- **5.3**. Ce paragraphe présente l'application de la méthode à un problème réel en robotique: la modélisation d'un système intégré "vision-robotique" et son asservissement.

5.1 Schéma général d'expérimentation

Dans ce paragraphe nous présentons notre méthodologie d'expérimentation. Nous avons développé un logiciel général à partir de la méthode, et nous l'avons utilisé dans un certain nombre de problèmes. Notre souci a été avant tout l'exploration des possibilités d'application de la méthode.

Nous décrivons les étapes de nos expérimentations dans les paragraphes suivants.

5.1.1 Caractérisation

Avant de débuter une modélisation, le concepteur doit préciser certains détails de son système.

Caractérisation initiale. Le concepteur fourni le nombre de variables d'entrée (I) et de sortie (O) du système à modéliser. Le concepteur doit prévoir aussi une liaison avec le système à modéliser pour que le logiciel puisse y chercher les données expérimentales nécessaires.

Caractérisation des fonctions de forme. Parmi les façons possibles de représenter les fonctions de forme présentées en 3.5 le concepteur a le choix entre trois:

- les polynômes algébriques (comme fonctions interpolatrices),
- les sinusoïdes (pour les cas de caractérisation idéale),
- l'interpolation linéaire (comme outil d'interpolation de points).

Pour les polynômes algébriques, le concepteur a le choix du degré du polynôme et pour l'interpolation le nombre de points à utiliser.

5.1.2 Identification structurelle

Nous voulons construire un modèle direct M_o pour chaque sortie o du système. Le logiciel exécutera l'algorithme présenté en 3.2 pour identifier le niveau de récurrence idéal de la transformation DP pour chaque modèle.

L'identification est progressive, c'est-à-dire, les variables d'entrée sont rajoutées une à une aux modèles. Pour chaque variable i rajoutée, un sous-modèle m_{o_i} est déterminé pour chaque sortie o. Pour déterminer chaque sous-modèle l'algorithme d'identification structurelle boucle jusqu'à que le niveau de récurrence idéal soit trouvé. Chaque boucle du procédé d'identification contient trois étapes importantes:

1) Acquisition des fonctions de forme Chaque niveau de récurrence rajouté à la transformation DP demande une nouvelle fonction de forme de la variable rajoutée i. Si le modèle (ou sous modèle) ne contient que deux variables ($i = 2$), chaque niveau de récurrence demandera aussi une fonction de forme de la première variable $(i-1)$.

L'acquisition de chaque fonction de forme est faite par l'identification des paramètres de la forme choisie pendant la caractérisation. L'ensemble des points nécessaires à l'identification paramétrique va être obtenu expérimentalement.

Si le concepteur désire une mesure des marges d'erreur du modèle dues au bruit contenu dans les données, un ensemble redondant de points doit être obtenu pour permettre une série d'essais d'identification de la fonction de forme et définir ainsi sa fonction de sensibilité (voir 4.3.1).

2) Acquisition des sous modèles Pour $i > 2$ chaque niveau de récurrence r rajouté à la transformation DP demande aussi un sous-modèle $m_{o_{i-1}}$ différent.

L'acquisition de ces sous modèles peut être faite par la répétition du procédé à $i-1$ variables. On peut aussi générer les sous modèles en utilisant la méthode de transformation de sous modèles présentée en 4.3. La méthode génère les nouveaux sous modèles $m_{o_{i-1}}$ à partir d'un ensemble de r_{i-1} sous modèles du même sous espace d'entrée $m_{o_{i-1}}$ déjà acquis (r_{i-1} est le niveau de récurrence d'un sous modèle $m_{o_{i-1}}$). Chaque transformation de sous modèle demande aussi r_{i-1} nouveaux points d'interpolation.

3) Test du niveau de récurrence Chaque niveau de récurrence rajouté à la transformation DP sera suivi d'un test du (sous) modèle. Le concepteur peut choisir le nombre de points de test et aussi une marge acceptable d'erreur. Les points de test sont choisis de façon aléatoire dans le (sous) espace d'entrée du (sous) modèle. L'erreur du (sous) modèle sera la différence entre la valeur calculée et la valeur obtenue directement du système à modéliser pour les points de test.

5.1.3 Mise au point du modèle

Construction de l'arbre d'opérations du modèle. Les modèles directs identifiés sont récurrents. Pour optimiser son exécution le logiciel va construire pour chaque modèle un arbre d'opérations ayant la même structure que le modèle à identifier, permettant une exécution séquentielle sans récurrence.

Réduction de l'erreur d'estimation d'un modèle. Pour les modèles acquis à partir de données bruitées nous réalisons la correction des paramètres du modèle en utilisant la technique proposée dans le paragraphe 4.4. Dans la pratique, l'application de la technique de correction est essentielle pour l'acquisition d'un modèle direct précis. Cependant, pour éviter l'instabilité numérique que ce type de technique peut entrainer il faut bien choisir le pas de correction. De façon générale, les pas réduits assurent la convergence de la correction, mais le processus de correction va devoir boucler un grand nombre de fois et demander ainsi un effort de calcul plus important.

5.1.4 Utilisation du modèle

Le modèle direct est prêt. Pour tout point de l'hyperespace d'entrée il peut calculer le point équivalent dans l'hyperespace de sortie.

En utilisant le modèle direct et les règles de différentiation décrites en 4.1 on obtient pour tout point de l'hyperespace d'entrée toutes les dérivées partielles de premier ordre du modèle (le jacobien du point). C'est qu'on appelle le modèle différentiel direct.

Pour la plupart des cas le modèle acquis (direct et différentiel) va être utilisé pour asservir le système. Les procédés d'asservissement à partir du Jacobien sont bien connus en robotique [McK91].

Pour les expérimentations où les données expérimentales contiennent du bruit on calcule les valeurs des fonctions de sensibilité et on propage ces valeurs jusqu'aux sorties du modèle pour obtenir les marges d'erreur du modèle direct et du modèle différentiel direct.

Un cas typique d'asservissement. Le système et le modèle vont être mis à un même point initial e de l'hyperespace d'entrée, auquel correspond un point s de l'hyperespace de sortie du système. L'utilisateur du modèle choisit un point but s^* dans l'hyperespace de sortie. Pour trouver un point d'entrée tel que sa sortie calculée par le modèle corresponde au but s^* on suit l'algorithme suivant:

- 0. Calculer la sortie s' du modèle direct.
- 1. Si la distance $(s^* - s')$ est supérieure à une erreur acceptable donnée, continuer l'algorithme.
- 2. Le modèle différentiel calcule le Jacobien $J = \partial s'/\partial e$.
- 3. En utilisant une technique de pseudo-inversion [1] estimer $J^\dagger = \partial e/\partial s'$ $(J^\dagger = (J^T J)^{-1} J^T)$.
- 4. Estimer une commande $\Delta e = gJ^\dagger(s^* - s')$, où g est un scalaire positif appelé gain de l'asservissement.
- 5. Appliquer la commande Δe au point e **du modèle**.
- 6. Aller à l'étape 0.

[1]Dans notre logiciel nous utilisons la méthode Greville de pseudo-inversion

Inversion finale. Un point modèle s' qui correspond au point but s^* a été calculé. Cependant, la commande calculée par le modèle entraînera une erreur résiduelle après son exécution par le système $(s' - s)$ due à l'erreur d'estimation du modèle direct. L'approche finale à la position but est faite par le même algorithme mais elle est guidée par la sortie réelle du système (s) et non plus du modèle (s').

L'erreur finale de l'asservissement peut être arbitrairement réduit par l'itération convergente basée sur le Jacobien.

5.2 Un problème simulé de caractérisation idéale

Dans ce paragraphe nous présentons en détail les expérimentations faites sur un problème simulé: la modélisation d'un robot manipulateur à six dégrés de liberté.

La description du robot simulé se trouve dans le paragraphe 5.2.1.

Le robot simulé a été choisi pour les raisons suivantes:

- C'est un problème bien connu qui a un rapport direct avec les problèmes réels en robotique. Le problème de la cinématique illustre bien l'utilité et le besoin de la modélisation, particulièrement pour permettre l'asservissement du système.

- La complexité et les dimensions du problème sont considérables. Les exemples d'application des méthodes adaptatives ont normalement 3 ou 4 dimensions d'entrée, pour pouvoir garder la quantité de données et le temps d'exécution dans des limites raisonnables. Un des avantages de notre méthode est la possibilité d'attaquer les problèmes de dimensions élevées sans que l'effort de modélisation augmente de façon exponentielle.

- La nature du système permet une caractérisation idéale des fonctions de forme. Dans ce cas il est possible d'obtenir un modèle parfait du système, car les données expérimentales sont aussi idéales. Quand on dispose d'un modèle parfait il est possible d'analyser l'influence du bruit sur les données ou une caractérisation approximative des fonctions de forme ont sur les sorties du système.

Dans les paragraphes suivants nous allons décrire les expérimentations faites avec le robot simulé.

5.2.1 Description du robot simulé

Le robot simulé correspond au robot manipulateur industriel **Comercy**. C'est une *chaîne cinématique ouverte* (serial link manipulator).

Le but de notre modélisation est de résoudre le problème de la cinématique du manipulateur. Une solution particulièrement utile du problème de la cinématique est l'obtention, pour une position particulière du robot, d'une matrice de transformation qui va nous donner les coordonnées cartésiennes et l'orientation de la pince. Cette transformation, détermine la localisation (position et orientation) de la pince dans l'espace par rapport à la base du robot. Elle ne dit rien de la configuration du bras requise pour une localisation donnée de la pince [McK91].

Une matrice de transformation T est composée par:

$$T = \left[\begin{array}{c|c} rotation & translation \\ shearing & \\ local - scaling & \\ \hline 0 & 1 \end{array}\right] = \left[\begin{array}{ccc|c} 3 & & & 3 \\ & * & & * \\ & & 3 & 1 \\ \hline 1 & * & 3 & 1 * 1 \end{array}\right]$$

rotation, *shearing*, *local-scaling* et $translation^T$ sont quatre vecteurs à trois valeurs chacun [McK91].

Notre premier problème consiste donc à trouver un modèle de la **cinématique direct** du robot, c'est-à-dire, trouver un modèle de l'équation de transformation qui calculera la matrice de transformation pour toutes les positions du robot. L'équation de transformation nous donne la localisation de la pince à partir des coordonnées des articulations.

Le modèle direct aura 6 variables d'entrée (les angles des articulations du robot) et 12 variables de sortie (les valeurs de la matrice de transformation homogène données par le simulateur). Le tableau 5.1 montre les dépendances entre les variables d'entrée et de sortie. Les cases marquées indiquent pour chaque variable de sortie (colonnes) quelles sont les variables d'entrée (lignes) qui déterminent ses valeurs.

Notre deuxième problème consiste à asservir le robot, c'est-à-dire, pour toute localisation désirée de la pince dans l'espace cartésien on doit amener le robot à une configuration articulaire correspondante. Pour cela on doit calculer les déplacements articulaires nécessaires à partir des différences, dans l'espace cartésien, entre la localisation actuelle de la pince et la localisation désirée.

La localisation cartésienne de la pince est déterminée par six valeurs: trois angles de rotation et trois translations. A partir des six différences cartésiennes on

axe	**matrice de transformation**											
	m_{11}	m_{12}	m_{13}	m_{14}	m_{21}	m_{22}	m_{23}	m_{24}	m_{31}	m_{32}	m_{33}	m_{34}
θ_1	•	•	•	•	•	•	•	•				
θ_2	•	•	•	•	•	•	•	•	•	•	•	•
θ_3	•	•	•	•	•	•	•	•	•	•	•	•
θ_4	•	•	•		•	•	•		•	•	•	
θ_5	•	•	•		•	•	•		•	•	•	
θ_6	•	•			•	•			•	•		

Table 5.1 : Dépendances entre les variables du robot

peut facilement calculer les douze différences dans la matrice de transformation. Ce calcul ne dépend pas du modèle du robot. Nous utilisons pour ce calcul la transformation *row-pitch-yaw* [McK91].

En utilisant le modèle différentiel (le jacobien), obtenu en même temps que le modèle direct, et les différences des 12 valeurs qu'on veut réduire, on calcule les commandes articulaires.

5.2.2 Acquisition du modèle direct parfait

La première modélisation du robot manipulateur a été faite en utilisant la méthode d'identification structurelle présentée au paragraphe 3.3.

Toutes les fonctions de forme du robot peuvent être caractérisées de façon idéale par des sinusoïdes.

Les valeurs de référence pour les fonctions de forme, ainsi que les points de test sont choisis de façon aléatoire à chaque essai de modélisation.

L'algorithme d'identification structurelle a trouvé pour le modèle et tous ses sous modèles un niveau de récurrence égal à trois. L'acquisition complète du modèle demande un total de 740 fonctions de forme. Les variables peuvent être rajoutées au modèle dans n'importe quel ordre. Chaque fonction de forme nécessite trois points d'interpolation a priori, mais pour réduire le nombre de points d'interpolation nous les choisissons au croisement des fonctions de forme. Au total cette première modélisation demande un minimum de 729 points d'interpolation plus 48 points de test.

Les tests du modèle direct attestent son exactitude. L'erreur entre les valeurs des sorties calculées par le modèle et ceux données par le simulateur est nulle.

5.2.3 Modélisation avec transformation de sous modèles

Pour la deuxième modélisation du robot manipulateur nous avons utilisé encore une fois la méthode d'identification structurelle. Cependant nous avons rajouté à la procédure de modélisation la méthode de transformation de sous modèles, décrite dans 4.2. Le but de cette nouvelle modélisation était de montrer la réduction de la quantité de données nécessaires à la modélisation, ainsi que la réduction du nombre d'opérations du modèle qui la transformation de sous modèles apporte.

Pour cette nouvelle modélisation l'acquisition complète du modèle demande un total de 76 fonctions de forme. Pour l'acquisition des fonctions de forme il faut 117 points d'interpolation. Pour les tests de l'algorithme on utilise encore 48 points.

La distribution des fonctions de forme par variable d'entrée (tableau 5.2) montre bien la différence entre les deux modélisations. Dans le cas présent (méthode 2) l'augmentation de la dimension du problème entraîne une croissance linéaire du nombre de fonctions de forme à obtenir, alors que dans le cas précèdent (méthode 1) la croissance était beaucoup plus importante.

variable	méthode 1	méthode 2
0	243	9
1	297	17
2	135	15
3	45	15
4	15	15
5	5	5

Table 5.2 : Distribution des fonctions de forme par variable d'entrée

Dans le tableau 5.2 l'identification des variables indique seulement un ordre d'inclusion dans le modèle.

Comme pour le cas précèdent, le modèle acquis est parfait, c'est-à-dire que les erreurs du modèle sont nulles.

5.2.4 Asservissement du robot

Dans ce paragraphe nous allons montrer brièvement quelques résultats expérimentaux d'asservissement du robot simulé en utilisant le modèle différentiel direct généré par la méthode.

Expérience 1: petit écart initial

Dans cette expérience le robot est dans une position articulaire initiale donnée par $[2.\pi/3, 2.\pi/3, 2.\pi/3, 2.\pi/3, 2.\pi/3, 2.\pi/3]$. La pince est localisée dans l'espace cartésien à la position $[0.34, -0.60, -0.30]$ avec une orientation $[0.80, 1.34, -2.15]$. La matrice t de transformation est donnée par:

$$\begin{vmatrix} 0.15 & -0.16 & -0.97 & 0.34 \\ 0.16 & -0.96 & 0.18 & -0.60 \\ -0.97 & -0.18 & -0.12 & -0.30 \end{vmatrix}$$

On veut effectuer un déplacement de la pince défini par les différences de position $[0.1, 0.1, 0.1]$ et les différences d'orientation $[\pi/10, \pi/10, \pi/10]$. On calcule la matrice de transformation désirée t^*, qui va être donnée par:

$$\begin{vmatrix} -0.03 & -0.17 & -0.98 & 0.44 \\ -0.07 & -0.98 & -0.17 & -0.50 \\ -0.99 & -0.08 & -0.02 & -0.20 \end{vmatrix}$$

Un processus itératif basé sur les dérivées partielles va réduire progressivement les valeurs de $t^* - t$. La position articulaire trouvée qui nous donné la localisation désirée de la pince est donnée par $[2.30, 1.96, 2.40, 1.94, 2.41, 2.15]$.

Les figures 5.1, 5.2, 5.3 et 5.4 nous montrent l'évolution de l'erreur $t^* - t$.

On constate que deux pas du processus itératif de réduction de l'erreur $t^* - t$ ont suffi pour trouver une configuration adéquate pour le robot. Nous avons utilisé un gain d'asservissement égal à 1 parce que l'écart initial était petit.

Expérience 2: grand écart initial

Pour cette expérience le robot est dans la même position articulaire initiale donnée par $[2.\pi/3, 2.\pi/3, 2.\pi/3, 2.\pi/3, 2.\pi/3, 2.\pi/3]$.

On veut effectuer un déplacement de la pince défini par les différences de position $[0.7, 0.7, 0.7]$ et les différences d'orientation $[\pi/2, \pi/2, \pi/2]$. On calcule la matrice de transformation désirée t^*, qui va être donnée par:

$$\begin{vmatrix} 0.70 & -0.48 & -0.52 & 1.04 \\ -0.67 & -0.69 & -0.26 & 0.10 \\ -0.23 & 0.53 & -0.81 & 0.40 \end{vmatrix}$$

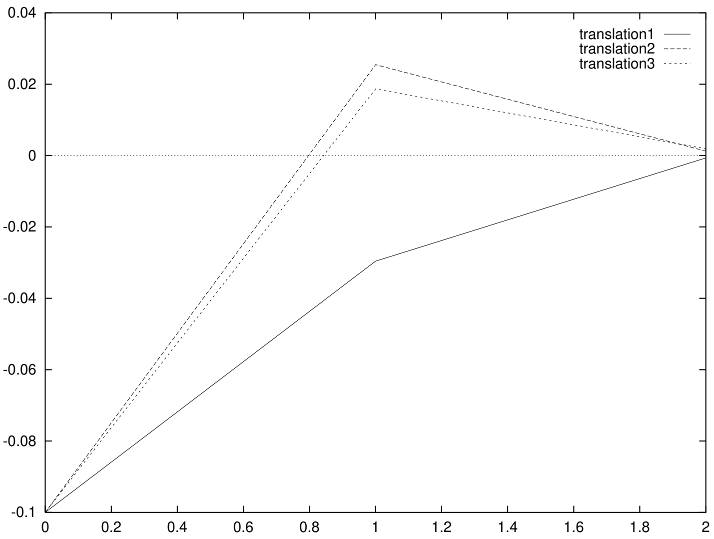

Figure 5.1 : Expérience 1: les composantes de l'erreur $t^* - t$ pour la translation

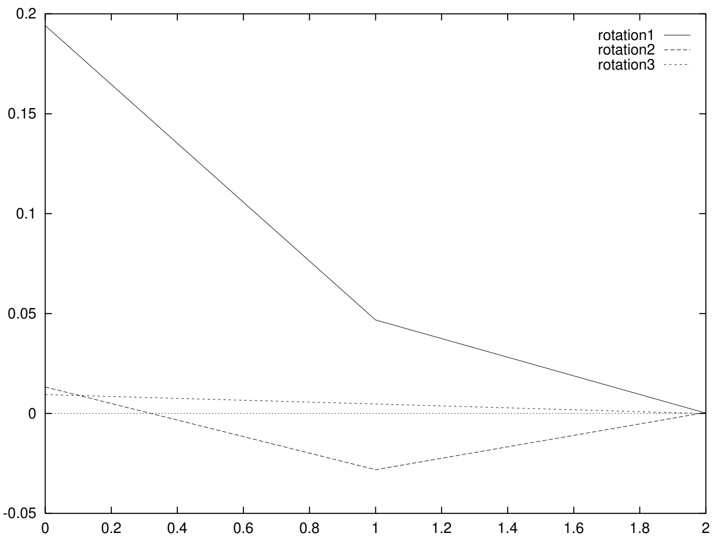

Figure 5.2 : Expérience 1: les composantes de l'erreur $t^* - t$ pour la rotation

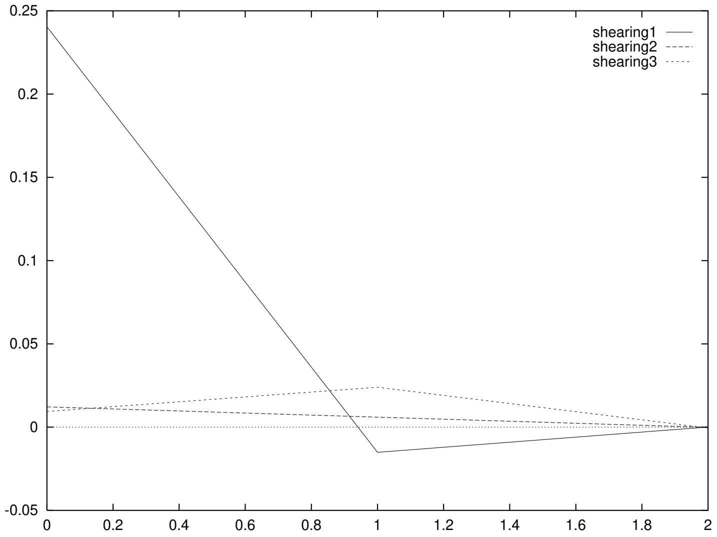

Figure 5.3 : Expérience 1: les composantes de l'erreur $t^* - t$ pour le 'shearing'

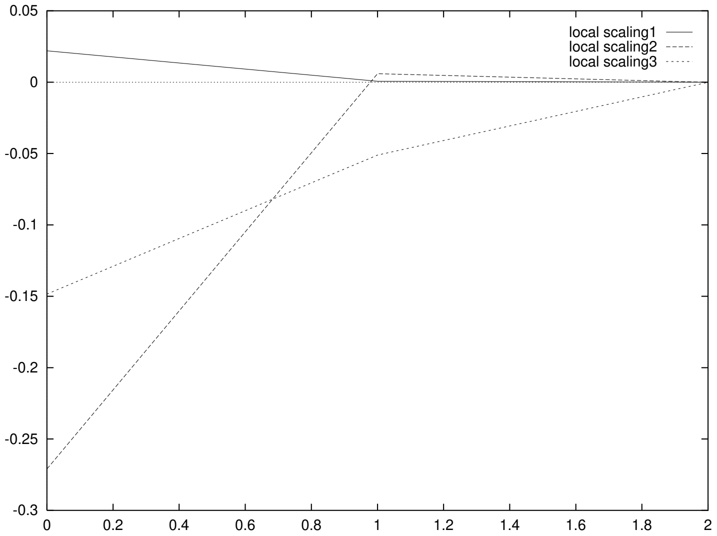

Figure 5.4 : Expérience 1: les composantes de l'erreur $t^* - t$ pour le 'local scaling'

Après le processus itératif, la position articulaire trouvé qui nous donné la localisation désirée de la pince est donnée par $[3.23, 2.85, 1.76, 2.89, 2.05, 3.87]$.

Les figures 5.5, 5.6, 5.7 et 5.8 nous montrent l'évolution de l'erreur $t^* - t$.

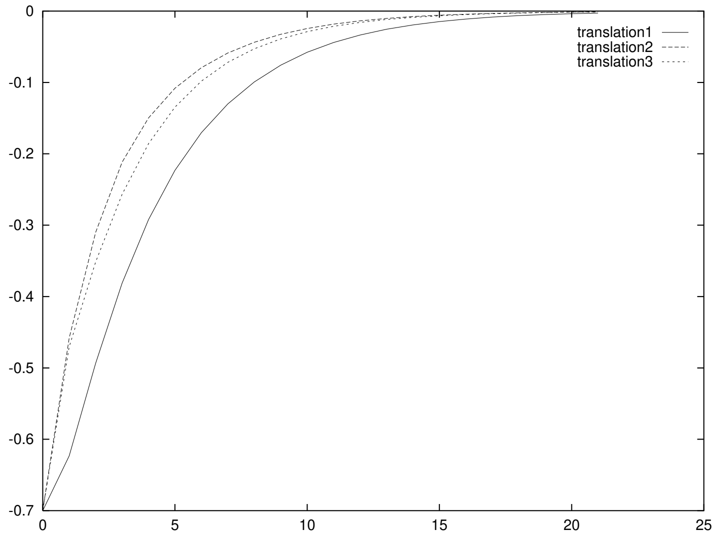

Figure 5.5 : Expérience 2: les composantes de l'erreur $t^* - t$ pour la translation

On constate que le processus itératif de réduction de l'erreur $t^* - t$ a bouclé 21 fois pour trouver une configuration adéquate pour le robot. Nous avons utilisé un gain d'asservissement égal à 0.25.

5.2.5 Modélisation à partir d'une caractérisation approximative

Dans ce paragraphe nous allons montrer les résultats expérimentaux obtenus en utilisant une caractérisation approximative des fonctions de forme. Le problème expérimental abordé ici est le même des paragraphes précédents, c'est-à-dire, le robot manipulateur simulé.

Nous connaissons les niveaux idéals de récurrence du modèle et des sous modèles, et nous savons aussi que les données expérimentales sont idéales. Cela nous permet d'analyser le comportement de la méthode face à une caractérisation non idéale. Nous pouvons mesurer l'effet de cette caractérisation et l'effet des variations possibles sur cette caractérisation.

Nous avons utilisé la caractérisation approximative en prenant des polynômes algébriques interpolateurs L'utilisation des polynômes algébriques comme outil de

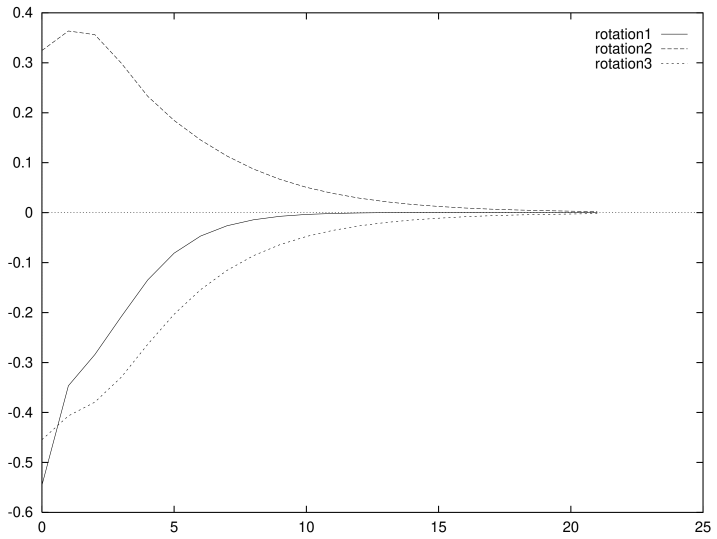

Figure 5.6 : Expérience 2: les composantes de l'erreur $t^* - t$ pour la rotation

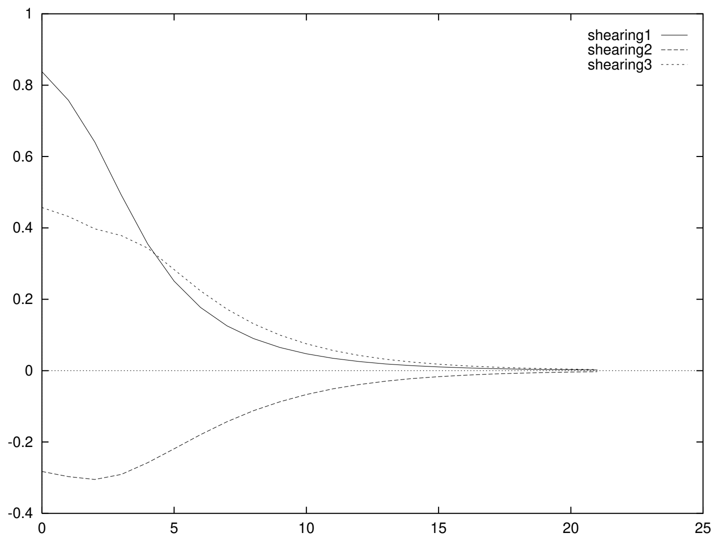

Figure 5.7 : Expérience 2: les composantes de l'erreur $t^* - t$ pour le 'shearing'

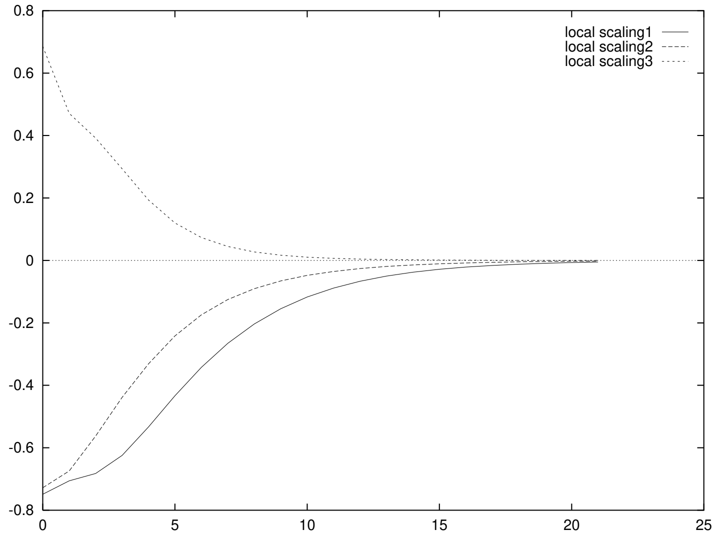

Figure 5.8 : Expérience 2: les composantes de l'erreur $t^* - t$ pour le 'local scaling'

représentation des fonctions de forme a été présentée dans le paragraphe 3.5.1.

A partir d'un ensemble de points d'interpolation nous estimons les coefficients du polynôme par la solution d'un système d'équations.

Le concepteur a le choix du dégré des polynômes utilisés. Les figures 5.9, 5.10, 5.11 et 5.12 montrent les courbes d'erreur d'estimation du modèle direct. Chaque figure montre les valeurs moyennes et maximales de l'erreur pour un sous ensemble des sorties du système. Chaque courbe a été obtenue en variant le dégrée des polynômes interpolateurs du modèle. A chaque essai toutes les fonctions de forme du modèle sont représentés par de polynômes le même dégré. L'axe horizontal indique le dégré des polynômes et l'axe vertical indique le pourcentage d'erreur d'estimation. Pour analyser chaque dégré de polynôme nous avons réalisé 500 tests.

On constate que les modèles directs sont assez précis quand nous utilisons les polynômes interpolateurs ayant de 5 à 8 degrés. A partir de 9 dégrés les modèles deviennent instables.

5.2.6 Modélisation à partir de données bruitées

Dans ce paragraphe nous allons montrer les résultats expérimentaux d'une série d'essais de modélisation utilisant les données bruitées. Le système modélisé est toujours le robot simulé.

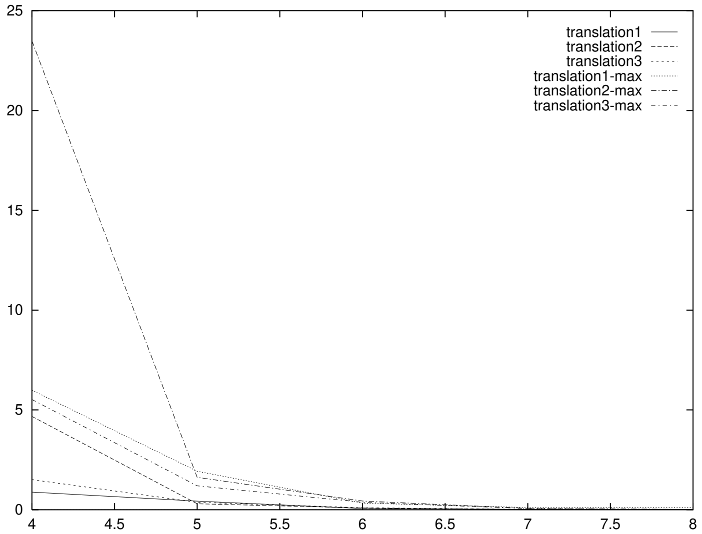

Figure 5.9 : Polynômes: les composantes de l'erreur du modèle pour la translation

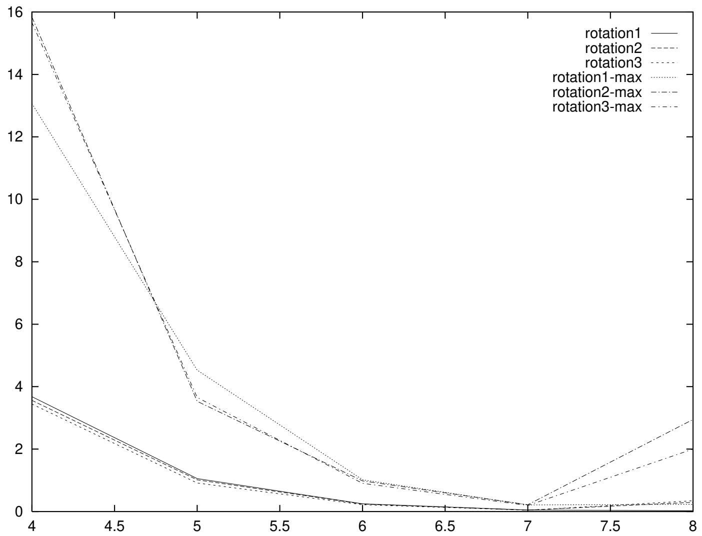

Figure 5.10 : Polynômes: les composantes de l'erreur du modèle pour la rotation

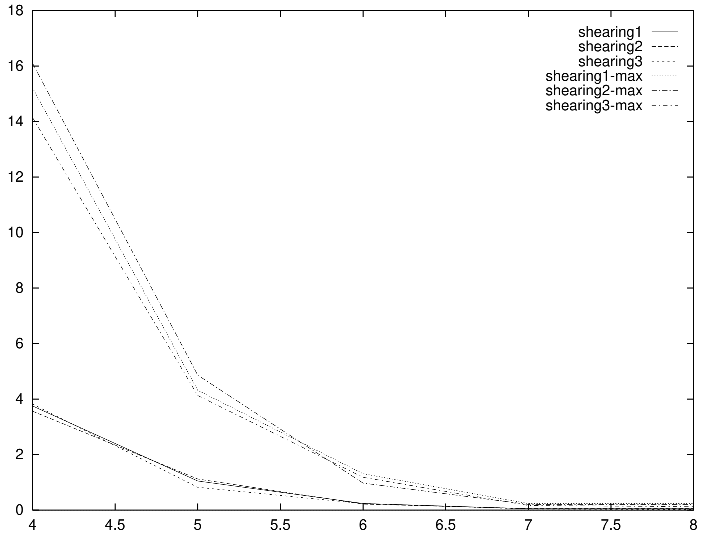

Figure 5.11 : Polynômes: les composantes de l'erreur du modèle pour le 'shearing'

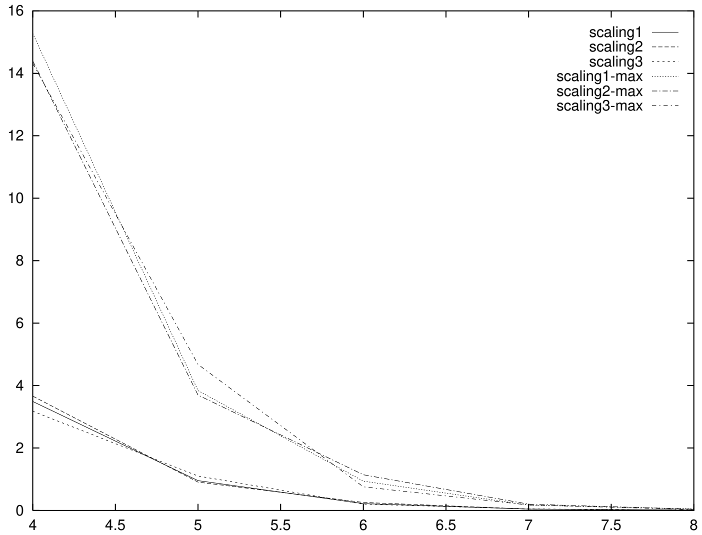

Figure 5.12 : Polynômes: les composantes de l'erreur du modèle pour le 'local scaling'

Les essais de modélisation ont montré que l'acquisition du modèle dépend fortement de la précision des données expérimentales, et que l'erreur du modèle direct croit très rapidement avec le bruit. L'adaptation du modèle par la méthode proposée en 4.4 est donc tout à fait nécessaire pour obtenir une bonne précision.

Dans les expérimentations de cette série nous avons rajouté aux données expérimentales un bruit gaussien déterminé par un écart type donné (et moyenne 0).

La figure 5.13 montre une estimation de l'erreur moyenne des modèles directs acquis à partir des données bruitées. L'axe horizontal représente l'écart type de l'erreur gaussian rajouté aux données. L'axe vertical représente le pourcentage d'erreur du modèle direct.

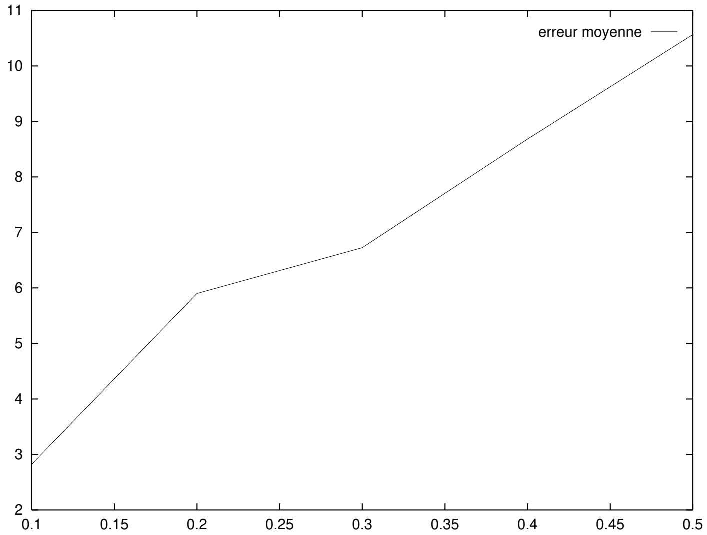

Figure 5.13 : L'erreur du modèle direct acquis à partir des données bruitées

L'erreur affichée est une moyenne de l'erreur des 12 sorties du modèle. Pour obtenir les valeurs affichées nous avons essayé 5 niveaux de bruit: 0.1%, 0.2%, 0.3%, 0.4% et 0.5% d'écart type. Pour chaque niveau de bruit (écart type) testé nous avons réalisé 5 essais de modélisation et pour chaque essai 500 tests du modèle direct. On constate la progression rapide de l'erreur du modèle direct même en présence d'un faible niveau de bruit.

En suite nous avons réalisé une série d'essais en utilisant la technique d'adaptation du modèle direct présentée dans 4.4. Pendant un essai de modélisation la technique d'adaptation est appliquée pour chacun des sous modèle acquis. L'adaptation d'un sous modèle est faite à l'aide d'un ensemble de nouveaux points de données aléatoirement choisis dans l'hyperespace du sous modèle. L'adaptation sert à corriger les paramètres des fonctions de forme rajoutées au modèle.

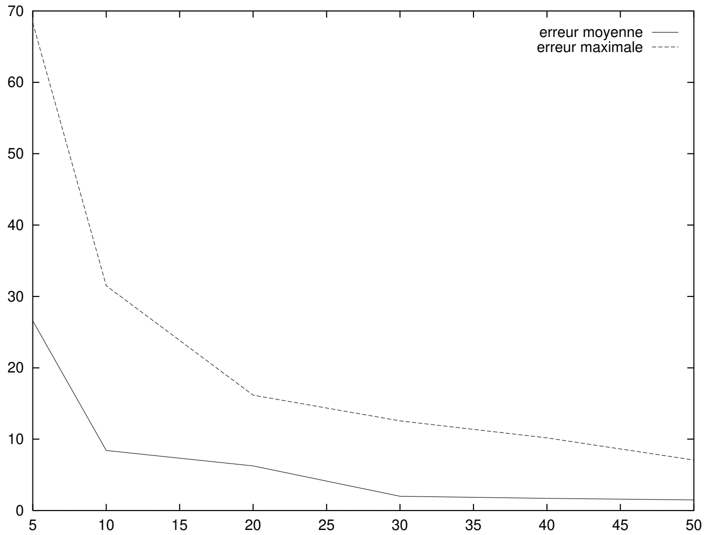

Figure 5.14 : L'erreur du modèle direct en fonction du nombre de points d'adaptation

La figure 5.14 montre l'erreur moyenne et maximale du modèle direct (axe vertical en pourcentage). Les mesures ont été obtenues en variant le nombre de points utilisés pour l'adaptation des sous modèles (axe horizontal). Le bruit rajouté aux données était de 2%.

La figure 5.15 montre l'erreur moyenne et maximale du modèle direct. Les mesures ont été obtenues en variant le bruit rajouté aux données et en gardant le nombre de points utilisés pour l'adaptation des sous modèles égal à 30.

Pour obtenir les valeurs affichées dans la figure 5.14 nous avons testé 6 configurations (5, 10, 20, 30 40 et 50 points d'adaptation), pour chaque configuration nous avons réalisé 500 tests d'un modèle direct acquis. Pour la figure 5.15 nous avons testé 6 niveaux de bruit: 0.5%, 1%, 2%, 3%, 4% et 5% de bruit rajouté aux données, et comme pour les cas precedents 500 tests d'un modèle direct. Pour les deux figures (5.14 et 5.15) l'erreur affichée est une moyenne de l'erreur des 4 sorties du modèle relatives à la translation du bout du bras. [2]

[2] Pour analyser l'erreur rajoutée par la modélisation il faut prendre en considération que les valeurs utilisées pour les tests des modèles directs dans ce paragraphe sont aussi bruitées du même écart type que celui rajouté pendant l'acquisition du modèle.

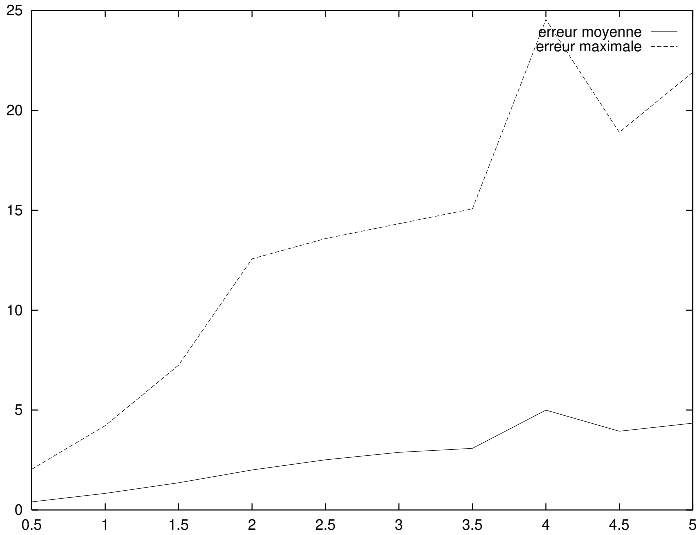

Figure 5.15 : L'erreur du modèle direct adapté en fonction du bruit rajouté aux données

5.3 Modélisation et asservissement d'un système intégré vision-robotique

Dans ce paragraphe nous présentons les expérimentations faites en appliquant la méthode d'identification structurelle à un problème réel en robotique: l'*asservissement visuel d'un robot manipulateur*.

Une cellule robotisée a deux composantes essentielles:

- **La perception**, qui permet de gérer les relations entre le robot et son environnement. Les organes de perception sont des capteurs dits *capteurs proprioceptifs* lorsqu'ils mesurent l'état interne du robot (positions et vitesse des articulations) et *extéroceptifs* lorsqu'ils recueillent des informations sur l'environnement (présence, mesure de distance, vision artificielle).

- **La commande**, qui synthétise les consignes des asservissements pilotant les actionneurs. A partir de la fonction de perception et des ordres de l'utilisateur, elle permet d'engendrer les mouvements du robot.

Parmi les types de capteurs *extéroceptifs* existants, les capteurs visuels sont les plus étudiés.

Il est très intéressant d'intégrer les informations fournies par les capteurs *extéroceptifs* dans les lois de commande du robot. Ainsi, les tâches peuvent s'exprimer directement dans l'espace du capteur et non dans l'espace articulaire. Les tâches sont spécifiées sous la forme d'un état dans une relation entre le robot et son environnement, et la régulation de cette relation permet d'obtenir la situation souhaitée du robot dans cet environnement.

L'asservissement sur des informations visuelles peut se mettre en œuvre de deux façons:

- la première est basée sur un *asservissement en position* de la caméra ou du robot par rapport à son environnement;
- la seconde, plus récente, est basée sur une régulation dans l'image (*asservissement visuel.*

L'*asservissement en position* consiste à positionner la caméra ou le robot par rapport à un objet. La situation courante vis à vis de l'objet est obtenue en interprétant les informations visuelles de l'objet qui sont extraites de l'image. Cette interprétation nécessite une reconstruction 3-D de la scène, faite, par exemple, par une paire de caméras stéréo. L'interprétation de la scène dépend de l'extraction de primitives dans l'image, de la modélisation de la caméra et de la modélisation du robot. Le déplacement de la caméra (ou du robot) est ensuite assuré par un asservissement classique en position dans l'espace cartésien. L'avantage de ce type d'asservissement est qu'il n'exige pas de connaissances géométriques préalables de la scène (positions des objets par rapport au robot et à la caméra).

Avec l'*asservissement visuel*, on cherche à atteindre un motif dans l'image et non plus à contrôler une situation entre la caméra et un objet. Ainsi on supprime l'interprétation de la scène, et on supprime du même coup les erreurs de cette interprétation [Dor95]. Une seule caméra est nécessaire. En contre partie, il faut avoir certaines connaissances préalables à propos de la géométrie de la scène.

Pour les deux types d'asservissement, la nature de la tâche robotique impose, soit l'utilisation d'une caméra embarquée sur le robot, soit l'utilisation d'une caméra indépendante de ce robot.

Pour illustrer l'application de notre méthode nous avons repris le système développé au LIFIA [3] par l'équipe MOVI [HDBE95] [HDBL95] [Dor95].

Dans les paragraphes suivants nous allons présenter ces expérimentations. Nous présentons d'abord dans le paragraphe 5.3.1 les détails du problème auquel le système s'attaque et quelques concepts importants à propos de l'asservissement visuel. Ensuite, dans les paragraphes 5.3.2 et 5.3.3 nous présentons l'application de notre méthode au problème.

[3] Laboratoire d'Informatique Fondamentale et Intelligence Artificielle, actuellement Laboratoire Leibniz, de l'institut IMAG, à Grenoble

5.3.1 Description du problème

Dans ce paragraphe nous décrivons plus précisément le problème qu'on se pose en réalisant les expérimentations d'asservissement visuel.

Les Jacobiens de l'asservissement visuel

Dans ce paragraphe nous présentons quelques concepts importants relatifs à l'asservissement visuel. Une notion importante de l'asservissement visuel est le Jacobien d'image. Cette entité quantifie la relation différentielle entre les variations visuelles fournies par un ou plusieurs capteurs et les mouvements de la scène engendrant ces variations.

Si on considère une caméra indépendante du robot. Le robot se trouve dans le champ visuel de cette caméra. Les déplacements de la pince du robot ou des objets embarqués sur cette pince sont réalisés à l'aide des différents axes constituant le robot. La situation de la pince ne dépend que de la valeur des coordonnées articulaires q. Si s est un vecteur d'information visuelle (également appelée signal capteur), alors on peut écrire:

$$s = s(q, t) \tag{5.1}$$

où le paramètre t représente la contribution du mouvement de la caméra si elle même est mobile.

La différentielle de s permet de relier les variations visuelles dans l'image aux mouvements de la caméra et de la scène. A partir de l'équation 5.1, on obtient:

$$\dot{s} = \frac{\partial s}{\partial q}\dot{q} + \frac{\partial s}{\partial t} \tag{5.2}$$

Le terme $\frac{\partial s}{\partial q}$ peut se décomposer sous la forme:

$$\frac{\partial s}{\partial q} = \frac{\partial s}{\partial r}\frac{\partial r}{\partial q} \tag{5.3}$$

où on reconnaît le classique Jacobien du robot $\frac{\partial r}{\partial q}$ si l'on choisit pour r la situation de la pince. Le terme $\frac{\partial s}{\partial r}$ ne dépend que de la tâche choisie, représentée par

s. On appelle ce terme le Jacobien de la tâche.

Une autre formulation de la différentielle de s est possible en utilisant les torseurs cinématiques.

$$\dot{s} = J\mathbf{T} \tag{5.4}$$

où :

- $\mathbf{T}$ est le torseur cinématique représentant la vitesse relative de la scène par rapport à la caméra.
- J est le Jacobien d'image ou matrice d'interaction qui représente un ensemble $\{H_1...H_n\}$ associé à $s = (s_1...s_n)^T$. H_j est un *torseur d'interaction* dont l'expression dépend à la fois des caractéristiques de l'environnement et du capteur.

Le Jacobien d'image dans cette formulation à partir des torseurs équivaut au Jacobien de la tâche dans la formulation matricielle initiale.

Le système d'asservissement visuel existant

Le système d'asservissement visuelle de l'équipe MOVI propose une méthode pour aligner la pince d'un robot - ou n'importe quel autre outil - avec un objet. Cette alignement est nécessaire dans certains types de problèmes, comme la saisie d'objets (grasping).

La méthode de l'équipe MOVI a été appliquée à un problème expérimental consistant à guider la saisie d'un objet par un robot avec une caméra. La caméra est indépendante du robot, qui se trouve dans le champ visuel de cette caméra.

Le robot en question est un manipulateur industriel SCEMI à six dégrés de liberté.

La tâche consiste à :

- calculer une condition d'alignement entre la pince du robot et l'objet à saisir,
- asservir le robot pour qu'il bouge et atteigne la position désirée.

La caractérisation de l'alignement de deux objets (l'objet et la pince) nécessite une représentation de la relation géométrique existant entre ces deux objets. A partir des modèles géométriques de l'objet et de la pince, et d'un certain nombre de points de l'objet identifiés dans l'image d'une caméra, le système calcule la position (les coordonnées images) qu'un certain nombre de points de la pince doivent avoir dans cette même image.

L'ensemble des coordonnées images des points de la pince définissent le motif qu'on veut retrouver dans l'image après l'asservissement du robot. La figure 5.16 montre l'exemple d'un motif retrouvé dans l'image d'un cube.

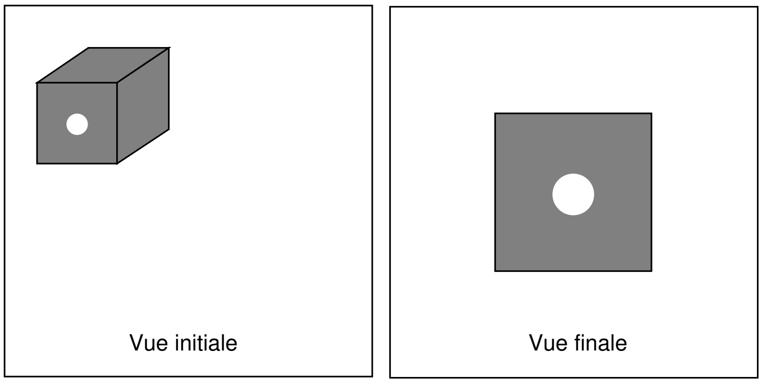

Figure 5.16 : Exemple des vues initiale et désirée d'un cube.

L'algorithme de l'équipe MOVI d'asservissement visuel est le suivant:

- 1. Prendre une image de la pince du robot.
- 2. Détecter les points image correspondants aux points spécifiés de la pince.
- 3. Apparier ces points avec leurs correspondants dans le motif. Si la position courante des points est assez proche de leur position finale alors arrêter. Sinon aller à l'étape suivante.
- 4. Calculer la pose de la pince par rapport à la camera.
- 5. Calculer la matrice J (Jacobien d'image) ainsi que sa pseudo-inverse.
- 6. Calculer le torseur cinématique de la pince et imprimer ce torseur à la pince.
- 7. Aller à l'étape 1.

La matrice J a pour dimension $2n \times 6$ (où n est le nombre de points spécifiés de la pince) et dépend des paramètres suivants:

- les facteurs d'échelle vertical et horizontal associés à l'ensemble caméra / convertisseur analogique numérique;

- les coordonnées des points de la pince exprimées dans le repère caméra. Ces coordonnées varient avec le temps puisque la pince se déplace;
- la rotation et la translation entre le repère pince et le repère caméra. Ces valeurs varient aussi avec le temps.

Le calcul de la pose entre la pince et la caméra est effectué grâce à un algorithme de calcul par approximations successives avec un modèle de projection para-perspectif de la pince.

Le torseur cinématique calculé par la méthode va être utilisé comme commande du robot.

Le système expérimental

Nous disposons d'un système expérimental composé de deux robots manipulateurs SCEMI à six dégrées de liberté (voir figure 5.17). Un robot effectue les tâches d'asservissement et l'autre maintient une ou plusieurs caméras CCD. L'utilisation d'une caméra embarquée sur le bras d'un robot permet de positionner (d'une manière automatique ou pas) la caméra en un endroit optimal: la pince du robot et les objets d'intérêt doivent être dans le champ visuel de la caméra.

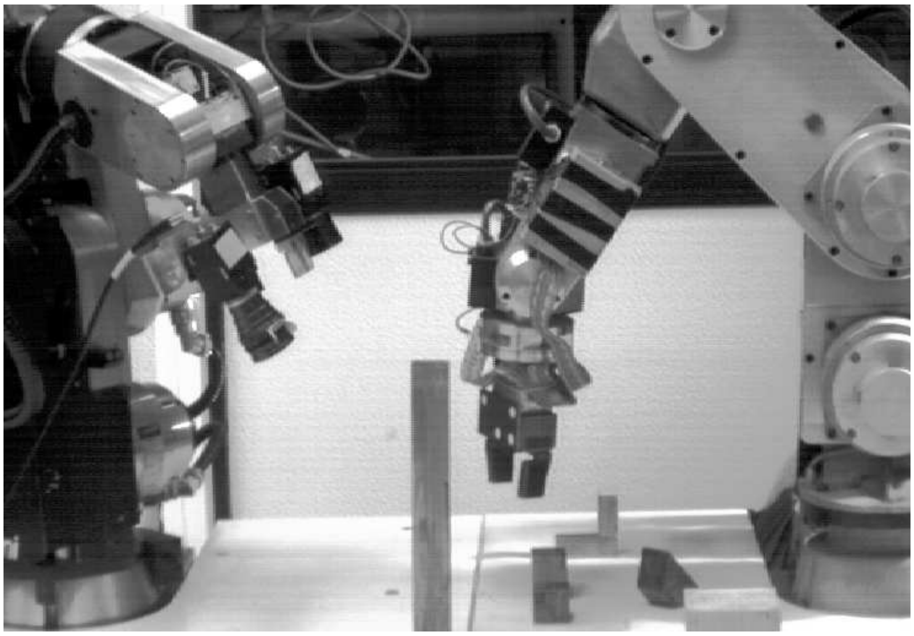

Figure 5.17 : Site expérimental.

L'asservissement visuel nécessite l'utilisation des primitives 3D liées à la pince, constituées dans notre cas de quatre cibles blanches imprimées sur une plaque noir qui est collée à la pince du robot (figure 5.18). Ainsi, les primitives géométriques sont les points 3D constitués par les centres de ces quatre cibles circulaires.

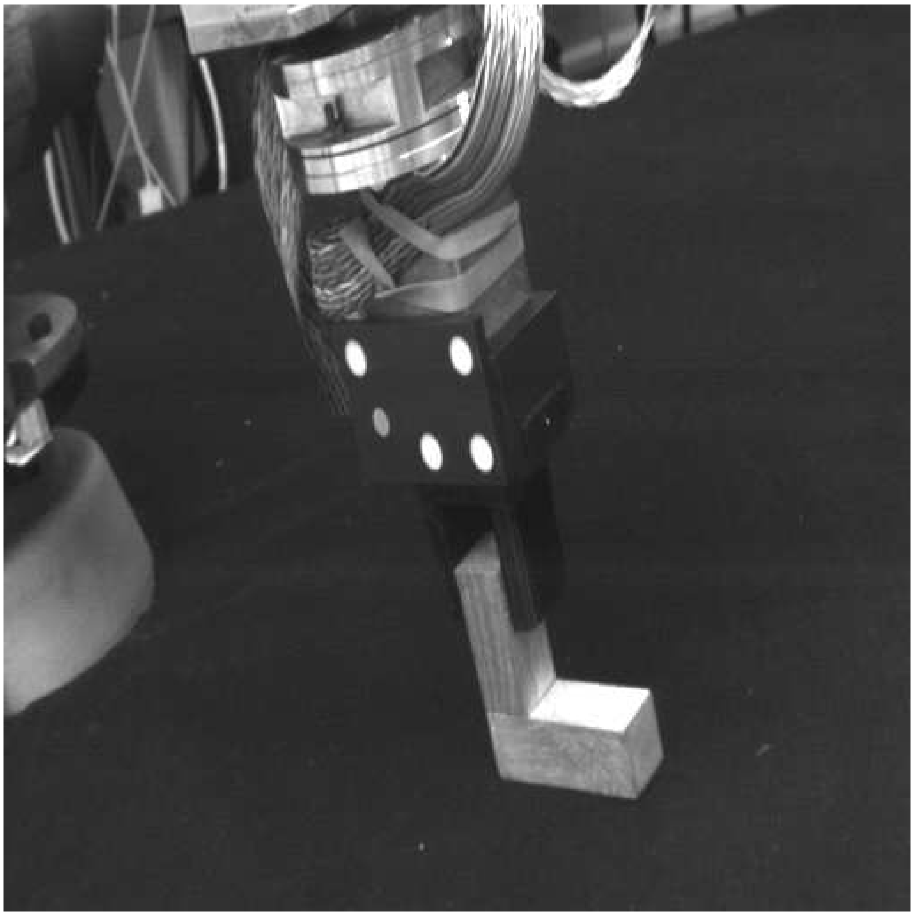

Figure 5.18 : La pince du robot et les cibles blanches.

Les coordonnées 2D des centres de gravité des quatre tâches claires dans l'image constitue notre vecteur position s, et le but de l'asservissement sera retrouver un motif dans l'image déterminé par les nouveaux centres des cibles.

En fait, n'importe quel autre type de marque aurait pu être utilisé pour la localisation de la pince dans l'image, mais en pratique les cibles blanches simplifient le traitement de bas niveau de l'image.

L'extraction des centres de gravité a une précision sub-pixel (de l'ordre du dixième de pixel). Pour diminuer le temps de traitement une technique de fenêtrage est utilisée qui consiste à traiter uniquement les parties utiles de l'image, c'est-à-dire, les parties qui encadrent les primitives 2D.

5.3.2 Notre modélisation du problème

Nous avons repris le même système expérimental utilisé par l'équipe MOVI. Nous avons aussi réutilisé leurs logiciels de traitement de l'image (étapes 1 et 2 de l'algorithme).

Du problème initial: saisie d'objets, nous ne retenons que la deuxième partie : l'asservissement du robot pour qu'il bouge et atteigne une position désirée. Nous ne faisons pas le calcul de la condition d'alignement entre la pince du robot et l'objet à saisir.

Le tableau 5.3 résume les différences essentielles entre notre approche du problème d'asservissement visuel et l'approche du système existant.

Méthode MOVI	Notre méthode
Caractérisation: de la pince (morphologie de la pince du robot), du système de vision (facteurs d'échelle).	Caractérisation: forme paramétrique des fonctions de forme (sinusoïdes).
Pas de modèle direct de l'ensemble	Acquisition d'un modèle direct de l'ensemble (entrées: articulaire; sorties: position des cibles dans l'image).
Acquisition et traitement d'une image à chaque boucle d'asservissement.	Boucle d'asservissement basée sur le modèle direct. Besoin d'information visuelle seulement pour l'approche finale au but.
Calcul de la pose de la pince par rapport à la camera à chaque boucle d'asservissement.	Pas de calcul de pose.
Calcul du Jacobien de l'image.	Calcul du Jacobien intégral.
Commande du robot par torseur cinématique.	Commande du robot dans l'espace articulaire.

Table 5.3 : Différences essentielles entre les deux approches

Dans les paragraphes suivantes nous allons détailler notre solution au problème d'asservissement visuel et montrer les résultats expérimentaux.

5.3.3 Acquisition du modèle direct

Le problème de l'asservissement visuel n'est pas élémentaire. Le système expérimental est déterminé par deux sous systèmes complexes (visuel et robotique), et son intégration dépend d'un ensemble important de paramètres qui peuvent changer son comportement. La caractérisation d'un modèle complet par une méthode classique exigerait un grand effort de son concepteur. Une méthode adaptative aurait aussi beaucoup de difficulté pour acquérir un bon modèle vu le nombre de dimensions du problème.

La solution de l'équipe MOVI réduit la complexité du problème en travaillant à partir de la localisation relative non-euclidienne entre les objets (pince et objets à saisir) et la caméra. En utilisant un modèle géométrique de l'objet le système estime sa position relative, et peut alors estimer le Jacobien d'image, qui va permettre le calcul d'un mouvement relatif désirée de la pince pour qu'un motif dans l'image soit trouvé. Dans le cas particulier du bras robot ce mouvement relatif de la pince peut être utilisé comme commande pour l'asservissement du robot. Cela oblige qu'un modèle complet du robot soit connu auparavant. La précision du positionnement final du robot dans son environnement ne dépend ni du modèle du robot ni du modèle de la caméra jouant le rôle du capteur. Seule l'extraction des primitives 2D doit être précise.

On constate que c'est une solution bien adaptée au type de tâche auquel le système s'attaque.

Nous voulons voir le problème d'une autre façon, moins contrainte par le système expérimental en question. Notre intérêt est l'acquisition de modèles dans des conditions où on ne dispose pas d'autant de connaissances préalables à propos du système à modéliser. Les seules données que nous allons utiliser seront les paires entrée-sortie les plus élémentaires, c'est-à-dire, positions articulaires du robot et les positions des centres de gravité des cibles dans l'image de la caméra. Nous ne connaissons pas les modèles géométriques des objets, les facteurs d'échelles du système visuel, le modèle du robot, les repères de la caméra ou du robot, etc. L'asservissement sera aussi fait dans l'espace élémentaire de commande du robot, l'espace articulaire, et nous allons comander directement chaque dégré de liberté.

Nous allons suivre les étapes d'expérimentation décrites dans le paragraphe 5.1 en montrant leurs détails.

Caractérisation

Caractérisation initiale. Le système expérimental contient six variables d'entrées, les six dégrés de liberté du robot. Il contient aussi huit variables de sortie, qui sont les coordonnées 2D des centres de gravité des quatre cibles blanches de la pince sur l'image de la caméra.

Caractérisation des fonctions de forme. Les fonctions de forme vont être représentées par de sinusoïdes (voir 3.5.2). La nature des relations entrée-sortie (rotations des articulations projetées sur l'image 2D) indique cette caractérisation. Dans la figure 5.19 nous montrons un exemple d'une fonction de forme pour la variation des valeurs de x (coordonée horizontale sur l'écran) en fonction du mouvement du deuxième degrée de liberté du robot, avec un ensemble de points de mesure et la représentation choisie.

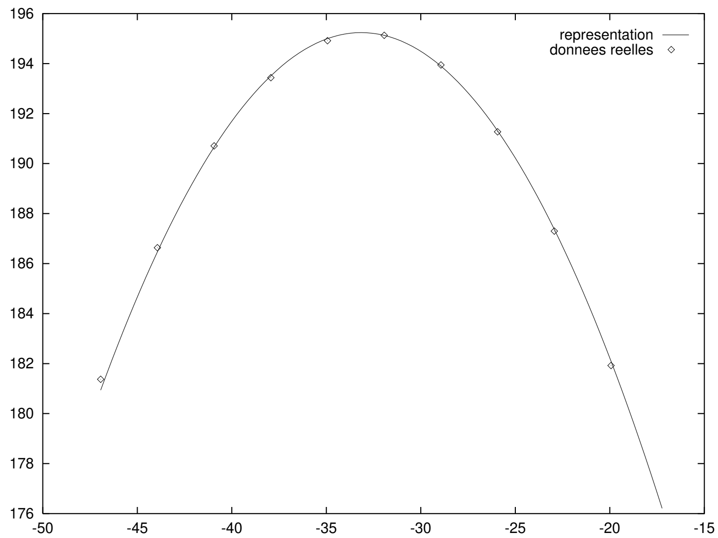

Figure 5.19 : Fonction de forme du deuxième axe du robot

Identification structurelle

Acquisition des fonctions de forme. Il faut au moins trois points de mesure pour faire l'identification des paramétres de chaque fonction de forme. Une acquisition du modèle utilisant les transformation de sous modèles demande un total de 48 fonctions de forme et un total de 81 points d'interpolation. Le nombre de points de mesure ne correspond pas à trois fois le nombre de fonctions de forme car chaque point de mesure peut servir à l'identification de plus d'une fonction de forme.

Test du niveau de récurrence. Nous faisons une séquence de trois tests dans des positions aléatoirement choisies dans l'hyperespace (ou sous espace) d'entrée pour déterminer le niveau de récurrence acceptable. Nous avons choisi un niveau de récurrence égal à trois pour le modèle et tous les sous-modèles. Au total il faut 20 points de tests.

Mise au point du modèle. La correction des paramètres du modèle est faite en utilisant 15 points d'adaptation à chaque sous modèle acquis.

5.3.4 Utilisation du modèle

Nous avons testé le modèle acquis par la méthode. Le paragraphe 5.3.4 présente une analyse du modèle direct suivie des résultats expérimentaux de l'utilisation du modèle direct et le paragraphe 5.3.4 les résultats expérimentaux de l'utilisation du modèle différentiel.

L'utilisation du modèle direct

L'architecture du système expérimental a la particularité de restreindre l'espace de configuration du robot. Parmi les positions possibles de l'espace de configuration du robot, seulement un volume réduit et irrégulier est utile. Les position utiles sont celles pour lesquelles la pince du robot apparaît dans l'image obtenue par la caméra. Comme conséquence, la plupart des fonctions de forme sont en pratique des morceaux de sinusoïdes, comme on le voit dans la figure 5.19.

A cause de la forme irrégulière de l'espace de configuration utile il n'a pas été possible de choisir les positions idéales pendant l'acquisition des fonctions de forme (voir 3.5.1). Une acquisition soucieuse des données a permis une bonne précision des points. Cependant, les positions non idéales des points d'interpolation ont entraîné des prévisions moins précises aux bords de l'image qu'au centre.

L'écran du moniteur qui nous donne l'image du bras robot contient 512x512 pixels. Nous avons réalisé 120 tests du modèle direct dans des positions aléatoirement choisies parmi les positions utiles de l'espace de configuration et nous avons obtenue une erreur moyenne de 4.95 pixels, c'est-à-dire 0.967%. Dans la portion centrale de l'écran l'erreur moyenne était de 2.01 pixels, c'est-à-dire 0.39%.

L'asservissement du système

Nous avons réalisé une série réussie d'essais d'inversion du système. La précision du modèle direct permet une réduction presque complète de l'écart jusqu'au motif désiré dans l'image (position de la pince) sans l'utilisation de la caméra. La réduction de l'écart résiduel est faite ensuite à l'aide des informations visuelles.

La figure 5.20 nous montre un cas typique d'asservissement visuel. la pince est schématisée par les positions des quatre tâches sur l'écran. La figure a) montre la position de départ (au centre de l'écran) et la position but (en haut). La figure b) montre le mouvement des cibles vers le motif désiré. La figure 5.21 montre l'évolution des positions articulaires pendant l'asservissement.

Pour cet exemple nous avons utilisé un gain d'asservissement égal 0.2.

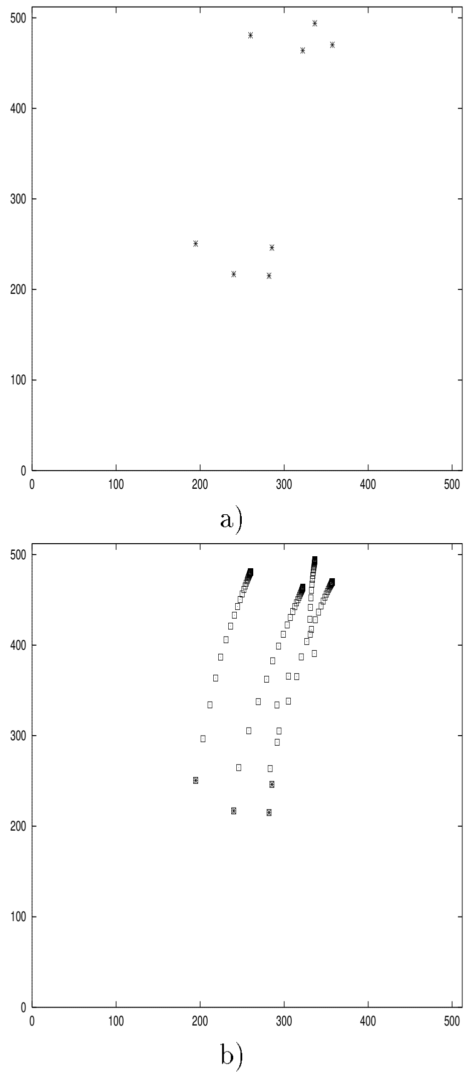

Figure 5.20 : Exemple d'asservissement visuel: mouvement des cibles

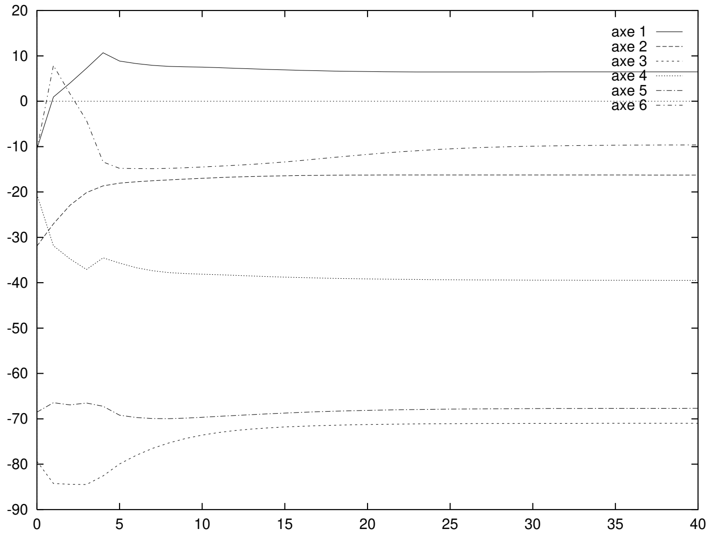

Figure 5.21 : Exemple d'asservissement visuel: mouvement des axes

Chapitre 6

Conclusion

Nous avons présenté une méthode de construction de modèles basée sur un algorithme original. Cette méthode se distingue de l'identification de paramètres classique dans la mesure où elle peut être utilisée sans forte connaissance préalable sur le système à modéliser. Elle se distingue par ailleurs des approches de type "boîte noire" dans la mesure où elle peut s'attaquer à des problèmes de modélisation faisant référence à des espaces d'entrée de grande dimension. Les modèles obtenus se présentent sous la forme de fonctions calculables et sont "manipulables" en tant que telles. Par exemple, les modèles peuvent être dérivés ou utilisés comme structure de départ lors d'une nouvelle modélisation.

Nous avons montré comment ils peuvent être rendu robustes en ne considérant que des sous-modèles et en appliquant des techniques de régression classiques.

Nous avons montré que la méthode peut, à partir d'un nombre restreint de points, acquérir des modèles dans des espaces de dimensions raisonnables et que ces modèles peuvent être utilisés en pratique avec une bonne précision. Nous ne connaissons pas de nos jours de méthodes d'approximation de type "boîte noire" capables de bien modéliser sur de dimensions équivalentes.

L'originalité de la méthode a aussi son prix. Sur le plan théorique il nous a semblé avoir manqué des outils mathématiques adaptés à la formulation et à l'analyse des mécanismes récurrents que la transformation DP engendre. Bien que l'explication que nous en avons donnée les justifie totalement, les outils que nous avons utilisés ne nous permettent pas de conclure sur un certain nombre de points. Par exemple, nous ne savons pas décider si il est possibile ou non de reconstruire les fonctions de base qui servent à la définition de l'ensemble $\mathcal{F}_n$. Nous avons aussi l'intuition d'avoir à faire à une classe beaucoup plus générale d'outils que ceux que nous avons présentés. N'existe-t-il pas d'autres hypothèses sur f qui conduisent à d'autres transformations que la transformation DP ?

Notre étude nous a conduit sur de nombreuses pistes que nous n'avons pas exposées ici faute de justifications mathématiques rigoureuses. Par exemple, il existe probablement une façon plus astucieuse de "rentrer" les variables dans le processus d'identification. En effet rien ne s'oppose en pratique à adopter une technique du type "divide and conquer" qui séparerait les variables en groupes avant de réaliser une synthèse du modèle. Il nous paraît aussi dispencieux de vouloir construire des modèles hors de leur contexte d'utilisation. En effet les modèles sont en général utilisés sur des sous variétés de dimension réduite de l'espace d'entrée. Pourquoi ne pas tenir compte de cette connaissance préalable ?

Finalement nos voudrions lever le voile sur cette thèse qui nous semble d'un point de vue épistémologique tout à fait conforme à ce qui se passe en général. La transformation *DP*, comme l'algorithme, ont d'abord été mis au point empiriquement, avec l'aide d'un système de calcul formel et d'un système de visualisation de surfaces 3D. La justification mathématique et algorithmique donnée dans ce document ne peut justement remplacer ce qui est par ailleurs si précieux au modélisateur : l'**intuition**.

Annexes

Annexe A

Application de la méthode au bras planaire

Dans cette annexe nous présentons l'application de la méthode d'identification structurelle à l'illustration du bras planaire de la section 3.2. Cette presentation est un fichier généré par Maple, un logiciel mathématique.

```
> f:=proc(x,y);
> RETURN(cos(x)+cos(x)*cos(y)-sin(x)*sin(y));
>  end;
```

```
f := proc(x,y) RETURN(cos(x)+cos(x)*cos(y)-sin(x)*sin(y)) end
```

```
> g:=proc(x,y,x0,y0);
> RETURN(f(x0,y)*f(x,y0)/f(x0,y0));
> end;
```

```
g := proc(x,y,x0,y0) RETURN(f(x0,y)*f(x,y0)/f(x0,y0)) end
```

```
> h:=((f(x,y1)-g(x,y1,x0,y0))*(f(x1,y)-g(x1,y,x0,y0)))/ (f(x1,y1)
> -g(x1,y1,x0,y0));
```

$$
\begin{aligned}
h := {} & (\cos(x) + \cos(x)\cos(y1) - \sin(x)\sin(y1) - \\
& (\cos(x0) + \cos(x0)\cos(y1) - \sin(x0)\sin(y1)) \\
& (\cos(x) + \cos(x)\cos(y0) - \sin(x)\sin(y0)) \Big/ (\%1))(\cos(x1) \\
& + \cos(x1)\cos(y) - \sin(x1)\sin(y) - \\
& (\cos(x0) + \cos(x0)\cos(y) - \sin(x0)\sin(y)) \\
& (\cos(x1) + \cos(x1)\cos(y0) - \sin(x1)\sin(y0)) \Big/ (\%1)) \Big/ (\cos(x1) \\
& + \cos(x1)\cos(y1) - \sin(x1)\sin(y1) - \\
& (\cos(x0) + \cos(x0)\cos(y1) - \sin(x0)\sin(y1))
\end{aligned}
$$

$$(\cos(x1) + \cos(x1)\cos(y0) - \sin(x1)\sin(y0)) \Big/ (\%1))$$
$$\%1 := \cos(x0) + \cos(x0)\cos(y0) - \sin(x0)\sin(y0)$$

```
> i:=g(x,y,x0,y0)+h;
```

$$\begin{aligned}
i := {}& (\cos(x0) + \cos(x0)\cos(y) - \sin(x0)\sin(y)) \\
& (\cos(x) + \cos(x)\cos(y0) - \sin(x)\sin(y0)) \Big/ (\%1) + (\cos(x) \\
& + \cos(x)\cos(y1) - \sin(x)\sin(y1) - \\
& (\cos(x0) + \cos(x0)\cos(y1) - \sin(x0)\sin(y1)) \\
& (\cos(x) + \cos(x)\cos(y0) - \sin(x)\sin(y0)) \Big/ (\%1))(\cos(x1) \\
& + \cos(x1)\cos(y) - \sin(x1)\sin(y) - \\
& (\cos(x0) + \cos(x0)\cos(y) - \sin(x0)\sin(y)) \\
& (\cos(x1) + \cos(x1)\cos(y0) - \sin(x1)\sin(y0)) \Big/ (\%1)) \Big/ (\cos(x1) \\
& + \cos(x1)\cos(y1) - \sin(x1)\sin(y1) - \\
& (\cos(x0) + \cos(x0)\cos(y1) - \sin(x0)\sin(y1)) \\
& (\cos(x1) + \cos(x1)\cos(y0) - \sin(x1)\sin(y0)) \Big/ (\%1)) \\
& \%1 := \cos(x0) + \cos(x0)\cos(y0) - \sin(x0)\sin(y0)
\end{aligned}$$

```
> simplify(i);
```

$$\cos(x) + \cos(y)\cos(x) - \sin(x)\sin(y)$$

Annexe B

Application de la méthode à un polynôme à trois variables et deux termes

Dans cette annexe nous présentons l'application de la méthode d'identification structurelle à un polynôme à trois variables d'entrée et deux termes. Comme pour les annexes précédentes cette présentation est un fichier généré par Maple.

```
> f:=proc(x,y,z);
> RETURN(f1(x)*f2(y)*f3(z)+f4(x)*f5(y)*f6(z));
> end;
f := proc(x,y,z) RETURN(f1(x)*f2(y)*f3(z)+f4(x)*f5(y)*f6(z)) end
```

```
> g:=proc(x,y,x0,y0,z0);
> RETURN(f(x,y0,z0)*f(x0,y,z0)/f(x0,y0,z0));
> end;
g := proc(x,y,x0,y0,z0) RETURN(f(x0,y,z0)*f(x,y0,z0)/f(x0,y0,z0)) end
```

```
> h:=proc(x,y,x0,y0,z0,x1,y1);
> RETURN((f(x1,y,z0)-g(x1,y,x0,y0,z0))*(f(x,y1,z0)-g(x,y1,x0,y0,z0))/
> (f(x1,y1,z0)-g(x1,y1,x0,y0,z0)));
> end;
h := proc(x,y,x0,y0,z0,x1,y1)
        RETURN((f(x1,y,z0)-g(x1,y,x0,y0,z0))*(f(x,y1,z0)-g(x,y1,x0,y0,z0))/
            (f(x1,y1,z0)-g(x1,y1,x0,y0,z0)))
     end
```

```
> i:=proc(x,y,x0,y0,z0,x1,y1);
> RETURN(g(x,y,x0,y0,z0)+h(x,y,x0,y0,z0,x1,y1));
> end;

i := proc(x,y,x0,y0,z0,x1,y1) RETURN(g(x,y,x0,y0,z0)+h(x,y,x0,y0,z0,x1,y1))
end
```

```
> j:=proc(x,y,z,x0,y0,z0,x1,y1);
> RETURN(i(x,y,x0,y0,z0,x1,y1)*f(x0,y0,z)/f(x0,y0,z0));
> end;

j := proc(x,y,z,x0,y0,z0,x1,y1)
         RETURN(i(x,y,x0,y0,z0,x1,y1)*f(x0,y0,z)/f(x0,y0,z0))
     end
```

```
> k:=proc(x,y,z,x0,y0,z0,x1,y1,z1);
> RETURN((i(x,y,x0,y0,z1,x1,y1)-j(x,y,z1,x0,y0,z0,x1,y1))*(f(x1,y1,z)
> -j(x1,y1,z,x0,y0,z0,x1,y1))/(f(x1,y1,z1)-j(x1,y1,z1,x0,y0,z0,x1,y1)));
> end;

k := proc(x,y,z,x0,y0,z0,x1,y1,z1)
         RETURN((i(x,y,x0,y0,z1,x1,y1)-j(x,y,z1,x0,y0,z0,x1,y1))*
             (f(x1,y1,z)-j(x1,y1,z,x0,y0,z0,x1,y1))/
             (f(x1,y1,z1)-j(x1,y1,z1,x0,y0,z0,x1,y1)))
     end
```

```
> l:=j(x,y,z,x0,y0,z0,x1,y1,z1)+k(x,y,z,x0,y0,z0,x1,y1,z1);
```

$$
\begin{aligned}
l := \Bigg(&\frac{\%8\,\%7}{\%1} + \Bigg(\mathrm{f1}(\,x1\,)\,\mathrm{f2}(\,y\,)\,\mathrm{f3}(\,z0\,) + \mathrm{f4}(\,x1\,)\,\mathrm{f5}(\,y\,)\,\mathrm{f6}(\,z0\,) - \frac{\%8\,\%5}{\%1}\Bigg) \\
&\Bigg(\mathrm{f1}(\,x\,)\,\mathrm{f2}(\,y1\,)\,\mathrm{f3}(\,z0\,) + \mathrm{f4}(\,x\,)\,\mathrm{f5}(\,y1\,)\,\mathrm{f6}(\,z0\,) - \frac{\%6\,\%7}{\%1}\Bigg) \Bigg/ \Bigg(\\
&\%4 + \%3 - \frac{\%6\,\%5}{\%1}\Bigg)\Bigg) \\
&(\,\mathrm{f1}(\,x0\,)\,\mathrm{f2}(\,y0\,)\,\mathrm{f3}(\,z\,) + \mathrm{f4}(\,x0\,)\,\mathrm{f5}(\,y0\,)\,\mathrm{f6}(\,z\,)\,) \Big/ (\,\%1\,) + \Bigg(\\
&(\,\mathrm{f1}(\,x0\,)\,\mathrm{f2}(\,y\,)\,\mathrm{f3}(\,z1\,) + \mathrm{f4}(\,x0\,)\,\mathrm{f5}(\,y\,)\,\mathrm{f6}(\,z1\,)\,) \\
&(\,\mathrm{f1}(\,x\,)\,\mathrm{f2}(\,y0\,)\,\mathrm{f3}(\,z1\,) + \mathrm{f4}(\,x\,)\,\mathrm{f5}(\,y0\,)\,\mathrm{f6}(\,z1\,)\,) \Big/ (\,\%2\,) + (\\
&\mathrm{f1}(\,x1\,)\,\mathrm{f2}(\,y\,)\,\mathrm{f3}(\,z1\,) + \mathrm{f4}(\,x1\,)\,\mathrm{f5}(\,y\,)\,\mathrm{f6}(\,z1\,) - \\
&(\,\mathrm{f1}(\,x0\,)\,\mathrm{f2}(\,y\,)\,\mathrm{f3}(\,z1\,) + \mathrm{f4}(\,x0\,)\,\mathrm{f5}(\,y\,)\,\mathrm{f6}(\,z1\,)\,) \\
&(\,\mathrm{f1}(\,x1\,)\,\mathrm{f2}(\,y0\,)\,\mathrm{f3}(\,z1\,) + \mathrm{f4}(\,x1\,)\,\mathrm{f5}(\,y0\,)\,\mathrm{f6}(\,z1\,)\,) \Big/ (\,\%2\,))(\\
&\mathrm{f1}(\,x\,)\,\mathrm{f2}(\,y1\,)\,\mathrm{f3}(\,z1\,) + \mathrm{f4}(\,x\,)\,\mathrm{f5}(\,y1\,)\,\mathrm{f6}(\,z1\,) - \\
&(\,\mathrm{f1}(\,x0\,)\,\mathrm{f2}(\,y1\,)\,\mathrm{f3}(\,z1\,) + \mathrm{f4}(\,x0\,)\,\mathrm{f5}(\,y1\,)\,\mathrm{f6}(\,z1\,)\,) \\
&(\,\mathrm{f1}(\,x\,)\,\mathrm{f2}(\,y0\,)\,\mathrm{f3}(\,z1\,) + \mathrm{f4}(\,x\,)\,\mathrm{f5}(\,y0\,)\,\mathrm{f6}(\,z1\,)\,) \Big/ (\,\%2\,)) \Big/ (
\end{aligned}
$$

$$\mathrm{f1}(\,x1\,)\,\mathrm{f2}(\,y1\,)\,\mathrm{f3}(\,z1\,)+\mathrm{f4}(\,x1\,)\,\mathrm{f5}(\,y1\,)\,\mathrm{f6}(\,z1\,)-$$
$$(\,\mathrm{f1}(\,x0\,)\,\mathrm{f2}(\,y1\,)\,\mathrm{f3}(\,z1\,)+\mathrm{f4}(\,x0\,)\,\mathrm{f5}(\,y1\,)\,\mathrm{f6}(\,z1\,)\,)$$
$$(\,\mathrm{f1}(\,x1\,)\,\mathrm{f2}(\,y0\,)\,\mathrm{f3}(\,z1\,)+\mathrm{f4}(\,x1\,)\,\mathrm{f5}(\,y0\,)\,\mathrm{f6}(\,z1\,)\,)\Big/(\,\%2\,))-\Bigg($$
$$\frac{\%8\,\%7}{\%1}+\left(\mathrm{f1}(\,x1\,)\,\mathrm{f2}(\,y\,)\,\mathrm{f3}(\,z0\,)+\mathrm{f4}(\,x1\,)\,\mathrm{f5}(\,y\,)\,\mathrm{f6}(\,z0\,)-\frac{\%8\,\%5}{\%1}\right)$$
$$\left(\mathrm{f1}(\,x\,)\,\mathrm{f2}(\,y1\,)\,\mathrm{f3}(\,z0\,)+\mathrm{f4}(\,x\,)\,\mathrm{f5}(\,y1\,)\,\mathrm{f6}(\,z0\,)-\frac{\%6\,\%7}{\%1}\right)\Bigg/\Bigg($$
$$\%4+\%3-\frac{\%6\,\%5}{\%1}\Bigg)\Bigg)\,\%2\Big/(\,\%1\,)\Bigg)\,\Bigg(\mathrm{f1}(\,x1\,)\,\mathrm{f2}(\,y1\,)\,\mathrm{f3}(\,z\,)$$
$$+\mathrm{f4}(\,x1\,)\,\mathrm{f5}(\,y1\,)\,\mathrm{f6}(\,z\,)$$
$$-\frac{(\,\%4+\%3\,)\,(\,\mathrm{f1}(\,x0\,)\,\mathrm{f2}(\,y0\,)\,\mathrm{f3}(\,z\,)+\mathrm{f4}(\,x0\,)\,\mathrm{f5}(\,y0\,)\,\mathrm{f6}(\,z\,)\,)}{\%1}\Bigg)\Bigg/\Bigg($$
$$\mathrm{f1}(\,x1\,)\,\mathrm{f2}(\,y1\,)\,\mathrm{f3}(\,z1\,)+\mathrm{f4}(\,x1\,)\,\mathrm{f5}(\,y1\,)\,\mathrm{f6}(\,z1\,)-\frac{(\,\%4+\%3\,)\,\%2}{\%1}\Bigg)$$
$$\%1:=\mathrm{f1}(\,x0\,)\,\mathrm{f2}(\,y0\,)\,\mathrm{f3}(\,z0\,)+\mathrm{f4}(\,x0\,)\,\mathrm{f5}(\,y0\,)\,\mathrm{f6}(\,z0\,)$$
$$\%2:=\mathrm{f1}(\,x0\,)\,\mathrm{f2}(\,y0\,)\,\mathrm{f3}(\,z1\,)+\mathrm{f4}(\,x0\,)\,\mathrm{f5}(\,y0\,)\,\mathrm{f6}(\,z1\,)$$
$$\%3:=\mathrm{f4}(\,x1\,)\,\mathrm{f5}(\,y1\,)\,\mathrm{f6}(\,z0\,)$$
$$\%4:=\mathrm{f1}(\,x1\,)\,\mathrm{f2}(\,y1\,)\,\mathrm{f3}(\,z0\,)$$
$$\%5:=\mathrm{f1}(\,x1\,)\,\mathrm{f2}(\,y0\,)\,\mathrm{f3}(\,z0\,)+\mathrm{f4}(\,x1\,)\,\mathrm{f5}(\,y0\,)\,\mathrm{f6}(\,z0\,)$$
$$\%6:=\mathrm{f1}(\,x0\,)\,\mathrm{f2}(\,y1\,)\,\mathrm{f3}(\,z0\,)+\mathrm{f4}(\,x0\,)\,\mathrm{f5}(\,y1\,)\,\mathrm{f6}(\,z0\,)$$
$$\%7:=\mathrm{f1}(\,x\,)\,\mathrm{f2}(\,y0\,)\,\mathrm{f3}(\,z0\,)+\mathrm{f4}(\,x\,)\,\mathrm{f5}(\,y0\,)\,\mathrm{f6}(\,z0\,)$$
$$\%8:=\mathrm{f1}(\,x0\,)\,\mathrm{f2}(\,y\,)\,\mathrm{f3}(\,z0\,)+\mathrm{f4}(\,x0\,)\,\mathrm{f5}(\,y\,)\,\mathrm{f6}(\,z0\,)$$

```
> simplify(l);
```

$$\mathrm{f2}(\,y\,)\,\mathrm{f1}(\,x\,)\,\mathrm{f3}(\,z\,)+\mathrm{f5}(\,y\,)\,\mathrm{f4}(\,x\,)\,\mathrm{f6}(\,z\,)$$

Annexe C

Application de la méthode de transformation de polynômes

Dans cette annexe nous présentons l'application de la méthode de transformation de polynômes décrit dans le paragraph 4.2 à un polynôme à quatre variables d'entrée et deux termes. Parmi les quatre variables d'entrée deux (x, y) sont dans le sous modèle et les deux autres (z, w) sont dehors. Comme pour les annexes précédentes cette présentation est un fichier généré par Maple.

```
> f:=proc(x,y,z,w);
> RETURN(f1(x)*f2(y)*f5(z)*f7(w)+f3(x)*f4(y)*f6(z)*f8(w));
>  end;

f := proc(x,y,z,w) RETURN(f1(x)*f2(y)*f5(z)*f7(w)+f3(x)*f4(y)*f6(z)*f8(w)) end
```

```
> g:=proc(x,y,z,w,x0,y0,z0,w0)
> RETURN(f(x0,y0,z,w)*f(x,y,z0,w0)/f(x0,y0,z0,w0));
> end;

g := proc(x,y,z,w,x0,y0,z0,w0)
        RETURN(f(x0,y0,z,w)*f(x,y,z0,w0)/f(x0,y0,z0,w0))
    end
```

```
> h:=proc(x,y,z,w,x0,y0,z0,w0,x1,y1,z1,w1) local ret;
>  ret:=((f(x,y,z1,w1)-g(x,y,z1,w1,x0,y0,z0,w0))*
> (f(x1,y1,z,w)-g(x1,y1,z,w,x0,y0,z0,w0)))/
> (f(x1,y1,z1,w1)-g(x1,y1,z1,w1,x0,y0,z0,w0));
> RETURN(ret);
> end;

h := proc(x,y,z,w,x0,y0,z0,w0,x1,y1,z1,w1)
    local ret;
        ret := (f(x,y,z1,w1)-g(x,y,z1,w1,x0,y0,z0,w0))*
```

```
            (f(x1,y1,z,w)-g(x1,y1,z,w,x0,y0,z0,w0))/
            (f(x1,y1,z1,w1)-g(x1,y1,z1,w1,x0,y0,z0,w0));
        RETURN(ret)
    end
```

```
> i:=g(x,y,zz,ww,x0,y0,z0,w0)+h(x,y,zz,ww,x0,y0,z0,w0,x1,y1,z1,w1);
```

$$
\begin{aligned}
i := &\,(\mathrm{f1}(\mathit{x0})\,\mathrm{f2}(\mathit{y0})\,\mathrm{f5}(\mathit{zz})\,\mathrm{f7}(\mathit{ww}) + \mathrm{f3}(\mathit{x0})\,\mathrm{f4}(\mathit{y0})\,\mathrm{f6}(\mathit{zz})\,\mathrm{f8}(\mathit{ww}))\\
&(\mathrm{f1}(x)\,\mathrm{f2}(y)\,\mathrm{f5}(\mathit{z0})\,\mathrm{f7}(\mathit{w0}) + \mathrm{f3}(x)\,\mathrm{f4}(y)\,\mathrm{f6}(\mathit{z0})\,\mathrm{f8}(\mathit{w0}))\Big/(\%1)\\
&+ (\mathrm{f1}(x)\,\mathrm{f2}(y)\,\mathrm{f5}(\mathit{z1})\,\mathrm{f7}(\mathit{w1}) + \mathrm{f3}(x)\,\mathrm{f4}(y)\,\mathrm{f6}(\mathit{z1})\,\mathrm{f8}(\mathit{w1}) -\\
&(\mathrm{f1}(\mathit{x0})\,\mathrm{f2}(\mathit{y0})\,\mathrm{f5}(\mathit{z1})\,\mathrm{f7}(\mathit{w1}) + \mathrm{f3}(\mathit{x0})\,\mathrm{f4}(\mathit{y0})\,\mathrm{f6}(\mathit{z1})\,\mathrm{f8}(\mathit{w1}))\\
&(\mathrm{f1}(x)\,\mathrm{f2}(y)\,\mathrm{f5}(\mathit{z0})\,\mathrm{f7}(\mathit{w0}) + \mathrm{f3}(x)\,\mathrm{f4}(y)\,\mathrm{f6}(\mathit{z0})\,\mathrm{f8}(\mathit{w0}))\Big/(\%1)\\
&)(\mathrm{f1}(\mathit{x1})\,\mathrm{f2}(\mathit{y1})\,\mathrm{f5}(\mathit{zz})\,\mathrm{f7}(\mathit{ww}) + \mathrm{f3}(\mathit{x1})\,\mathrm{f4}(\mathit{y1})\,\mathrm{f6}(\mathit{zz})\,\mathrm{f8}(\mathit{ww}) -\\
&(\mathrm{f1}(\mathit{x0})\,\mathrm{f2}(\mathit{y0})\,\mathrm{f5}(\mathit{zz})\,\mathrm{f7}(\mathit{ww}) + \mathrm{f3}(\mathit{x0})\,\mathrm{f4}(\mathit{y0})\,\mathrm{f6}(\mathit{zz})\,\mathrm{f8}(\mathit{ww}))\\
&(\mathrm{f1}(\mathit{x1})\,\mathrm{f2}(\mathit{y1})\,\mathrm{f5}(\mathit{z0})\,\mathrm{f7}(\mathit{w0}) + \mathrm{f3}(\mathit{x1})\,\mathrm{f4}(\mathit{y1})\,\mathrm{f6}(\mathit{z0})\,\mathrm{f8}(\mathit{w0}))\Big/(\\
&\%1))\Big/(\mathrm{f1}(\mathit{x1})\,\mathrm{f2}(\mathit{y1})\,\mathrm{f5}(\mathit{z1})\,\mathrm{f7}(\mathit{w1})\\
&+ \mathrm{f3}(\mathit{x1})\,\mathrm{f4}(\mathit{y1})\,\mathrm{f6}(\mathit{z1})\,\mathrm{f8}(\mathit{w1}) -\\
&(\mathrm{f1}(\mathit{x0})\,\mathrm{f2}(\mathit{y0})\,\mathrm{f5}(\mathit{z1})\,\mathrm{f7}(\mathit{w1}) + \mathrm{f3}(\mathit{x0})\,\mathrm{f4}(\mathit{y0})\,\mathrm{f6}(\mathit{z1})\,\mathrm{f8}(\mathit{w1}))\\
&(\mathrm{f1}(\mathit{x1})\,\mathrm{f2}(\mathit{y1})\,\mathrm{f5}(\mathit{z0})\,\mathrm{f7}(\mathit{w0}) + \mathrm{f3}(\mathit{x1})\,\mathrm{f4}(\mathit{y1})\,\mathrm{f6}(\mathit{z0})\,\mathrm{f8}(\mathit{w0}))\Big/(\\
&\%1))\\
&\%1 := \mathrm{f1}(\mathit{x0})\,\mathrm{f2}(\mathit{y0})\,\mathrm{f5}(\mathit{z0})\,\mathrm{f7}(\mathit{w0}) + \mathrm{f3}(\mathit{x0})\,\mathrm{f4}(\mathit{y0})\,\mathrm{f6}(\mathit{z0})\,\mathrm{f8}(\mathit{w0})
\end{aligned}
$$

```
> simplify(i);
```

$$\mathrm{f3}(x)\,\mathrm{f4}(y)\,\mathrm{f6}(\mathit{zz})\,\mathrm{f8}(\mathit{ww}) + \mathrm{f5}(\mathit{zz})\,\mathrm{f7}(\mathit{ww})\,\mathrm{f1}(x)\,\mathrm{f2}(y)$$

Annexe D

Les polynômes interpolateurs

D.1 Les polynômes algébriques

Dans ce paragraphe nous allons montrer la façon la plus élémentaire de faire l'identification des paramètres des polynômes interpolateurs, particulièrement des **polynômes algébriques**.

Pour illustrer la méthode d'interpolation classique on va estimer une fonction arbitraire $y = f(x)$ à l'aide un polynôme algébrique du deuxième dègré.

$$P(x) = a_0 + a_1 x + a_2 x^2 \tag{D.1}$$

On va choisir un point x_1 à l'interieur de l'intervalle $[x_0, x_2]$. On dispose alors de trois points pour lesquels on obtient les valeurs correspondantes de notre fonction de forme.

$$y_0 = f(x_0),\ y_1 = f(x_1),\ y_2 = f(x_2) \tag{D.2}$$

On va alors construire un polynôme de la forme D.1 tel qu'aux points x_0, x_1, x_2 ce polynôme s'accorde à la fonction de forme en question. C'est-à-dire qu'on doit choisir les coefficients a_0, a_1, a_2 du polynôme D.1 de telle façon qu'il puisse satisfaire les équations :

$$P(x_0) = y_0,\ P(x_1) = y_1,\ P(x_2) = y_2 \tag{D.3}$$

Les points sur lequels le polynôme interpolateur doit s'accorder à la fonction de forme sont appelés *points d'interpolation.*

Pour résoudre le problème d'interpolation nous pouvons réécrire D.3 :

$$\begin{aligned} y_0 &= a_0 + a_1 x + a_2 x^2 \\ y_1 &= a_0 + a_1 x + a_2 x^2 \\ y_2 &= a_0 + a_1 x + a_2 x^2 \end{aligned}$$

On doit maintenant résoudre ces trois équations pour a_0, a_1, a_2 et finalement substituer les valeurs de ces coefficients en D.1.

Il est clair qu'une fonction de forme complexe ne peut pas être bien estimée par un polynôme de dégré deux. Dans la pratique on doit utiliser polynômes d'un dégré plus élevé (pas moins de 4).

Le problème général d'interpolation consiste à construire un polynôme $P_n(x) = a_0 + a_1 x + a_2 x^2 + ... + a_n x^n$ de dégré n qui s'accorde avec une fonction donnée en $n + 1$ équations :

$$P(x_0) = f(x_0),\ P(x_1) = f(x_1), ...,\ P(x_n) = f(x_n)$$

La méthode d'interpolation présentée est un moyen universel d'approximation de fonctions. En principe, la fonction à interpoler n'est censée avoir aucune propriété particulière pour que cette interpolation fonctionne [AKL86].

Bien sur, les questions qui surviennent à chaque cas sont : combien de points d'interpolation utiliser (quel degré de polynôme) et quelle distribution de points choisir de façon à que l'erreur résiduelle soit satisfaisante.

Le choix du nombre de points d'interpolation n'est pas élémentaire, et même l'idée intuitive qu'un nombre toujours plus grand de points signifie une erreur résiduelle toujours moins importante est fausse dans le cas des polynômes algébriques. Cette question n'étant pas au centre du sujet de cette thèse, nous n'avons pas approfondi la recherche d'une solution idéale, et nous nous contentons d'un choix empirique du nombre de points au cas par cas, basé sur des essais.

La meilleure distribution des points d'interpolation dans l'intervalle $[-1, 1]$ pour une fonction continue est donnée par les points x_k zéros du polynôme de Čebyšev (D.4) [AKL86].

$$x_k = cos\frac{2k+1}{2(n+1)}\pi \ (k = 0, 1, ..., n) \tag{D.4}$$

n étant le nombre de points d'interpolation.

D.2 Les polynômes trigonométriques

Dans ce paragraphe nous allons étudier une deuxième classe de polynômes interpolateur, les **polynômes trigonométriques**, aussi connus comme les *séries de Fourier*. Un polynôme trigonométrique d'ordre n est une fonction de la forme :

$$u_n(x) = \alpha_0 + \alpha_1 cosx + \beta_1 sinx + \alpha_2 cos2x + \beta_2 sin2x + ... + \alpha_n cosnx + \beta_n sinnx \tag{D.5}$$

où dans une forme plus concise :

$$u_n(x) = \alpha_0 + \sum_{k=1}^{n}(\alpha_k cos(kx) + \beta_k sin(kx)) \tag{D.6}$$

où α_k et β_k sont des constantes.

Les polynômes trigonométriques ont été initialement proposés dans le cadre de l'étude de certains phénomènes physiques, en particulier des petites oscillations d'objets élastiques.

Une fonction périodique donnée $\phi(t)$ qui décrit une oscillation arbitraire de période $2\pi/\alpha$ [1] d'un point x_0 peut être représentée par la série :

$$\phi(t) = A_0 + \sum_{k=1}^{\infty}(A_k cos\alpha kt + B_k sin\alpha kt) \tag{D.7}$$

Il y a pourtant beaucoup d'autres situations en physique où il est naturel de considérer une fonction donnée comme étant la somme d'une série trigonométrique infinie de la forme D.7, même si cette fonction ne décrit pas une oscillation.

En 1829 le mathématicien allemand Dirichlet a démontré que toute fonction continue de période $\frac{2\pi}{\alpha}$, ayant dans une période un nombre fini de maxima et minima, peut être développée en une seule série trigonométrique.

Pour estimer une fonction $f(x)$ dans la pratique on doit utiliser de période 2π la somme finie D.6, et on doit estimer les valeurs de α_k et β_k.

Cependant, on peut voir ce problème d'identification de paramètres des polynômes trigonométriques d'un autre point de vue en utilisant les "méthodes de transformation de Fourier".

Un processus physique peut être décrit soit dans le *domaine temporel*, par les valeurs d'une quantité h étant fonction du temps t, c'est-à-dire $h(t)$, soit dans le *domaine fréquentiel*, où le processus est spécifié par une amplitude calculée H [2] étant fonction d'une fréquence f, c'est-à-dire $H(f)$, avec $-\infty < f < \infty$. Il est très utile dans plusieurs situations de voir $h(t)$ et $H(f)$ comme étant deux *représentations* de la *même* fonction. On peut passer d'une représentation à l'autre par les équations de la *Transformation de Fourier* [Pre92].

$$\begin{aligned} H(f) &= \int_{-\infty}^{\infty} h(t)e^{2\pi ift}dt \\ h(t) &= \int_{-\infty}^{\infty} H(f)e^{-2\pi ift}df \end{aligned} \tag{D.8}$$

En pratique, pour passer d'une representation temporelle à une représenta-

[1] La fonction $f(x)$ a la période ω si elle satisfait l'équation $f(x+\omega) = f(x)$

[2] H est généralement un nombre complexe indiquant aussi la phase. Les fonctions trigonométriques dans le domaine des complexes ont une forme exponentielle donnée par $e^{iz} = cosz + isinz$

tion fréquentielle on va utiliser une *transformation de Fourier discrète*, basée sur un ensemble de N points de h, et l'équation D.8 va être estimée par l'équation D.9.

$$H_n = \sum_{k=0}^{N-1} h_k e^{2\pi ikn/N} \tag{D.9}$$

Et $H(f_n) \approx \Delta H_n$ où Δ est un intervalle d'échantillonnage de h.

Le calcul des amplitudes H_n équivaut au problème initial d'estimation des paramètres α_k et β_k des polynômes trigonométriques.

Annexe E

Preuve du théorème de la méthode de transformation de modèles

Dans cet annexe nous allons développer une preuve du théorème proposé au paragraphe 4.2. Cette preuve est équivalente à celle décrite aux paragraphes 3.3.1 et 3.3.2.

E.0.1 Première preuve : $\Gamma_n^* \subset \mathcal{F}_n^*$

Pour la démonstration de la première partie prenons d'abord une fonction g_m appartenant à Γ_0^* et démontrons que $g_m \in \mathcal{F}_0^*$. Nous allons réécrire 4.6 pour cette fonction (E.1).

$$\begin{aligned} DC^0 g_m &= 0 \\ g_m(x) &= \frac{g_{i_0}(x).g_m(x_0)}{g_{i_0}(x_0)} \end{aligned} \tag{E.1}$$

Nous pouvons réécrire E.1 en utilisant :

$$\phi_0 = \frac{g_m(x_0)}{g_{i_0}(x_0)} \tag{E.2}$$

et nous allons obtenir :

$$g_m(x) = g_{i_0}(x).\phi_0 \tag{E.3}$$

Nous avons prouvé que g_m appartient à $\mathcal{F}_0^*$.

Supposons maintenant que :

$$\Gamma_{n-1}^* \subset \mathcal{F}_{n-1}^* \tag{E.4}$$

Pour la preuve par induction nous devons prendre une fonction k_m appartenant à Γ_n^* et réécrire 4.6 pour cette fonction (équation E.5). La définition récursive de l'ensemble Γ^* nous dit que $DC^0 k_m$ est une fonction qu'appartient à Γ_{n-1}^*. Par hypothèse une fonction appartenant à Γ_{n-1}^* appartient aussi à $\mathcal{F}_{n-1}^*$.

$$DC^0 k_m(x) = k_m(x) - \frac{k_{i_0}(x).k_m(x_0)}{k_{i_0}(x_0)} = \varphi_0(x).\psi_0^m + ... + \varphi_{n-1}(x).\psi_{n-1}^m \tag{E.5}$$

Nous pouvons réécrire E.5 en utilisant :

$$\begin{aligned} \varphi_n(x) &= k_{i_0}(x) \\ \psi_n^m &= \frac{k_m(x_0)}{k_{i_0}(x_0)} \end{aligned}$$

et nous allons obtenir :

$$k_m(x) = \varphi_0(x).\psi_0^m + ... + \varphi_{n-1}(x).\psi_{n-1}^m + \varphi_n(x).\psi_n^m \tag{E.6}$$

CQFD.

E.0.2 Deuxième preuve : $\mathcal{F}_n^* \subset \Gamma_n^*$

Dans la deuxième partie de la preuve nous voulons prouver que toute fonction appartenant à l'ensemble $\mathcal{F}_n^*$ appartient aussi à Γ_n^*.

Nous commençons par montrer que les fonctions de $\mathcal{F}_0^*$ appartiennent aussi à Γ_0^*. Par définition les fonctions de $\mathcal{F}_0^*$ s'ecrivent :

$$g_m(x) = \varphi_0(x).\psi_0^m \tag{E.7}$$

d'où :

$$DC^0 g_m(x) = \varphi_0(x).\psi_0^m - \frac{\varphi_0(x).\psi_0^{i_0}.\varphi_0(x_0).\psi_0^m}{\varphi_0(x_0).\psi_0^{i_0}} \tag{E.8}$$

On voit bien que $\psi_0^{i_0}$ et les termes en x_0 s'annulent et que la différence résulte en 0. De cette façon on prouve que les fonctions de $\mathcal{F}_0^*$ appartiennent aussi à Γ_0^*.

Nous allons maintenant supposer comme hypothèse de recurrence que :

$$f_m \in \mathcal{F}_{n-1}^* \Rightarrow f_m \in \Gamma_{n-1}^* \tag{E.9}$$

Nous allons utiliser une fonction générale f_m de $\mathcal{F}_{n-1}^*$ (équation E.10).

$$f_m(x) = \sum_{i=0}^{n-1} \varphi_i(x).\psi_i^m \tag{E.10}$$

Nous allons utiliser encore deux autres fonctions :

$$g_m(x) = \varphi_n(x).\psi_n^m \tag{E.11}$$

$$h_m(x) = f_m(x) + g_m(x) \tag{E.12}$$

h_m est une fonction de $\mathcal{F}_n^*$, et sa DC^0 aura la forme :

$$\begin{aligned} &DC^0 h_m(x) = ((\sum_{i=0}^{n-1} \varphi_i(x).\psi_i^m) + \varphi_n(x).\psi_n^m) - \\ &\frac{((\sum_{i=0}^{n-1} \varphi_i(x).\psi_i^{i_0}) + \varphi_n(x).\psi_n^{i_0}).((\sum_{i=0}^{n-1} \varphi_i(x_0).\psi_i^m) + \varphi_n(x_0).\psi_n^m)}{((\sum_{i=0}^{n-1} \varphi_i(x_0).\psi_i^{i_0}) + \varphi_n(x_0).\psi_n^{i_0})} \end{aligned} \tag{E.13}$$

Nous changons maintenant les notations de E.13. Notons :

- $\alpha = \sum_{i=0}^{n-1} \varphi_i(x).\psi_i^{i_0}$
- $\alpha_n = \varphi_n(x).\psi_n^{i_0}$
- $\beta = \sum_{i=0}^{n-1} \varphi_i(x_0).\psi_i^m$
- $\beta_n = \varphi_n(x_0).\psi_n^m$
- $p = \sum_{i=0}^{n-1} \varphi_i(x_0).\psi_i^{i_0}$
- $p_n = \varphi_n(x_0).\psi_n^{i_0}$

En utilisant notre nouvelle notation $DC^0 h_m$ se présente de la forme suivante :

$$DC^0 h_m(x) = \sum_{i=0}^{n-1} \varphi_i(x).\psi_i^m + \varphi_n(x).\psi_n^m - \frac{(\alpha + \alpha_n)(\beta + \beta_n)}{(p + p_n)} \tag{E.14}$$

En développant E.14 on obtient :

$$DC^0 h_m(x) = \sum_{i=0}^{n-1} \varphi_i(x).\psi_i^m + \varphi_n(x).\psi_n^m - \frac{(\alpha.\beta + \alpha.\beta_n + \alpha_n.\beta + \alpha_n.\beta_n)}{p + p_n} \tag{E.15}$$

On sépare les termes :

$$DC^0 h_m(x) = \sum_{i=0}^{n-1} \varphi_i(x).\psi i^m + \varphi_n(x).\psi_n^m - \left(\frac{\alpha.\beta}{p+p_n} + \frac{\alpha.\beta_n + \alpha_n.\beta}{p+p_n} + \frac{\alpha_n.\beta_n}{p+p_n}\right) \quad \text{(E.16)}$$

Nous allons utiliser la même transformation déjà proposée dans l'équation 3.28 pour réécrire deux termes de l'équation E.16 :

$$\frac{\alpha.\beta}{p_n+p} = \frac{\alpha.\beta}{p} - \frac{\alpha.\beta.p_n}{(p_n+p).p}$$
$$\frac{\alpha_n.\beta_n}{p+p_n} = \frac{\alpha_n.\beta_n}{p_n} - \frac{\alpha_n.\beta_n.p}{(p+p_n).p_n}$$

Après ces transformations on obtient :

$$\begin{aligned} DC^0 h_m(x) = \quad & \textstyle\sum_{i=0}^{n-1} \varphi_i(x).\psi_i^m) + \varphi_n(x).\psi_n^m \\ & -\tfrac{\alpha.\beta}{p} + \tfrac{\alpha.\beta.p_n}{(p_n+p).p} - \tfrac{\alpha.\beta_n+\alpha_n.\beta}{p+p_n} - \tfrac{\alpha_n.\beta_n}{p_n} + \tfrac{\alpha_n.\beta_n.p}{(p+p_n).p_n} \end{aligned} \quad \text{(E.17)}$$

Parmi les termes de E.17 nous retrouvons $\frac{\alpha.\beta}{p}$. Si on réécrit ce terme avec la notation originale :

$$\frac{\alpha.\beta}{p} = \frac{(\sum_{i=0}^{n-1} \varphi_i(x).\psi_i^{i_0}).(\sum_{i=0}^{n-1} \varphi_i(x_0).\psi i^m)}{(\sum_{i=0}^{n-1} \varphi_i(x_0).\psi_i^{i_0})}$$

Si on regroupe ce terme avec le premier terme de E.17 on obtient :

$$\sum_{i=0}^{n-1} \varphi_i(x).\psi_i^m - \frac{(\sum_{i=0}^{n-1} \varphi_i(x).\psi_i^{i_0}).(\sum_{i=0}^{n-1} \varphi_i(x_0).\psi i^m)}{(\sum_{i=0}^{n-1} \varphi_i(x_0).\psi_i^{i_0})} = DC^0 f_m(x) \quad \text{(E.18)}$$

Le terme $\frac{\alpha_n.\beta_n}{p_n}$ nous intéresse aussi :

$$\frac{\alpha_n.\beta_n}{p_n} = \frac{\varphi_n(x).\psi_n^{i_0}.\varphi_n(x_0).\psi_n^m}{\varphi_n(x_0).\psi_n^{i_0}}$$

Si on regroupe ce terme et $\varphi_n(x).\psi_n^m$ on obtient :

$$\varphi_n(x).\psi_n^m - \frac{\varphi_n(x).\psi_n^{i_0}.\varphi_n(x_0).\psi_n^m}{\varphi_n(x_0).\psi_n^{i_0}} = DC^0 g_m(x) = 0$$

Après ces simplifications $DC^0 h$ se présente comme :

$$DC^0 h_m(x) = DC^0 f_m(x) + \frac{\alpha.\beta.p_n}{(p_n+p).p} - \frac{\alpha.\beta_n + \alpha_n.\beta}{p+p_n} + \frac{\alpha_n.\beta_n.p}{(p+p_n).p_n} \qquad \text{(E.19)}$$

Si on regroupe les trois derniers termes :

$$DC^0 h_m(x) = DC^0 f_m(x) + \frac{\alpha.\beta.p_n^2 + p^2.\alpha_n.\beta_n - p.p_n.\alpha.\beta_n - p.p_n.\beta.\alpha_n}{p^2.p_n + p.p_n^2} \qquad \text{(E.20)}$$

On peut réécrire le nouveau terme comme un produit de deux termes divisé par un troisième terme constant :

$$DC^0 h_m(x) = DC^0 f_m(x) + \frac{(\alpha.p_n - p.\alpha_n).(\beta.p_n - p.\beta_n)}{p^2.p_n + p.p_n^2} \qquad \text{(E.21)}$$

Nous allons réécrire E.21 en utilisant une nouvelle fonction et un nouveau scalaire :

$$\sigma(x) = \frac{\alpha.p_n - p.\alpha_n}{p^2.p_n + p.p_n^2}$$
$$= \frac{(\sum_{i=0}^{n-1} \varphi_i(x).\psi_i^{i_0}).\varphi_n(x_0).\psi_n^{i_0} - (\sum_{i=0}^{n-1} \varphi_i(x_0).\psi_i^{i_0}).\varphi_n(x).\psi_n^{i_0}}{(\sum_{i=0}^{n-1} \varphi_i(x_0).\psi_i^{i_0})^2.\varphi_n(x_0).\psi_n^{i_0} + (\sum_{i=0}^{n-1} \varphi_i(x_0).\psi_i^{i_0}).\varphi_n(x_0)^2.(\psi_n^{i_0})^2}$$

$$\rho = \beta.p_n - p.\beta_n$$
$$= (\sum_{i=0}^{n-1} \varphi_i(x_0).\psi_i^m).\varphi_n(x_0).\psi_n^{i_0} - (\sum_{i=0}^{n-1} \varphi_i(x_0).\psi_i^{i_0}).\varphi_n(x_0).\psi_n^m$$

L'équation E.21 devient donc :

$$DC^0 h_m(x) = DC^0 f_m(x) + \sigma(x).\rho \tag{E.22}$$

Sachant que :

- d'après notre hypothèse de recurrence, la fonction f_m appartient à $\mathcal{F}_{n-1}^*$ et donc à Γ_{n-1}^*.
- par définition la transformation DC^0 des fonctions de Γ_{n-1}^* définit une fonction qui à son tour appartient à Γ_{n-2}^*.
- nous avons prouvé auparavant que les fonctions de Γ_{n-2}^* appartiennent à $\mathcal{F}_{n-2}^*$ (paragraphe E.0.1).

Nous pouvons donc conclure que le terme $DC^0 f_m(x)$ dans E.22 est une fonction de $\mathcal{F}_{n-2}^*$.

Dans l'équation E.22 nous avons en conséquence une fonction de $\mathcal{F}_{n-2}^*$ plus une fonction de $\mathcal{F}_0^*$, ce qui nous fait conclure que $DC^0 h(x)^*$ est une fonction de $\mathcal{F}_{n-1}^*$.

Depuis notre hypothèse de recurrence nous pouvons conclure aussi que si $DC^0 h_m(x) \in \mathcal{F}_{n-1}^*$ alors $DC^0 h_m(x) \in \Gamma_{n-1}^*$, et en consequence que $h_m \in \Gamma_n^*$. Donc par induction nous pouvons conclure que effectivement $h_m \in \mathcal{F}_n^* \Rightarrow h \in \Gamma_n^*$.

CQFD.

Bibliographie

[AKL86] Aleksandrov (D.), Kolmogorov (A. N.) et Lavrent'ev (M.A.) (édité par). – *MATHEMATICS Its Content, Methods, and Meaning.* – The MIT Press, 1986.

[AO90] Ahmad (S.) et Omohundro (S.). – *A network for extracting the locations of point clusters using selective attention.* – Rapport technique, University of California, 1990. Tech. Rep. 90-011, Int. Computer Science Institute.

[Atk90] Atkeson (Christopher G.). – Memory-based approaches to approximating continuous functions. *In : 1990 Workshop on Nonlinear Modeling and Forecasting*, pp. 17–21.

[BDML94] Bessière (Pierre), Dedieu (Eric), Mazer (Emmanuel) et Lebeltel (Olivier). – The beam in the bin experiment; an application of probability as logic to autonomous robotic. *In : MaxEnt94.* – Cambridge, Angleterre, 1994.

[Ber93] Berns (K.). – Applications of neural networks in robotics. – reproduction autorisée dans l'annexe du rapport : *La Robotique Connexionniste : Enquête documentaire sur l'utilisation des algorithmes connexionnistes en robotique*, 1993.

[BF92] Baker (Walter L.) et Farrel (Jay A.). – An introduction to connectionist learning control systems. *In : Handbook of Intelligent Control, Neural, Fuzzy and Adaptative Approches*, éd. par White (David A.) et Sofge (Donald A.). – Van Nostrand Reinhold, 1992.

[BMA93] Balaniuk (Remis), Mazer (Emmanuel) et Amy (Bernard). – *La Robotique Connexionniste : Enquête documentaire sur l'utilisation des algorithmes connexionnistes en robotique.* – Rapport technique n° 91/451, DRET - Direction des Recherches, Études et Techniques (Délégation Générale pour l'Armement), octobre 1993.

[BMB95] Balaniuk (Remis), Mazer (Emmanuel) et Bessière (Pierre). – Fast direct and inverse model acquisition by function decomposition. *In : IEEE International Conference on Robotics and Automation.* – Nagoya, Japan, 1995.

[Cox79] Cox (R. T.). – *Of inference and inquiry, an essay in inductive logic; in The maximum entropy formalism.* – M.I.T. Press, 1979.

[Ded95] Dedieu (Eric). – *La représentation contingente : Vers une reconciliation des approches fonctionnelles et structurelles de la robotique autonome.* – Thèse de PhD, Institut National Polytechnique de Grenoble, 1995.

[Dor95] Dornaika (Fadi). – *Contributions à l'integration vision/robotique : calibrage, localisation et asservissement.* – Thèse de PhD, Institut National Polytechnique de Grenoble, 1995.

[Fun89] Funahashi (K.). – On the approximate realization of continuous mappings by neural networks. *Neural Networks*, vol. 2, 1989, pp. 183 – 191.

[HDBE95] Horaud (R.), Dornaika (F.), Bard (C.) et Espiau (B.). – *Visually Guided Object Grasping.* – Rapport technique, INRIA, March 1995. Submitted to *IEEE Trans. on Robotics & Automation.*

[HDBL95] Horaud (R.), Dornaika (F.), Bard (C.) et Laugier (C.). – Integrating grasp planning and visual servoing for automatic grasping. *In : Proceedings of the Fourth International Symposium on Experimental Robotics*, pp. 49–54. – Stanford, California, June/July 1995.

[HN89] Hecht-Nielsen (R.). – Theory of the backpropagation network. *In : Proc. of IJCNN'89.* pp. 593 – 611. – New York, 1989.

[Jay95] Jaynes (E.T.). – Probability theory - the logic of science. – Draft of a forthcomming book; personnal communication, 1995.

[JHB+95] Juditsky (A.), Hjalmarsson (H.), Benveniste (A.), Delyon (B.), Ljung (L.), Sjöberg (J.) et Zhang (Q.). – *Nonlinear black-Box Modeling in System Identification: Mathematical Foundations.* – Rapport technique n° LiTH-ISY-R-1743, University of Linköping, Sweden, 1995.

[JR91] Jordan (Michael I.) et Rumelhart (David E.). – *Forward models: Supervised learning with a distal teacher.* – Rapport technique, MIT, Center of Cognitive Science, 1991. Occasional Paper 40.

[Kha90] Khanna (Tarun). – *Foundations of Neural Networks.* – Addison-Wesley, 1990.

[Koh82] Kohonen (Teuvo). – Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, vol. 43, 1982, pp. 59–69.

[LBM94] Lebeltel (Olivier), Bessière (Pierre) et Mazer (Emmanuel). – La poubelle lumineuse : Acquisition probabiliste d'organisation sensorimotrice. *In : NSI94 (Neurosciences et Sciences de l'Ingénieur).* – Chamonix, 1994.

[Lju87] Ljung (Lennart). – *System Identification : theory for the user.* – P T R Prentice Hall, 1987.

[Lor86] Lorentz (G.G). – *Approximation of Functions.* – New York, Chelsea Publishing Co, 1986.

[Mac92] MacKay (David J. C.). – *Bayesian Methods for Adaptative Models.* – Pasadena, California, Thèse de PhD, California Institute of Technology, 1992.

[McK91] McKerrow (Phillip J.). – *Introduction to Robotics.* – Addison-Wesley, 1991.

[MHJ92] Moore (Andrew W.), Hill (Daniel J.) et Jonhson (Michael P.). – An empirical investigation of brute force to choose features, smoothers and function approximators. *In : Computational Learning Theory and Natural Learning Systems.* – Madison Wisconsin, august 1992.

[PG89] Poggio (Tomaso) et Girosi (Federico). – *A theory of Networks for Approximation and Learning.* – Rapport technique n° A.I. Memo no. 1140, Massachusetts Institute of Technology, 1989.

[Pre92] Press (William H. et al.). – *Numerical Recipes in C.* – Cambridge University Press, 1992.

[PW93] Plutowski (Mark) et White (Halbert). – Selecting concise training sets from clean data. *IEEE Transactions on Neural Networks*, vol. 4, n° 2, march 1993.

[Ric91] Richalet (Jacques). – *Pratique de l'identification.* – Hermes Paris, 1991.

[RMG86] Rumelhart (D.), McClelland (J.) et Group (PDP Research) (édité par). – *Parallel Distributed Processing, Explorations in the Microstructure of Cognition.* – MIT Press, 1986.

[RMS89] Ritter (Helge J.), Martinez (Thomas M.) et Schulten (Klaus J.). – Topology-conserving maps for learning visuo-motor-coordination. *Neural Networks*, vol. 2, 1989, pp. 159–168.

[SA94] Schaal (Stefan) et Atkeson (Christopher G.). – Robot juggling: Implementation of memory-based learning. *IEEE Control Systems*, february 1994, pp. 57–71.

[Sto82] Stone (C.J.). – Optimal global rates of convergence for nonparametric regression. *Ann. Stat.*, vol. 10, 1982, pp. 1040–1053.

[Sto85] Stone (C.J.). – Additive regression and other nonparametric models. *Ann. Stat.*, vol. 13, 1985, pp. 689–705.

[SV89] Spong (Mark W.) et Vidyasagar. – *Robot Dynamics Control.* – Jonh Wiley & Sons, 1989.

[SZL+95] Sjöberg (J.), Zhang (Q.), Ljung (L.), Benveniste (A.), Deylon (B.), Glorennec (P-Y), Hjalmarsson (H.) et Juditsky (A.). – *A theory of Networks for Approximation and Learning.* – Rapport technique n° LiTH-ISY-R-1742, University of Linköping, Sweden, 1995.

[ZB92] Zhang (Qinghua) et Benveniste (Albert). – Wavelet networks. *IEEE Transactions on Neural Networks*, vol. 3, n° 6, november 1992, pp. 889–898.