THÈSE de DOCTORAT

de

l' Université Pierre & Marie Curie

École Doctorale de Sciences Mécanique, Acoustique, Électronique et Robotique de Paris

Spécialité

Mécanique - Robotique

presentée par

Camille SALAÜN

APPRENTISSAGE DE MODÈLES POUR LA COMMANDE DE LA MOBILITÉ INTERNE EN ROBOTIQUE

Résumé

Soutenue le 30 août 2010

JURY

M. P. Bidaud	Professeur de l'Université Pierre & Marie Curie	Examinateur
M. M. Butz	Maître de conférences de l'Université de Würzburg, HDR	Rapporteur
M. J.-Y. Fourquet	Professeur de l'École Nationale d'Ingénieurs de Tarbes	Rapporteur
M. V. Padois	Maître de conférences de l'Université Pierre & Marie Curie	Co-directeur de thèse
M. O. Sigaud	Professeur de l'Université Pierre & Marie Curie	Directeur de thèse
M. P. Souères	Directeur de recherche au Laboratoire d'Analyse et d'Architecture des Systèmes – CNRS	Examinateur

Remerciements

Je remercie Vincent Padois et Olivier Sigaud pour la confiance et l'aide qu'ils m'ont accordée ainsi qu'aux critiques nécessaires à la structuration et à la réalisation de ce travail. Je les remercie pour leur disponibilité et leur volonté d'aller de l'avant.

Je remercie les rapporteurs de ce travail, Martin V. Butz et Jean-Yves Fourquet qui ont fait un travail de relecture permettant de faire ressortir les incohérences et les erreurs qui auraient pu rendre ce texte plus difficile à lire qu'il ne l'est aujourd'hui. Ce n'est qu'avec les yeux des autres que l'on peut bien voir ses défauts. Je remercie Philippe Souères pour les remarques et questions pertinentes qu'il a pu me poser qui ont pu mettre l'accent sur les points essentiels de cette thèse et de ses perspectives. Je remercie Philippe Bidaud en qualité de président du jury, en qualité de directeur de laboratoire mais également comme professeur ayant contribué à ma formation de roboticien.

Je remercie Emmanuel Guigon et Alain Micaelli pour les discussions scientifiques que nous avons pu avoir et qui ont contribué à donner un cadre de travail à cette thèse.

Je remercie les stagiaires, Renaud Durlin, Céline Boudier, Laure Baffie, Pierre Jarrault, Paul Tonelli, Charles Clerq, Céline Hamon, Jérémie Rubinsztajn et Guillaume Sicard qui ont contribué à ce travail de près ou de loin.

Je remercie les responsables et personnes qui font vivre le projet Robotcub sans qui la partie expérimentale de mon travail aurait été moindre.

Je remercie mes enseignants, devenus collègues de travail, pour leur volonté (que je partage) de vouloir transmettre leurs connaissances à des étudiants désireux d'apprendre.

Je remercie les anciens doctorants du Laboratoire de Robotique de Paris qui ont su répondre à mes questions avec précision et dans la bonne humeur.

Je remercie Nathanaël et Éric avec qui l'aventure a commencée et avec qui j'espère la continuer. Je remercie également Sébastien avec qui j'ai eu le plaisir de travailler, pour ses discussions et son perfectionnisme qui font de lui un grand chercheur. Je remercie tous les doctorants du laboratoire, devenus amis, qui ont contribué à ce que ce travail ne soit pas qu'un travail mais une expérience humaine riche en émotion.

Je remercie ma famille qui a su me soutenir dans mes périodes d'indécision et avec qui il est possible de discuter.

Je remercie Julie-Marie pour son soutien, sa joie et notre envie partagée d'avancer ensemble.

Tout obstacle renforce la détermination.
Celui qui s'est fixé un but n'en change pas.
—Léonard de Vinci

Résumé

La robotique de service est un domaine émergent où il est nécessaire de commander des robots en interaction forte avec leur environnement. Ce travail présente une méthode adaptative de commande combinant de l'apprentissage de modèles de la mécanique à de la commande dans l'espace opérationnel de robots redondants. L'apprentissage des modèles cinématiques est obtenu soit par dérivation de modèles géométriques appris, soit par apprentissage direct. Ces modèles cinématiques, également appelés matrices Jacobiennes, peuvent être utilisés dans le calcul de pseudo-inverses ou de projecteurs pour la commande de robots. Cette combinaison de méthodes permet d'obtenir un contrôleur qui s'adapte à la géométrie du robot commandé. En utilisant les mêmes algorithmes d'apprentissage, il est possible d'apprendre un modèle dynamique inverse du robot contrôlé de manière à le commander en couple plutôt qu'en vitesse, l'avantage étant de pouvoir s'adapter aux modifications dynamiques qui s'appliquent sur le robot comme par exemple l'application d'une force extérieure ou l'ajout d'un poids.

Des expériences en simulation menées dans le cadre de cette thèse montrent comment réaliser plusieurs tâches hiérarchiques ou comment s'adapter à des perturbations avec des modèles appris. Des expériences sur le robot iCub ont également été menées afin de rendre compte de la plausibilité de l'approche proposée sur un système réel.

mots clés : apprentissage automatique, mobilité interne, cinématique inverse, dynamique inverse, robotique humanoïde, adaptation

Cette thèse a été préparée à l'*Institut de Systèmes Intelligents et de Robotique*, dépendant de l'*École Doctorale de Sciences Mécanique, Acoustique, Électronique et Robotique de Paris* - 4, place Jussieu 75005 Paris France.

Table des matières

Introduction

La robotique évolue d'un monde industriel, structuré, bien défini, vers la robotique de service, plus difficile à modéliser car les missions ne sont pas forcément connues à l'avance. Contrairement à la robotique industrielle où les robots ont été conçus pour une tâche précise, la robotique de service s'intéresse aux robots polyvalents en interaction forte avec leur environnement et les humains qui les entourent. Parallèlement à cette complexité, les robots deviennent de plus en plus sophistiqués tant par le nombre de capteurs que par le nombre de degrés de libertés qui les composent.

Les applications robotiques sont souvent séparées en deux niveaux, la mission et la tâche. Une mission peut être décomposée en un ensemble de tâches qui doivent être planifiées puis exécutées séquentiellement pour que la mission réussisse. Le niveau de description d'une mission ne fait pas apparaître le détail de la réalisation de chacune des tâches qui la composent. Nous définissons une tâche comme un ensemble de buts à atteindre, totalement ou partiellement, simultanément ou non, avec un ou plusieurs organes terminaux, en effectuant une coordination adéquate de mouvements.

Afin de décrire les différentes méthodes de commande, nous définissons deux espaces : l'espace de la tâche, aussi appelé espace opérationnel, dans lequel sont décrits facilement les buts à atteindre ainsi que les contraintes appliquées au système et l'espace articulaire dans lequel sont décrits les mouvement des liaisons du robot.

La réalisation d'une tâche peut être vue comme un problème de commande, supervisé par un planificateur de mouvement [Latombe, 1991; Laumond, 1998], qui consiste à créer un chemin dans l'espace des configurations depuis un état initial vers un état final. La trajectoire ainsi générée garantit que le robot peut exécuter le mouvement sans rencontrer d'obstacle. Cependant, pour planifier dans l'espace des configurations, malgré les récentes avancées dans le domaine [LaValle, 2006; Kavraki et Latombe, 1998], il est nécessaire de connaître parfaitement l'environnement. Ces méthodes sont donc difficilement applicables directement dans le cadre de la robotique de service. On pourrait éventuellement penser à utiliser des méthodes de commande optimale qui optimisent un critère afin de générer une trajectoire pour atteindre un but. Mais ces méthodes prennent difficilement en compte les incertitudes et les effets non modélisés de l'environnement et reste limitées à des problèmes de petites dimensions. Contrairement à ces méthodes qui nécessitent d'avoir une vision globale du problème, les méthodes telle que *Resolved Motion Rate Control* (RMRC) sont réactives et résolvent le problème localement à chaque pas de temps. D'autres méthodes un peu plus sophistiquées telle que *Sequential Quadratic Programming* (SQP), peuvent être utilisées dans des commandes de type prédictives afin de minimiser localement une fonction de coût sous certaines contraintes. Ces méthodes sont en plein essor pour la commande de robots humanoïdes [Kajita *et al.*, 2003;

Wieber, 2006; Kanoun *et al.*, 2009].

Nous nous intéressons dans cette thèse à la réalisation de tâches hiérarchiques et simultanées c'est-à-dire à la commande de la mobilité interne des robots redondants. Un robot est dit redondant relativement à une tâche s'il possède plus de degrés de liberté que ceux nécessaires à la réalisation de cette tâche. Un formalisme mathématique, proposé par Liégeois [1977] et inspiré de *Resolved Motion Rate Control* apporte une solution à ces problèmes en utilisant ces degrés de liberté surnuméraires afin d'accomplir des tâches supplémentaires sans perturber la tâche principale.

Identification ou apprentissage ? Les contrôleurs développés doivent être soit extrêmement robustes aux incertitudes du modèle du robot ou de l'environnement, soit être adaptatifs, c'est-à-dire capables de créer un modèle en ligne. Toutes les lois de commande citées précédemment utilisent des modèles géométriques, cinématiques ou dynamiques. De bonnes connaissances en mécanique ainsi qu'un long travail de modélisation sont nécessaires pour créer ces modèles, particulièrement dans le cas de robots complexes comme les robots humanoïdes. S'ils sont seulement extraits des équations de la physique ou d'un modèle informatique, ces modèles sont appelés *modèles déduits*. Ils ne peuvent représenter correctement les robots car pendant la conception, la réalisation ou l'assemblage, des différences peuvent apparaître au niveau des frictions, des poids, des longueurs ou des jeux dans les liaisons de chaque robot.

Afin de pouvoir tout de même commander des robots avec des modèles, certains ingénieurs raffinent ces modèles déduits en utilisant des méthodes d'identification paramétrique [Ljung, 1986]. Ces modèles créés, appelées *modèles induits*, sont générés à l'aide d'un grand nombre de données issues du système réel. Ces méthodes d'identification sont souvent utilisées hors ligne mais peuvent être également utilisées de manière incrémentale dans une loi de commande de manière à s'adapter aux perturbations qui pourraient apparaître durant la mission. Cette approche s'appelle *commande adaptative* [Sastry et Bodson, 1989; Landau *et al.*, 1998; Siciliano *et al.*, 2008]. Cependant, l'identification paramétrique introduit un biais dans le modèle induit car le modèle possède une structure figée et l'ajustement des paramètres du modèle peut ne pas être suffisant pour atteindre la précision requise. Des modèles non-paramétriques peuvent résoudre ces problèmes car ils n'ont pas de structure a priori et dépendent uniquement des données fournies. Ils peuvent être obtenus en utilisant des méthodes d'apprentissage automatique (*machine learning* en anglais) en évitant ainsi d'introduire une connaissance a priori dans les modèles.

L'apprentissage automatique en robotique permet également de doter les robots d'une capacité d'adaptation aux changements de l'environnement ou du robot lui-même.

But de la thèse Le contexte robotique décrit ci-dessus est plutôt éloigné des projets contemporains de robotique industrielle. Dans cette thèse, nous ne prétendons pas fournir une méthode prête à l'emploi permettant de résoudre les problèmes énoncés. Nous tentons de poser les bases d'une méthode permettant de résoudre certains aspects de ces problèmes. Plus précisément, nous essayons de créer une loi de commande utilisant des modèles appris afin de commander la mobilité interne des robots redondants dans le cas où les modèles ne sont pas connus à l'avance ou peuvent évoluer dans le temps. Afin de réaliser cette loi de commande, il est nécessaire de s'assurer de deux choses.

- Comme la cinématique des robots est suffisante pour commander la mobilité interne, nos architecture considère séparément la cinématique et la dynamique des robots. Cela nous permet d'utiliser notre architecture sur des robots commandés en vitesse comme de nombreux robots humanoïdes.
- Tous les modèles de la mécanique appris doivent l'être en chaque état du système de manière à pouvoir commander le robot d'une infinité de façon.

1 Commande de la mobilité interne de robots dans l'espace opérationnel

Cette section présente le formalisme mathématique utilisé dans cette thèse pour représenter les systèmes de corps rigides et une méthode classique de commande basée sur l'inversion des modèles cinématiques et dynamique.

1.1 Mécanique des corps rigides

La description des corps rigides utilisés dans cette thèse est largement inspirée de la thèse de Duindam [2006], basée sur les travaux de Selig [2005] et Stramigioli et Bruyninckx [2001].

1.1.1 Configuration des systèmes poly-articulés

Les systèmes poly-articulés sont définis par un nombre fini de corps rigidement liés entre eux par des liaisons cinématiques. Une liaison E_i^j est dite idéale si elle ne fait pas intervenir de frottement ou de jeu dans la relation cinématique entre deux corps B_i et B_j.

Un corps rigide B_i peut être décrit par un repère Ψ_i qui lui est attaché. La pose d'un repère est composée de la position de l'origine du repère ainsi que de l'orientation définie par une base orthonormale.

On appelle degrés de liberté, les déplacement indépendants qui spécifient la pose d'un corps ; trois pour la position et trois pour l'orientation. De la même façon, on appelle degrés de liberté les mouvements possibles et indépendants dans une chaîne poly-articulée.

Les n paramètres indépendants, notés $\boldsymbol{q}$, qui décrivent la configuration d'une chaîne poly-articulée sont appelés les coordonnées généralisées et décrivent l'espace articulaire.

Tandis que ces paramètres décrivent facilement les liaisons de la chaîne cinématique, il est difficile de les utiliser pour décrire une tâche complexe. On utilise pour cela l'espace opérationnel ou espace de la tâche.

La tâche unitaire considérée consiste à positionner un repère dans l'espace relativement à un autre, i.e., définir la pose d'un corps.

L'espace opérationnel est décrit par le repère lié aux organes terminaux du robot, i.e., les outils utilisés pour interagir avec l'environnement. Tandis que l'espace opérationnel et l'espace de la tâche expriment le même espace, on utilisera le terme *opérationnel* ou *de la tâche* pour décrire respectivement l'organe terminal où la tâche. On considère une tâche pour chaque organe terminal. Les m degrés de liberté d'un organe terminal, notés $\boldsymbol{\xi}$, varient indépendamment pour accomplir une tâche. Ils sont dépendants de la configuration du système. Il est courant de définir la pose d'un organe terminal avec six paramètres de manière à ce qu'il puisse être compatible avec la tâche qui lui correspond.

Tandis qu'une tâche est, par définition, facilement décrite dans l'espace opérationnel, elle est difficilement exprimée dans l'espace articulaire. Par opposition, la commande est généralement exprimée dans l'espace articulaire. Il est donc utile de connaître le modèle géométrique inverse qui lie l'espace opérationnel à l'espace articulaire.

Selon les valeurs de m et n, différents cas de figure se présentent à nous :

Un robot est qualifié de complètement contraint s'il possède autant de degrés de liberté que ceux nécessaires à la réalisation de ces tâches, soit $m = n$. Il est qualifié de sur-contraint si $m > n$, c'est-à-dire s'il n'a pas suffisamment de degrés de liberté actionnés pour réaliser ses tâches. Le robot est aussi qualifié de sous-actionné et ne peut réaliser son objectif que partiellement. Le cas sous-contraint, $m < n$, correspond au cas où le système possède plus de degrés de libertés que ceux nécessaires à la réalisation de ses tâches. Le robot est dit redondant vis-à-vis de cette tâche. (voir Figure 1). Dans ce cas, il existe une infinité d'inverses qui permettent au robot de réaliser complètement ses tâches. Il possède en outre une mobilité interne de degré $n-m$ qui peut être exploitée au niveau cinématique afin de réaliser d'autres tâches.

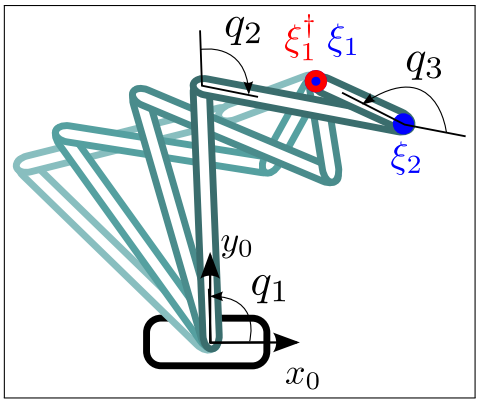


FIGURE 1 – Vue stroboscopique d'un robot planaire à trois degrés de liberté. Une infinité de configurations permettent au robot de positionner son organe terminal (disque bleu) sur la position désirée (anneau rouge). Le robot peut continuer à bouger tout en maintenant cette position.

1.1.2 Cinématique des systèmes poly-articulés

Considérant uniquement la position et non l'orientation d'un organe terminal, la dérivation du modèle géométrique par rapport au temps s'écrit :

$$\frac{d\boldsymbol{\xi}}{dt} = \frac{\partial \mathcal{FK}(\boldsymbol{q})}{\partial \boldsymbol{q}} \frac{d\boldsymbol{q}}{dt}. \tag{1}$$

ou bien

$$\boldsymbol{\nu}_{0,i}^{0} = J_{0,i}\left(\boldsymbol{q}\right)\dot{\boldsymbol{q}} \tag{2}$$

où $\boldsymbol{\nu}_{0,i}^{0}$ est la vitesse linéaire du point $\boldsymbol{\xi}$. La matrice $J_{0,i}\left(\boldsymbol{q}\right) = \partial\mathcal{FK}\left(\boldsymbol{q}\right)/\partial\boldsymbol{q}$, de taille $m \times n$, est appelée matrice Jacobienne du corps B_i exprimée dans le repère Ψ_0.

On définit le noyau d'une matrice Jacobienne comme étant l'ensemble des vitesses articulaires qui ne produisent pas de vitesse sur l'organe terminal.

$$\ker(J) = \{\dot{\boldsymbol{q}} \in \mathbb{R}^n : J\dot{\boldsymbol{q}} = 0\} \tag{3}$$

1.2 Commande basée modèle

Un robot étant commandé dans l'espace articulaire, il est nécessaire de connaître le modèle liant l'espace opérationnel à l'espace articulaire, i.e., de connaître les actions requises pour réaliser une tâche donnée.

Une commande classique basée modèle, illustrée Figure 2, est divisée en trois parties.

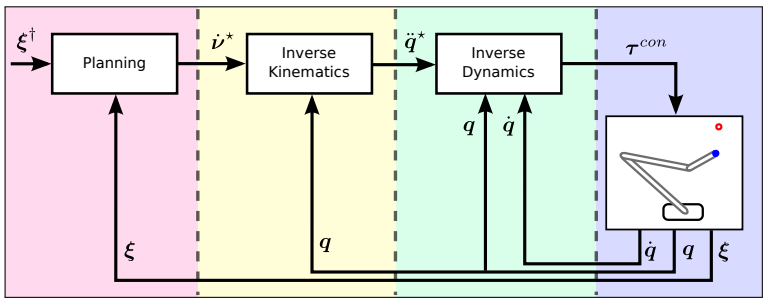


FIGURE 2 – Boucle de commande classique utilisant de la planification, un modèle cinématique inverse et un modèle dynamique inverse.

Une première partie appelée planification consiste à spécifier une trajectoire qui doit être suivie par un des organes terminaux du robot pour réaliser sa tache associée $\boldsymbol{\xi}^{\dagger}$ afin de la transformer en accélération opérationnelle. Ce type de contrôleur, proportionnel à l'erreur en position s'écrit

$$\dot{\boldsymbol{\nu}}^{\star} = K_p\left(\boldsymbol{\xi}^{\dagger} - \boldsymbol{\xi}\right), \tag{4}$$

où K_p est une matrice de gains et $\boldsymbol{\xi}^{\dagger}$ est la position désirée. Ce contrôleur peut être vu comme un ressort de raideur K_p entre le but et l'organe terminal.

La seconde étape de la commande transforme cette accélération opérationnelle désirée en accélérations articulaires désirées à l'aide du modèle cinématique inverse

$$\boldsymbol{q} = \mathcal{IK}\left(\boldsymbol{\xi}\right). \tag{5}$$

Dans le cas entièrement contraint, $m = n$, le modèle cinématique inverse s'écrit

$$\dot{\boldsymbol{q}} = {J_i}^{-1}(\boldsymbol{q})\,\boldsymbol{\nu}_{0,i}. \tag{6}$$

Dans le cas sous-contraint, $m < n$, la matrice Jacobienne n'est pas carrée et il existe une infinité d'inverses possibles notées $J^{\sharp}(\boldsymbol{q})$ (voir Ben Israel et Greville [2003] pour plus de détails).

$$\dot{\boldsymbol{q}} = {J_i}^{\sharp}(\boldsymbol{q})\,\boldsymbol{\nu}_{0,i}. \tag{7}$$

Parmi ces inverses, la pseudo-inverse pondérée [Doty *et al.*, 1993; Park *et al.*, 2001] fournit une solution de norme minimale au sens des moindres carrés pondérés. Si $\operatorname{rang}(J(\boldsymbol{q})) = m$, elle peut être notée

$${J_i}^{W+}(\boldsymbol{q}) = W^{-1}{J_i}^{T}(\boldsymbol{q})\left[J_i(\boldsymbol{q})\,W^{-1}{J_i}^{T}(\boldsymbol{q})\right]^{-1}, \tag{8}$$

où W est une matrice symétrique définie positive. La pseudo-inverse de Moore-Penrose ${J_i}^{+}(\boldsymbol{q})$ correspond au cas où $W = I_n$.

Whitney [1969] nomme *Resolved Motion Rate Control* (RMRC) la méthode qui consiste à utiliser le modèle cinématique inverse pour contrôler l'organe terminal d'un robot. Étant donnée une vitesse souhaitée pour l'organe terminal $\boldsymbol{\nu}^{\star}_{0,i}$, il est possible de calculer la vitesse articulaire désirée grâce à l'équation suivante

$$\dot{\boldsymbol{q}}^{\star} = {J_i}^{+}(\boldsymbol{q})\,\boldsymbol{\nu}^{\star}_{0,i}. \tag{9}$$

La mobilité interne des robots redondants peut être commandée au niveau cinématique grâce à un projecteur sur le noyau de la matrice Jacobienne. Cette mobilité interne peut être utilisée pour réaliser des tâches supplémentaires comme minimiser l'énergie dépensée, limiter les couples, éviter des obstacles ou des butées articulaires.

Un projecteur d'une matrice Jacobienne permet de commander la mobilité interne sans générer de vitesse sur l'organe terminal correspondant. Il s'écrit usuellement

$$P_J(\boldsymbol{q}) = \left(I_n - J^{\sharp}(\boldsymbol{q})\,J(\boldsymbol{q})\right). \tag{10}$$

Maciejewski et Klein [1985] propose de l'utiliser de la manière suivante

$$\dot{\boldsymbol{q}}^{\star} = J_a^{+}\boldsymbol{\nu}^{\star}_{0,a} + (J_b P_{J_a})^{+}\left(\boldsymbol{\nu}^{\star}_{0,b} - J_b J_a^{+}\boldsymbol{\nu}^{\star}_{0,a}\right). \tag{11}$$

où J_a et J_b sont les matrices Jacobiennes de deux différentes tâches, P_{J_a} le projecteur orthogonal de la matrice J_a et $\boldsymbol{\nu}^{\star}_{0,a}$ et $\boldsymbol{\nu}^{\star}_{0,b}$, les vitesses désirées correspondant aux tâches à accomplir.

Dans le cas compatible, les deux tâches sont complètement réalisées tandis que dans le cas incompatible, seule la tâche dite principale est accomplie ; la tâche secondaire est réalisée partiellement avec une erreur minimisée au sens des moindres carrés.

La troisième étape utilise la dynamique inverse pour calculer les couples. Les équations de la dynamique d'une chaîne poly-articulée de corps rigides dans le formalisme de Newton-Euler ou de Lagrange s'écrivent :

$$\boldsymbol{\tau}^{con} = A(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \boldsymbol{n}(\boldsymbol{q},\dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\epsilon}(\boldsymbol{q},\dot{\boldsymbol{q}}) - \boldsymbol{\tau}^{ext} \tag{12}$$

où $A(\boldsymbol{q})$, $\boldsymbol{n}(\boldsymbol{q},\dot{\boldsymbol{q}})$, $\boldsymbol{g}(\boldsymbol{q})$, $\boldsymbol{\epsilon}(\boldsymbol{q},\dot{\boldsymbol{q}})$ et $\boldsymbol{\tau}^{ext}$ sont respectivement la matrice d'inertie du système, les vecteurs des effets de Coriolis et centrifuge, le vecteur des effets gravitationnels, le vecteur des effets non-modélisés ainsi que les couples résultants des forces extérieures appliquées au système. $\boldsymbol{\tau}^{con}$ est le vecteur des couples articulaires de contrôle appliqués au robot.

Par simplicité d'écriture, l'Équation (12) est notée

$$\boldsymbol{\tau}^{con} = \mathcal{ID}\left(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}},\boldsymbol{\tau}^{ext}\right). \tag{13}$$

La simulation dynamique peut être résumée par l'Algorithme 1

Algorithme 1 Boucle de simulation

$\dot{\boldsymbol{\nu}}^{\star} = K_p\left(\boldsymbol{\xi}^{\star} - \boldsymbol{\xi}\right)$	*vitesse opérationnelle désirée*
$\ddot{\boldsymbol{q}}^{\star} = J^{+}(\boldsymbol{q})\left(\dot{\boldsymbol{\nu}}^{\star} - \dot{J}(\boldsymbol{q})\,\dot{\boldsymbol{q}}\right)$	*vitesses articulaires désirées*
$\boldsymbol{\tau}^{con} = \mathcal{ID}\left(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}^{\star},\boldsymbol{\tau}^{ext}\right)$	*couples désirés*
$\ddot{\boldsymbol{q}} \leftarrow \mathcal{D}\left(\boldsymbol{q},\dot{\boldsymbol{q}},\boldsymbol{\tau}^{con},\boldsymbol{\tau}^{ext}\right)$	*intégration du simulateur*

Pour plus de détails, le lecteur peut se référer à Khalil et Dombre [2002].

2 Apprentissage en robotique

Cette section présente les méthodes d'apprentissage utilisées en robotique. Dans une première section, nous présentons deux algorithmes d'apprentissage supervisé qui, dans un cadre robotique, s'apparentent à estimer des fonctions non-linéaires.

Dans une seconde partie, nous présentons diverses méthodes d'apprentissage supervisé utilisées dans un cadre robotique. Nous expliquons les contraintes imposées par certaines architectures de commande.

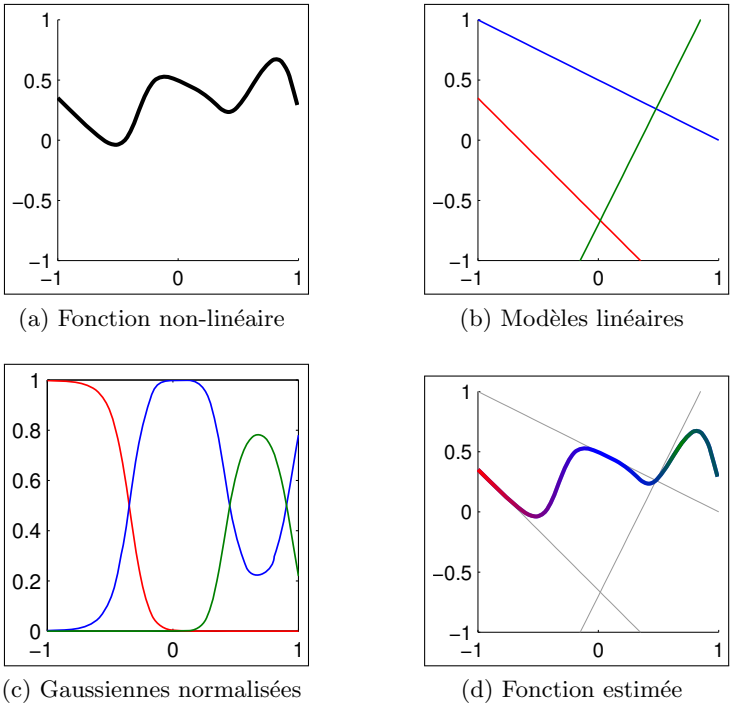
2.1 Apprentissage supervisé

L'apprentissage supervisé est une méthode utilisée lorsqu'il existe un superviseur qui fournit un couple (entrée, sortie) qui peut être utilisé pour apprendre le modèle entre les données d'entraînement.

2.1.1 *Locally Weighted Projection Regression*

Locally Weighted Projection Regression (LWPR) [Vijayakumar et Schaal, 2000] est une méthode d'estimation de fonctions non-linéaires par des sommes

de modèles linéaires pondérés par des gaussiennes (voir Figure 3). Cette méthode a la particularité d'être incrémentale afin de ne pas conserver les données et projette les entrées sur des sous-espaces pertinents afin de réduire les dimensions d'apprentissage.



(a) Fonction non-linéaire (b) Modèles linéaires

(c) Gaussiennes normalisées (d) Fonction estimée

FIGURE 3 – Estimation d'une fonction non-linéaire (a) par une somme de modèles linéaires (b) pondérés par des gaussiennes normalisées (c). L'approximation finale (d) est proche de la fonction initiale.

LWPR utilise des fonctions gaussiennes comme champs récepteurs (région de validité)

$$\phi_i(\boldsymbol{x}) = e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{c}_i)^T D_i(\boldsymbol{x}-\boldsymbol{c}_i)}. \tag{14}$$

où D_i, positif, délimite la région de validité de la gaussienne.

Chaque champ récepteur possède un modèle linéaire β_i, utilisé pour prédire une sortie y_i relativement à l'entrée $\boldsymbol{x}$

$$\hat{y}_i(\boldsymbol{x}) = \beta_i \boldsymbol{x}. \tag{15}$$

L'approximation globale réalisée par LWPR est donc la somme de tous les modèles linéaires, pondérés par leur gaussienne respective

$$\hat{y}(\boldsymbol{x}) = \frac{1}{\Phi(\boldsymbol{x})} \sum_{i=1}^{n_n} \phi_i(\boldsymbol{x})\, \hat{y}_i(\boldsymbol{x}) \tag{16}$$

où $\Phi(\boldsymbol{x}) = \sum_{i=1}^{n_n} \phi_i(\boldsymbol{x})$ et n_n est le nombre de champs récepteurs.

L'algorithme LWPR consiste à faire évoluer les gaussiennnes ainsi que les modèles linéaires qui sont adaptés à chaque pas de temps en fonction de l'erreur entre la sortie réelle et celle prédite.

Le lecteur pourra se référer à Schaal *et al.* [2002] ou Vijayakumar et Schaal [2000] pour une présentation plus en profondeur de l'algorithme.

Il est également important de noter qu'il est possible d'obtenir la dérivée du modèle appris par rapport aux entrées.

2.1.2 *Extended Classifier System for Function approximation*

L'algorithme *eXtended Classifier System for Function approximation* (XCSF) est une autre méthode d'estimation de fonctions issue de la famille des systèmes de classeurs proposées par Holland [1975] qui combine les mécanismes de l'apprentissage par renforcement [Sutton et Barto, 1998] et des algorithmes génétiques [Goldberg, 1989].

XCSF [Wilson, 2001] est un système de classeurs où chaque classeur possède un *condition space*, nommé $\mathcal{Z}$, qui spécifie son domaine d'application, i.e., le contexte dans lequel une loi peut s'appliquer. Dans cette thèse, nous nous intéressons au cas où cet espace est pavé de gaussiennes, équivalentes aux champs récepteurs dans LWPR, qui définissent un domaine d'application de la manière suivante

$$\phi_i(\boldsymbol{z}) = e^{-\frac{1}{2}(\boldsymbol{z}-\boldsymbol{c}_i)^T D_i(\boldsymbol{z}-\boldsymbol{c}_i)} \tag{17}$$

où c_i et D_i sont respectivement le centre de la gaussienne i et la métrique, inverse du carré de la variance de la gaussienne i. L'entrée $\boldsymbol{z} \in \mathcal{Z}$ du *condition space* définit l'espace où la gaussienne a une influence.

Chaque classeur possède également un *prediction input space*, nommé $\mathcal{X}$, qui correspond à l'entrée utilisée pour calculer le modèle linéaire β_i par une régression. Chaque modèle linéaire peut alors être utilisé pour prédire une sortie $\boldsymbol{y}_i$ correspondant à l'entrée $\boldsymbol{x}$

$$\hat{\boldsymbol{y}}_i(\boldsymbol{x}) = \beta_i \boldsymbol{x} + \beta_0. \tag{18}$$

Les classeurs de XCSF forment une population P qui partitionne l'espace d'entrée en un ensemble de modèles entrelacés. XCSF sélectionne parmi la population un sous-ensemble de ces modèles qui sont fiables pour l'entrée $\boldsymbol{z}$. Le modèle linéaire de chaque classeur est pondéré par sa *fitness*. Les modèles pondérés sont ensuite sommés afin d'estimer la fonction non-linéaire apprise

$$\hat{\boldsymbol{y}}(\boldsymbol{x}, \boldsymbol{z}) = \frac{1}{F(\boldsymbol{z})} \sum_{i=1}^{n_M} F_i(\boldsymbol{z})\, \hat{\boldsymbol{y}}_i(\boldsymbol{x}) \tag{19}$$

où $F(\boldsymbol{z}) = \sum_{i=1}^{n_M} F_i(\boldsymbol{z})$ et n_M est le nombre de classeurs sélectionnés.

La mise à jour du modèle linéaire β_i de chaque classeur est réalisée avec des algorithmes de régression de type moindre carrés récursifs. Les paramètres c_i et D_i des classeurs sont mis à jour à l'aide d'un algorithme génétique. Une description complète de l'algorithme peut être trouvée dans Butz *et al.* [2004] ou dans Butz et Herbort [2008].

2.2 Apprentissage de la mécanique

L'apprentissage supervisé permet d'estimer des fonctions non-linéaires et peut donc être utilisé pour apprendre des modèles de la mécanique. Ainsi, les données ne sont que dépendantes du système mécanique. Cette section détaille comment des modèles cinématiques et dynamiques peuvent être appris et utilisés pour commander des systèmes robotiques.

2.2.1 Apprentissage de la cinématique inverse

Nous avons vu en Section 1 qu'il était nécessaire, pour commander un système dans l'espace opérationnel, de connaître sa cinématique inverse. Il parait donc naturel d'apprendre directement ce modèle inverse. Pour ceci, il est nécessaire de fournir le couple (vitesse organe terminal, vitesses articulaires) comme couple (entrées, sorties) à la méthode d'apprentissage.

Dans le cas de robots non-redondants, cette approche ne soulève aucune difficulté car le modèle cinématique inverse est unique, voir sur-contraint. Kohonen [2001]; Ritter *et al.* [1989]; Martinetz *et al.* [1990]; Walter et Schulten [1993] apprennent des modèles cinématiques inverses de robots à trois degrés de liberté à l'aide de deux caméras.

Cette approche soulève quelques problèmes dans le cas redondant. L'apprentissage de modèles comme nous l'avons décrit ne permet d'estimer que des fonctions injectives or, comme nous l'avons vu en Section 1, il existe une infinité de modèles cinématiques inverses d'un robot redondant. Ce modèle n'est donc pas injectif. En utilisant cette méthode, il sera possible d'apprendre une seule solution parmi cette infinité. Les approches proposées dans la littérature apprennent des modèles de robots redondants en ajoutant des contraintes de manière à transformer le problème sous-contraint en problème entièrement contraint. Pour exemple, D'Souza *et al.* [2001] décrit l'apprentissage d'un modèle cinématique inverse avec LWPR sur un bras anthropomorphique. Le modèle est appris le long d'une trajectoire en ajoutant une contrainte sur la posture globale du robot. Ces méthodes permettent donc d'apprendre un modèle sans pouvoir par la suite commander explicitement la mobilité interne des robots. Par exemple, il ne sera pas possible de modifier une contrainte sans réaliser un nouvel apprentissage.

2.2.2 Apprentissage de la cinématique

Alors qu'il existe une infinité de modèles cinématiques inverses dans le cas redondant, il n'existe qu'un seul modèle cinématique direct. Proches de nos travaux, Sun et Scassellati [2004, 2005] apprennent un modèle cinématique à l'aide de réseaux de fonctions à base radiale. Ils apprennent un modèle géométrique direct puis dérivent chaque fonction à base radiale, les somment pour former une matrice Jacobienne. Ils utilisent RMRC comme base pour leur algorithme incrémental de génération de trajectoire et deux caméras sur une tête de robot humanoïde pour obtenir une position opérationnelle de l'organe terminal.

Également proche de nos travaux, Butz *et al.* [2007]; Butz et Herbort [2008]; Butz *et al.* [2009]; Stalph *et al.* [2009] apprennent un modèle cinématique d'un bras anthropomorphique à quatre degrés de liberté. Ils utilisent XCSF comme méthode d'apprentissage afin de modéliser une adaptation motrice sur une expérience de pointage. Ils choisissent différentes tâches secondaires afin de montrer qu'ils commandent bien la mobilité interne dans la boucle de commande et non pendant l'apprentissage. Cette méthode est plus précisément décrite au chapitre suivant car c'est celle que nous emploierons lorsque nous utiliserons XCSF.

2.2.3 Apprentissage de la dynamique inverse

Nous avons vu en Section 1 qu'il était possible et utile d'inclure un modèle dynamique inverse dans une boucle de commande. Pour un système complètement actionné, la dimension des couples $\boldsymbol{\tau}^{con}$ est égale au nombre de degrés de liberté n. La relation est donc unique et apprenable.

Pour apprendre un modèle dynamique inverse, l'apprentissage est toujours auto-supervisé mais le couple (entrées, sorties) du système d'apprentissage devient $\boldsymbol{q}(t)$, $\dot{\boldsymbol{q}}(t)$, $\ddot{\boldsymbol{q}}(t+dt)$ pour les entrées et $\boldsymbol{\tau}^{con}(t)$ pour les sorties. En apprenant ainsi la dynamique, il est nécessaire de connaître l'accélération au pas de temps suivant et donc de réaliser l'apprentissage au pas de temps suivant. Contrairement aux modèles géométriques ou cinématiques, le modèle dynamique appris va donc dépendre du pas de temps. Ceci peut poser problème au passage de la simulation vers la robotique réelle.

2.2.4 Apprentissage de la dynamique

Narendra et Parthasarathy [1990] ont montré qu'il était possible d'apprendre un modèle dynamique inverse dans l'espace d'état avec des réseaux de neurones afin de les utiliser dans une loi de commande. Jordan et Rumelhart [1992] ont fait une très bonne revue des façons d'apprendre la dynamique et de l'utiliser en commande. Lin et Goldenberg [2001] apprennent un modèle dynamique inverse d'un robot à quatre degrés de liberté en trois dimensions en utilisant un réseau de fonctions à base radiale. Plus

récemment, Mitrovic *et al.* [2008] apprennent un modèle dynamique inverse avec LWPR qu'ils utilisent dans une loi de commande de type ILQG de manière à commander, en simulation, un bras anthropomorphique à deux degrés de liberté composé de six muscles artificiels.

Un modèle dynamique étant très complexe et surtout de grande dimension, certains travaux visent à n'apprendre que les états où il est nécessaire de commander le système. Par exemple, Vijayakumar et Schaal [1997], Vijayakumar et Schaal [2000], Schaal *et al.* [2002] et Vijayakumar *et al.* [2005] apprennent avec LWPR un modèle dynamique inverse le long d'une trajectoire suivie grâce à un correcteur proportionnel dérivé. L'ajout d'un bruit additionnel dans leur boucle de commande leur permet d'apprendre autour de la trajectoire une enveloppe de configurations. Malheureusement, avec cette approche, il parait difficile de commander la mobilité interne des systèmes.

3 Commande de la mobilité interne à l'aide de modèles appris

Dans la section précédente, nous avons vu qu'aucune des architectures - [Butz et Herbort, 2008] mis à part - n'est capable de commander explicitement la mobilité interne des robots redondants à l'aide de modèles appris. Ce chapitre développe une méthode qui combine l'apprentissage de modèles mécaniques et une commande dans l'espace opérationnel permettant de réaliser des tâches hiérarchiquement ordonnées par des robots redondants vis-à-vis de ces tâches.

3.1 Apprentissage de cinématique et commande de la mobilité interne

La spécificité de notre approche réside dans le fait que l'on souhaite commander la mobilité interne de robots redondants. Ceci induit les contraintes suivantes. Il est nécessaire d'apprendre un modèle cinématique direct pour calculer un projecteur et donc commander la mobilité interne de robots redondants ; un modèle cinématique inverse ou un modèle géométrique ne le permet pas. Il est nécessaire que le modèle appris ne soit pas sous la forme de relation entre les entrées et les sorties du système d'apprentissage mais bien sous une forme linéaire explicite qu'il est possible d'inverser. Autrement dit, apprendre un modèle cinématique sous la forme $\boldsymbol{\nu} = \mathcal{FVK}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ est conceptuellement différent d'apprendre une matrice Jacobienne $\boldsymbol{J}(\boldsymbol{q})$ sous la forme $\boldsymbol{\nu} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}$. Même s'il est difficile d'apprendre sur un espace de grande dimension, il est nécessaire de pouvoir obtenir un modèle dans tous les états que pourrait atteindre le système en utilisant sa mobilité interne.

3.1.1 Apprentissage d'un modèle cinématique avec lwpr

Une solution qui satisfait les contraintes exprimées consiste à apprendre un modèle avec LWPR en utilisant la capacité de l'algorithme à fournir la dérivée du modèle appris par rapport aux entrées. En apprenant un modèle géométrique direct, il est alors possible d'en extraire la dérivée par rapport aux entrées $\boldsymbol{q}$ appelée matrice Jacobienne. Nous écrivons cette relation sous la forme

$$\left[\hat{\boldsymbol{\xi}}, \hat{J}\right] = \text{LWPR}_{predict}\left([\mathcal{FK}_{\text{LWPR}}], [\boldsymbol{q}]\right)$$

Pour utiliser LWPR, il est nécessaire d'apprendre au préalable un modèle calculé à partir de données aléatoires issues d'un modèle analytique.

3.1.2 Apprentissage d'un modèle cinématique avec xcsf

Nous avons écrit en Section 2.1.2 que XCSF distingue un *prediction input space* et un *condition space*. Pour illustrer les possibilités ouvertes par cette capacité, nous rappelons la relation cinématique (Équation 2) $\boldsymbol{\nu} = J(\boldsymbol{q})\,\dot{\boldsymbol{q}}$ où la matrice Jacobienne J lie $\boldsymbol{\nu}$ à $\dot{\boldsymbol{q}}$ tout en n'étant dépendante que des paramètres articulaires $\boldsymbol{q}$.

Nous écrivons la relation qui consiste à apprendre un modèle cinématique avec XCSF sous la forme

$$\mathcal{FVK}_{\text{XCSF}} = \text{XCSF}_{learn}([\boldsymbol{q}], [\dot{\boldsymbol{q}}], [\boldsymbol{\nu}]) \tag{20}$$

où $\boldsymbol{q}$ définit le *condition space*, i.e., l'espace spécifiant la dépendance explicite du modèle appris tandis que la paire $[\dot{\boldsymbol{q}}], [\boldsymbol{\nu}]$ définit le *prediction space* utilisé pour calculer les erreurs et apprendre le modèle.

Les domaines d'activation des classeurs dans XCSF peuvent se superposer. Extraire la matrice Jacobienne d'une configuration particulière s'écrit

$$\hat{J}(\boldsymbol{q}) = \sum_{i=1}^{n_M} \phi_i(\boldsymbol{q})\,\beta_i. \tag{21}$$

où n_M est le nombre de classeurs actifs.

L'Équation 21 sera notée

$$\hat{J}(\boldsymbol{q}) = \text{XCSF}_{predictJ}\left([\mathcal{FVK}_{\text{XCSF}}], [\boldsymbol{q}]\right) \tag{22}$$

quand le modèle est utilisé pour prédire une matrice Jacobienne.

3.1.3 Combinaison de tâches

Afin de combiner des tâches, nous utilisons les deux algorithmes d'apprentissage dans une boucle de commande utilisant le formalisme RMRC. Deux modèles cinématiques différents, correspondant chacun à une tâche

différente, sont appris séparément puis sont utilisés de manière hiérarchique comme décrit dans [Maciejewski et Klein, 1985] (voir Équation 11)

$$\dot{\boldsymbol{q}}^{\star} = \hat{J}_1^{+}\boldsymbol{\nu}_1^{\star} + \left(\hat{J}_2 P_{\hat{J}_1}\right)^{+}\left(\boldsymbol{\nu}_2^{\star} - \hat{J}_2\hat{J}_1^{+}\boldsymbol{\nu}_1^{\star}\right) \tag{23}$$

où $\hat{J}_1$ et $\hat{J}_2$ sont les matrices Jacobiennes estimées par les algorithme d'apprentissage. $P_{\hat{J}_1}$ et $\hat{J}_2 P_{J_1}$ sont calculés en utilisant une décomposition en valeur singulière.

Les vitesses opérationnelles désirées $\boldsymbol{\nu}_i^{\star}$ sont calculées en utilisant l'erreur opérationnelle

$$\boldsymbol{\nu}_i^{\star} = K_{pi}\left(\boldsymbol{\xi}_i^{\dagger} - \boldsymbol{\xi}_i\right) \tag{24}$$

où K_{pi} est une matrice de gain définie positive et $\boldsymbol{\xi}_i^{\dagger}$ est le but à atteindre.

Le schéma de commande peut être résumé par l'Algorithme 2.

Algorithme 2 Boucle de commande avec modèles cinématiques appris

$\boldsymbol{\nu}_1^{\star}(t) = K_p\left(\boldsymbol{\xi}_1^{\dagger}(t) - \boldsymbol{\xi}_1(t)\right)$ *vitesse opérationnelle désirée*

$\boldsymbol{\nu}_2^{\star}(t) = K_p\left(\boldsymbol{\xi}_2^{\dagger}(t) - \boldsymbol{\xi}_2(t)\right)$ *vitesse opérationnelle désirée*

$\dot{\boldsymbol{q}}^{\star}(t) = \hat{J}_1^{+}\boldsymbol{\nu}_1^{\star}(t) + \left(\hat{J}_2 P_{\hat{J}_1}\right)^{+}\left(\boldsymbol{\nu}_2^{\star}(t) - \hat{J}_2\hat{J}_1^{+}\boldsymbol{\nu}_1^{\star}(t)\right)$ *vitesses articulaires désirées*

$\ddot{\boldsymbol{q}}^{\star}(t) = \left(\dot{\boldsymbol{q}}^{\star}(t) - \dot{\boldsymbol{q}}(t)\right)/dt$ *accélérations articulaires désirées*

$\boldsymbol{\tau}^{con} = \mathcal{ID}\left(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \ddot{\boldsymbol{q}}^{\star}(t)\right)$ *couples désirés*

$\dot{\boldsymbol{q}}(t+dt) \leftarrow \mathcal{D}\left(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \boldsymbol{\tau}^{con}\right)$ *intégration du simulateur*

3.2 Apprentissage de la dynamique inverse

Il est possible d'apprendre la dynamique inverse le long d'une trajectoire. Cependant, si l'on souhaite commander la mobilité interne, le système va se retrouver dans des états où le modèle n'a pas été appris. Il est donc nécessaire d'apprendre le modèle dynamique inverse sur tout l'espace d'état et pour toutes les accélération possibles.

La boucle de commande est similaire à celle présentée à la section précédente avec comme modification l'utilisation d'un modèle dynamique inverse $\mathcal{ID}$ appris et non pas fourni par le simulateur.

Il est important de noter ici que, contrairement à de nombreuses approches, le modèle dynamique inverse étant appris dans l'espace articulaire, il reste valable quelle que soit la matrice Jacobienne utilisée dans la commande.

3.2.1 Apprentissage de la dynamique inverse avec lwpr

LWPR nous permet d'estimer la dynamique inverse comme une relation entre les entrées $[\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \ddot{\boldsymbol{q}}(t+dt)]$ et les sorties $[\boldsymbol{\tau}^{con}(t)]$.

L'apprentissage nous fournit un modèle sous la forme

$$\mathcal{ID}_{\text{LWPR}} = \text{LWPR}_{learn}\left([\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \ddot{\boldsymbol{q}}(t+dt)], [\boldsymbol{\tau}^{con}(t)]\right).$$

Il est alors possible de commander notre système avec le modèle dynamique inverse appris

$$\boldsymbol{\tau}^{con}(t) = \text{LWPR}_{predict}\left(\mathcal{ID}_{\text{LWPR}}, [\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \ddot{\boldsymbol{q}}^{\star}(t+dt)]\right)$$

où $\ddot{\boldsymbol{q}}^{\star}(t+dt) = \dfrac{\dot{\boldsymbol{q}}^{\star}(t+dt) - \dot{\boldsymbol{q}}(t)}{dt}$ est l'accélération désirée.

3.2.2 Apprentissage de la dynamique inverse avec xcsf

Afin d'expliquer comment XCSF est utilisé pour apprendre un modèle dynamique inverse, nous rappelons l'équation de la dynamique des systèmes poly-articulés

$$\boldsymbol{\tau}^{con} = A(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \boldsymbol{b}(\boldsymbol{q}, \dot{\boldsymbol{q}}). \tag{25}$$

Cette équation peut être vue comme une équation linéaire entre $\ddot{\boldsymbol{q}}$ et $\boldsymbol{\tau}^{con}$ dépendant des paramètres $\boldsymbol{q}(t)$ et $\dot{\boldsymbol{q}}(t)$

$$\boldsymbol{\tau}^{con} = \begin{bmatrix} A(\boldsymbol{q}) & \boldsymbol{b}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}} \\ 1 \end{bmatrix}. \tag{26}$$

Dans ce contexte, $\boldsymbol{b}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ peut être vu comme un *offset*.

Il est donc possible d'apprendre un modèle dynamique inverse en fournissant à chaque pas de temps l'état $(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t))$ comme entrée du *condition space* et $(\ddot{\boldsymbol{q}}(t+dt), \boldsymbol{\tau}^{con}(t))$ comme couple (entrées, sorties) du *prediction space*.

$$\mathcal{ID}_{\text{XCSF}} = \text{XCSF}_{learn}\left([\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t)], [\ddot{\boldsymbol{q}}(t+dt)], [\boldsymbol{\tau}^{con}(t)]\right) \tag{27}$$

On peut donc calculer le couple avec le modèle appris de la manière suivante

$$\boldsymbol{\tau}^{con} = \text{XCSF}_{predict}\left(\mathcal{ID}_{\text{XCSF}}, [\boldsymbol{q}, \dot{\boldsymbol{q}}], [\ddot{\boldsymbol{q}}]\right). \tag{28}$$

Cette approche est computationnellement moins gourmande en termes de dimension que celle utilisée avec LWPR.

3.3 Évaluation de la qualité de prédiction

Pour évaluer la qualité de prédiction des algorithmes d'apprentissage, nous utilisons deux types d'erreurs de prédiction.

– La *Normalized Mean Square Error* (NMSE) est utilisée pour se comparer à la littérature et calculée de la manière suivante

$$\text{NMSE} = \frac{1}{\sigma^2}\frac{1}{n_p}\sum_{i}^{n_p}(y_i - \hat{y}_i)^2$$

où σ^2 est la variance de y : $\sigma^2 = \frac{1}{n_p}\sum_{k=1}^{n_p}(y_k - \overline{y}_k)^2$.

– La *Normalised Mean Absolute Error* (NMAE) est calculée de la manière suivante

$$\text{NMAE} = \frac{1}{n_p\left(\max(y_i) - \min(y_i)\right)}\sum_{i=1}^{n_p}\|\hat{y}_i - y_i\|$$

où $\max(y_i)$ et $\min(y_i)$ sont respectivement le maximum et le minimum des données et n_p est le nombre de points utilisés pour calculer l'erreur. Cette erreur nous semble la plus adaptée car elle a la même dimension et la même échelle que les données estimées.

4 Combinaison de tâches hiérarchiques

Cette section met en œuvre, en simulation, le schéma de commande détaillé à la section précédente en considérant que la dynamique du système commandé est connue. Nous présentons des simulations simples de chaînes poly-articulées à trois degrés de liberté dans le plan de manière à tester notre loi de commande dans le cas sur-contraint ou non, en utilisant des modèles appris. Les longueurs des segments sont 0.50m, 0.40m et 0.20m avec des masses respectives de 1kg, 0.8kg et 0.2kg. Pour simuler nos robots, nous utilisons le simulateur dynamique ARBORIS [Barthélemy et Micaelli, 2008]. Les résultats correspondants ont été publiés dans [Salaün *et al.*, 2009a, 2010]. Des résultats préliminaires ont été publiés dans [Salaün *et al.*, 2009b].

4.1 Initialisation d'un modèle cinématique avec lwpr

Pour apprendre nos modèles, nous utilisons l'algorithme développé par l' *Institute of Perception, Action and Behaviour by the Statistical Machine Learning and Motor Control Group*[1] de l'Université d'Edinburgh.

LWPR est un algorithme difficile à régler. Il est nécessaire d'initialiser un modèle fondé sur des données aléatoires de manière à régler ces paramètres

1. http ://www.ipab.inf.ed.ac.uk/slmc/software/lwpr/

de la manière décrite par Klanke *et al.* [2008]. Chaque paramètre est successivement réglé expérimentalement. Les données choisies aléatoirement sont les configurations articulaires et leurs positions opérationnelles correspondantes. Les paramètres obtenus sont regroupés dans le tableau 1.

Paramètres	$norm_{in}$	$init_D$	$update_D$	$init_\alpha$	w_{gen}	γ
Valeurs	2π	20	1	10000	0.5	$1e^{-6}$

TABLE 1 – Paramètres utilisés pour apprendre un modèle cinématique inverse avec LWPR

La Figure 4 représente la décroissance de la NMSE pendant l'étape d'initialisation aléatoire. On peut voir qu'à partir de 5000 exemples, le modèle créé est suffisamment précis pour obtenir une matrice Jacobienne utilisée dans une boucle de commande.

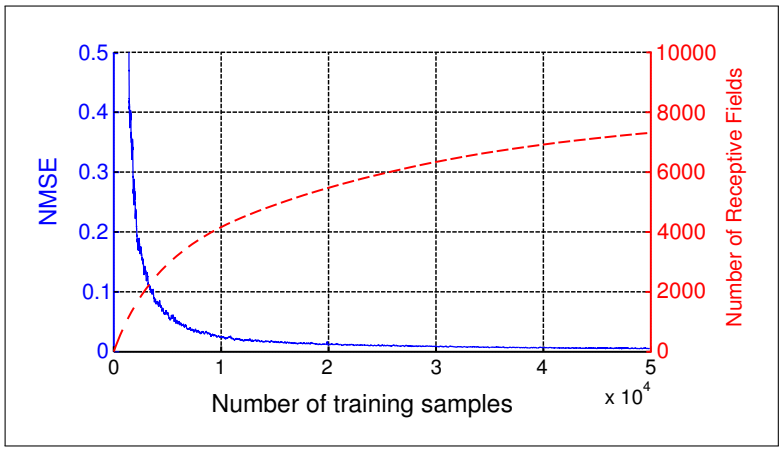


FIGURE 4 – Évolution de la NMSE de la vitesse opérationnelle prédite (bleu, lignes pleines, échelle de gauche) et du nombre de champs récepteurs pour une seule sortie (rouge, ligne pointillée, échelle de droite) en utilisant LWPR (moyenne sur 40 essais). Les 50000 configurations sont choisies aléatoirement.

4.2 Combinaisons de tâches avec lwpr

Tandis que, pendant l'initialisation, les résultats sont issus d'une moyenne sur 40 exemples, les résultats présentés dans cette section sont issus d'un exemple représentatif.

La première tâche étudiée est une tâche de pointage depuis une position opérationnelle initiale $\boldsymbol{\xi}_1^i = [0.10\ 1.00]^T\ m$ vers la cible $\boldsymbol{\xi}_1^\dagger = [0.20\ 0.50]^T\ m$ avec une précision de 0.01 mètre. Une fois la cible atteinte, l'organe terminal

est repositionné à sa position initiale avec la même commande et la même précision. Ce mouvement est répété jusqu'à la fin de la simulation.

La tâche étant de dimension 2, le système est sous-contraint. La mobilité interne n'est cependant pas commandée.

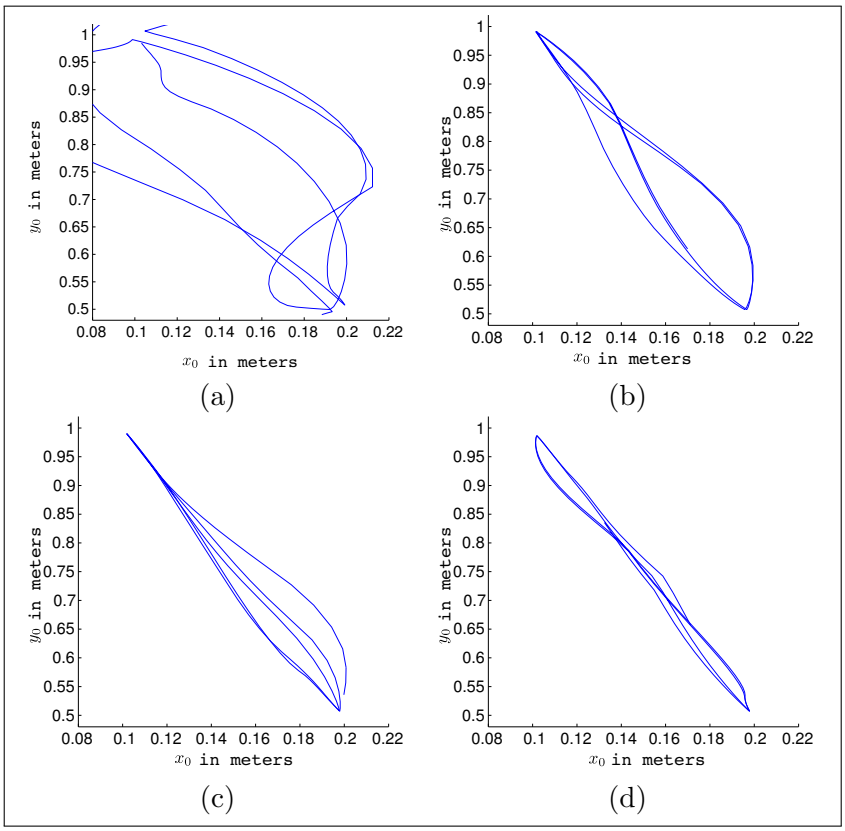


FIGURE 5 – Évolution de la position opérationnelle de l'organe terminal pendant l'apprentissage de la matrice Jacobienne servant à commander un robot à trois degrés de liberté dans le plan. Chaque figure représente deux secondes de simulation.(a) : 0s à 2s.(b) : 2s à 4s.(c) : 6s à 8s.(d) : 20s à 22s.

Afin de montrer une évolution du modèle, son initialisation n'est réalisée que sur 2000 exemples. On peut voir une évolution allant de l'incapacité à pouvoir commander le système pendant les premières secondes jusqu'à obtenir une trajectoire quasiment rectiligne après 20 secondes de simulation.

La seconde expérience correspond au cas sur-contraint qui consiste à atteindre la position $\boldsymbol{\xi}_1^\dagger = [0.20\ 0.50]^T\, m$ avec un organe terminal paramétré par $\boldsymbol{\xi}_1$ puis à conserver cette position tout en réalisant une tâche secondaire. Cette tâche consiste à atteindre alternativement, avec un autre organe terminal $\boldsymbol{\xi}_2$, la position $\boldsymbol{\xi}_2^\dagger = [0.45\ 0.25]^T\, m$. Cette seconde tâche n'est pas compatible avec la première. Cette expérience, illustrée en Figure 6, met

en évidence l'efficacité de la méthode utilisée pour la commande de robots redondants. La commande maintient une distance inférieure à 0.01 entre l'organe terminal $\boldsymbol{\xi}_1$ et sa position désirée $\boldsymbol{\xi}_1^\dagger$. En même temps, la tâche secondaire est achevée partiellement avec une erreur la plus petite possible sans perturber la tâche principale.

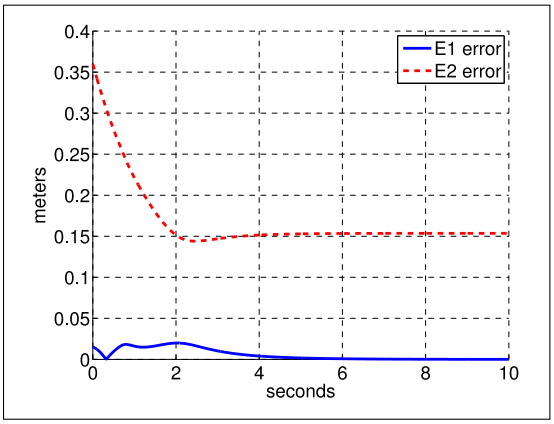


FIGURE 6 – Erreurs de pointage pour la première (bleue, ligne pleine) et la seconde (rouge, ligne pointillée) tâche dans le cas compatible après apprentissage.

4.3 Apprentissage de la cinématique avec xcsf

Dans cette thèse, nous utilisons l'implémentation de XCSF du *COgnitive BOdy Spaces : Learning And Behavior (*COBOSLAB*)* de l'*University of Würzburg*[2].

Les paramètres qui doivent être réglés sont, par ordre d'importance

- le seuil de l'erreur, réglé à 0.1, sous lequel le système arrête de créer de nouveaux classeurs mais essaie à la place de généraliser,
- le nombre maximum de micro-classeurs, réglé à 350,
- le nombre maximum d'itérations, réglé à 150000,
- le nombre d'itérations avant condensation, réglé à 100000,
- le nombre de simulations d'apprentissage, réglé à 20 pour assurer une stabilité du modèle.

Contrairement à LWPR, XCSF ne nécessite pas de phase d'initialisation et peut donc être utilisé dans une loi de commande immédiatement.

La Figure 7 montre l'évolution de la MAE en fonction du nombre d'itérations. La courbe est une moyenne sur 20 simulations. Elle montre l'adaptation du modèle pendant la phase de commande avec une erreur qui décroit avec une variance assez grande. On peut voir sur la même figure qu'une

2. http ://medal.cs.umsl.edu/files/XCSF_Ellipsoids_Java.zip

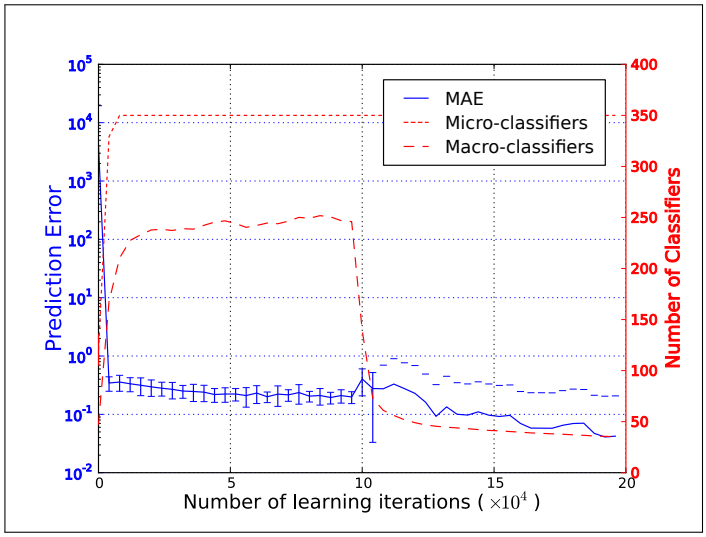


FIGURE 7 – MAE (bleu, ligne pleine) pour la prédiction le long de l'axe x. Nombre de classeurs (rouge, lignes pointillées).

fois le nombre maximum de micro-classeurs atteint, le nombre de macro-classeurs décroit pendant la phase de compaction. Le seuil d'erreur maximal est atteint.

Afin de pouvoir comparer les résultats obtenus, l'expérience menée est la même expérience de pointage que celle utilisant LWPR comme algorithme d'apprentissage. Contrairement à LWPR, il n'y a aucune étape d'initialisation. Après 40 secondes, la trajectoire devient rectiligne.

4.4 Discussion

Les résultats des simulations dans le cas sous-contraint montre que notre méthode est capable de générer un modèle du système tout en le commandant (capitaine ?) avec la précision requise. La trajectoire converge avec les deux algorithmes d'apprentissage jusqu'à devenir rectiligne.

À notre connaissance, les résultats obtenus dans le cas sur-contraint sont originaux dans le fait d'utiliser des modèles appris pour commander la mobilité interne de robots [Salaün *et al.*, 2009a]. La hiérarchie entre les tâches est bien respectée.

Il est important de noter que la plupart des résultats présentés dans cette thèse ont été obtenus en utilisant LWPR comme outil d'apprentissage. XCSF étant été découvert tardivement, il sert surtout ici de système de comparaison pour nos résultats. Cependant, une comparaison directe des deux algorithmes est difficile pour plusieurs raisons.

– La performance des deux algorithmes est sensible au paramètrage et

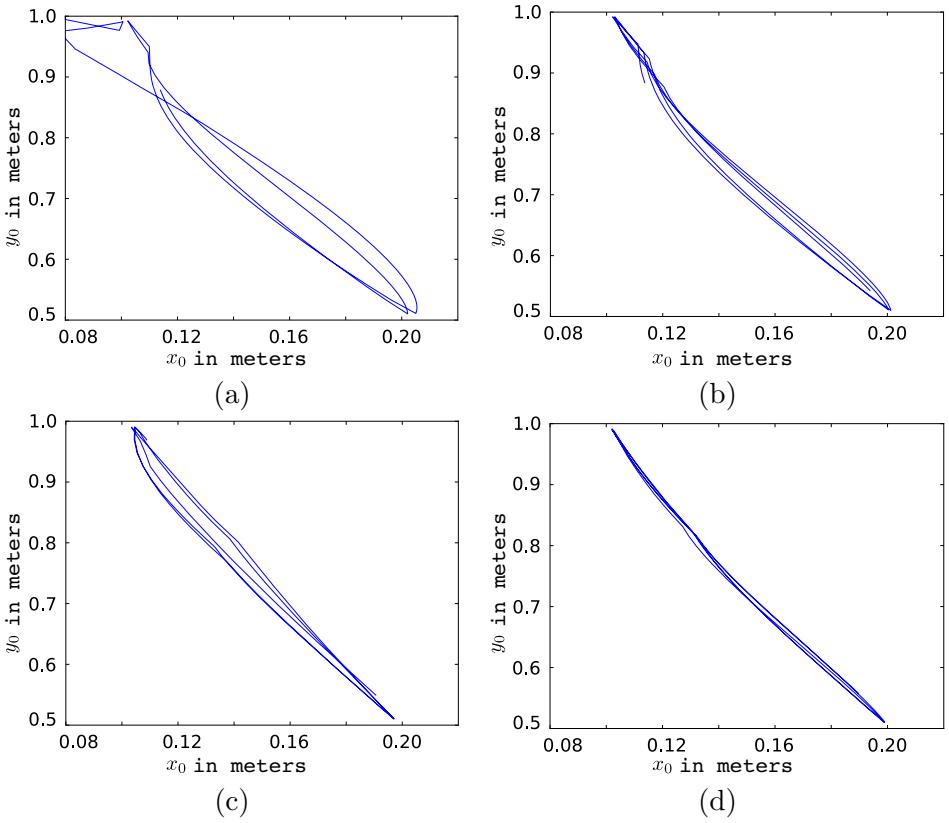


FIGURE 8 – Évolution des trajectoires opérationnelles pendant l'apprentissage, avec XCSF, du modèle cinématique réalisant une tâche sous contrainte. Chaque figure représente 4 secondes de simulation.(a) : 0s à 4s.(b) : 4s à 8s.(c) : 12s à 16s.(d) : 36s à 40s.

donc la performance est sensible au temps passé à régler les paramètres.

- Pour bien fonctionner, LWPR nécessite une initialisation tandis que XCSF peut être utilisé directement dans une loi de commande. La comparaison de la performance est donc également dépendante de l'initialisation de LWPR.
- Les erreurs de chaque algorithmes sont calculées sur des espace différents. La cinématique apprise par LWPR est calculée sur la base d'erreurs sur les positions cartésiennes tandis que l'apprentissage avec XCSF dépend des vitesses opérationnelles.

Il est néanmoins possible de comparer quelque peu ces expériences préliminaires.

- Le fait que XCSF ne nécessite aucune initialisation est une qualité de l'algorithme car cette initialisation est souvent difficile à pratiquer avec de véritables robots.

– La capacité que possède XCSF à distinguer un *condition space* et un *prediction input space* en fait un algorithme plus proche de la représentation de la matrice Jacobienne qui dépend d'une configuration tout en faisant une liaison entre des vitesses articulaires et opérationnelles du robot.
– Les simulations préliminaires avec XCSF tendent à montrer qu'il est aussi précis que LWPR sans pour autant qu'il soit nécessaire de réaliser un réglage très fin ainsi qu'une initialisation.
– Cependant, nous pouvons souligner que XCSF est, dans un premier temps, moins stable que LWPR. Ceci peut s'expliquer par le fait que nous l'utilisons sans initialisation. Après compaction, on obtient par contre la même précision pour les deux algorithmes.

5 Expériences sur le robot humanoïde iCub

Dans ce chapitre, nous présentons les résultats de l'application de la loi de commande décrite au Chapitre 3 en utilisant LWPR sur le robot humanoïde ICUB [3] [Metta *et al.*, 2008].

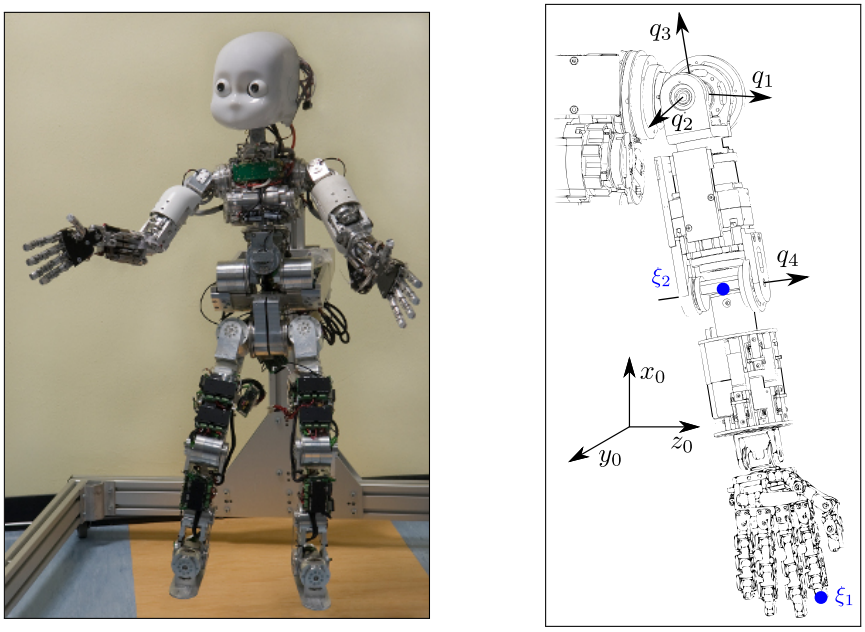
Dans les sections suivantes, nous expliquons tout d'abord comment régler LWPR afin de pouvoir ensuite utiliser l'algorithme dans une loi de commande permettant de l'utiliser sur un robot réel. Les expériences sont menées sur une partie du bras gauche du robot soit trois degrés de liberté pour l'épaule et un degré de liberté pour le coude.

5.1 Initialisation d'un modèle cinématique appris avec lwpr pour iCub.

Nous utilisons LWPR pour apprendre le modèle géométrique direct des deux organes terminaux qui sont le bout du doigt nommé $\boldsymbol{\xi}_1$ et le coude nommé $\boldsymbol{\xi}_2$. Comme décrit dans les sections précédentes, nous fournissons $\boldsymbol{q}$ comme entrée et $\boldsymbol{\xi}_1$ ou $\boldsymbol{\xi}_2$ comme sortie selon le modèle que l'on souhaite apprendre.

Comme il est difficile de réaliser l'initialisation sur le robot réel, elle est réalisée sur le simulateur ICUB [Tikhanoff *et al.*, 2008] où il est possible de calculer un modèle analytique du robot pour n'importe quel état. Afin de tout de même apprendre un modèle avec des données issues du robot, nous avons mesuré les positions articulaires du robot avec les codeurs incrémentaux pris comme entrée du système d'apprentissage. Les positions opérationnelles étant difficilement accessibles par une mesure issue de capteur extéroceptifs (vision binoculaire), nous avons choisi, pour des raisons de simplicité, d'utiliser un modèle analytique du robot calculé à l'aide de la *Kinematic Dynamics Library* (KDL) du projet *Orocos* [4] [Smits *et al.*, 2008].

3. www.robotcub.org
4. http ://www.orocos.org



(a) Front view of iCub.

(b) iCub left arm front view.

FIGURE 9 – Le robot humanoïde iCub et son bras gauche. Les axes de l'épaule et du coude sont représentés ainsi que le repère commandés.

Dans les expériences qui suivent, nous allons comparer les données issues du modèle LWPR avec les données issues du modèle KDL. Étant donné que l'on utilise des données issues du modèle KDL pour calculer le modèle LWPR, ce dernier ne pourra être plus précis que le modèle KDL.

Afin d'utiliser LWPR avec suffisamment de précision, il est nécessaire de régler les paramètres de l'algorithme. La normalisation des entrées $norm_{in}$ est réglée par rapport aux possibilités de chaque liaison :
$norm_{in} = [105.5, 160.8, 117, 111.5]$. $init_D$ est réglé expérimentalement en comparant les performances de plusieurs valeurs sur un échantillon représentatif. On fixe $init_D = 150$ et $w_{gen} = 0.2$.

5.2 Combinaison de tâches incompatibles sur iCub

Nous avons réalisé une expérience en simulation et sur le robot réel dans le cas où les tâches sont incompatibles. Le but $\boldsymbol{\xi}_1^\dagger$, à atteindre avec le bout du doigt, possède la plus grande priorité. Il est réglé en coordonnées absolues à $[0.00, -0.05, 0.03]$ mètres sur les axes x, y et z respectivement. La seconde tâche $\boldsymbol{\xi}_2^\dagger$, associée au coude, est une tâche secondaire réalisée avec une erreur minimale. Elle est réglée en coordonnées absolues à $[-0.20, 0.03]$ sur les axes x et z respectivement. Les tâches sont incompatibles car on souhaite commander cinq paramètres avec seulement quatre degrés de liberté.

$\boldsymbol{\nu}^\star$ est calculé avec un contrôleur proportionnel : $\boldsymbol{\xi}^\star = K_p\left(\boldsymbol{\xi}^\dagger - \boldsymbol{\xi}\right)$. Nous choisissons la matrice K_p diagonale avec $K_{p_{11}}$ et $K_{p_{22}}$ respectivement égaux à $3.0s^{-1}$ et $0.2s^{-1}$.

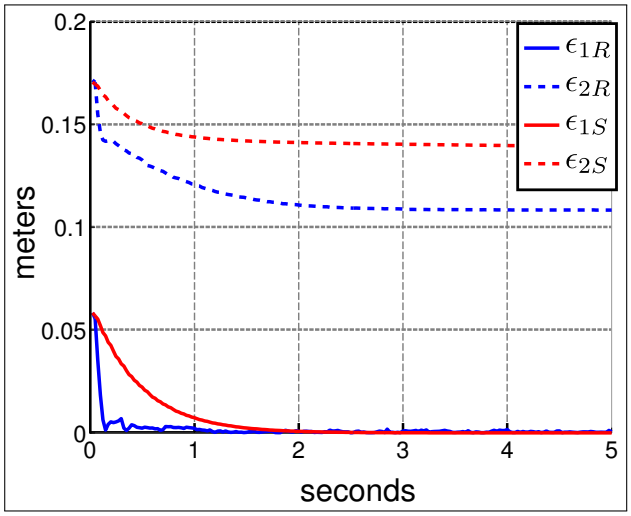


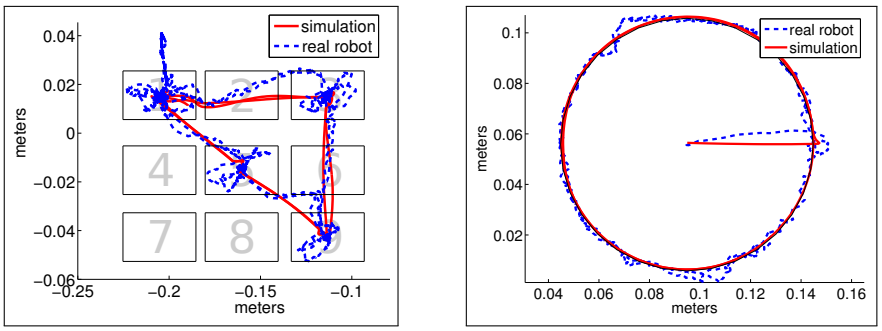
FIGURE 10 – Évolution des erreurs opérationnelles relatives aux tâches 1 et 2 avec apprentissage en simulation (a) et sur le robot ICUB (b). $\boldsymbol{\epsilon}_{1S}$ est l'erreur de position entre $\boldsymbol{\xi}_1$ et $\boldsymbol{\xi}_1^\dagger$ en simulation. $\boldsymbol{\epsilon}_{2S}$ est l'erreur relative à la seconde tâche en simulation. $\boldsymbol{\epsilon}_{1R}$ et $\boldsymbol{\epsilon}_{2R}$ sont les erreurs relatives aux deux tâches sur le robot réel.

La Figure 10 montre l'évolution de l'erreur opérationnelle pour chacune des deux tâches dans le cas simulé et dans le cas réel indexés S et R respectivement. La tâche principale est bien réalisée tandis que la tâche secondaire converge vers un minimum d'erreur.

5.3 Expériences pratiques avec iCub

Afin d'illustrer une tâche de pointage sur une application réelle, nous avons choisi de faire réaliser deux tâches au robot ICUB. Une des tâches consiste à appuyer séquentiellement sur les touches $1-5-9-3$ d'un digicode, les positions des touches étant connues a priori. Une seconde tâche consiste à dessiner un cercle sur un tableau blanc avec un marqueur. Techniquement, cette tâche consiste à suivre un point $\boldsymbol{\xi}^\dagger(t)$ évoluant sur une trajectoire circulaire dans le plan du tableau.

La Figure 11 montre les trajectoires opérationnelles obtenues lors de la commande du robot avec un modèle appris avec LWPR après convergence du modèle. La Figure 11a montre que la séquence est bien réalisée même si les trajectoires ne sont pas rectilignes. La Figure 11b montre l'évolution de la trajectoire du bout du stylo dans le plan (x_0, z_0). La qualité du suivi



(a) Expérience sur le digicode.

(b) Dessin d'un cercle.

FIGURE 11 – Trajectoires de l'organe terminal dans l'expérience du pavé numérique et du cercle tracé, utilisant un modèle appris avec LWPR en simulation (ligne pleine rouge) puis sur ICUB (ligne bleue pointillée) dans le plan (x_0, z_0).

dépend du gain proportionnel K_p qui peut être vu comme la raideur d'un ressort entre l'organe terminal et le point suivi. La qualité du suivi dépend aussi de la vitesse du point suivi. Nous avons pris $K_p = 3.0s^{-1}$ et $2.0rad.s^{-1}$ comme vitesse du point suivi sur la trajectoire circulaire. En simulation, la trajectoire est quasiment parfaite tandis que, sur le robot réel, elle est bruitée.

En examinant les résultats, nous pouvons conclure qu'il est possible d'apprendre un modèle cinématique avec LWPR afin de commander un véritable robot. Nous pouvons également voir que l'approche présentée ne permet pas de commander précisément le robot ICUB, que ce soit avec un modèle analytique obtenu avec KDL ou avec un modèle appris par LWPR. L'utilisation de la vision binoculaire du robot ainsi que le contrôle de la dynamique pourraient être utilisées pour surmonter cette difficulté.

6 Simulations dynamiques

Nous avons vu dans le chapitre précédent qu'il pouvait être nécessaire de contrôler la dynamique d'un robot, particulièrement dans le cadre de la robotique de service où les interactions avec l'environnement sont difficilement prévisibles. Ce chapitre décrit les résultats obtenus lorsque l'on souhaite également apprendre un modèle dynamique et l'incorporer dans une loi de commande.

La première section décrit comment apprendre un modèle dynamique inverse sur l'espace des positions, vitesses et accélérations articulaires. La méthode est appliquée, à titre d'exemple, sur un robot plan à trois degrés de liberté. La seconde section décrit comment il est possible de compenser

une force appliquée sur l'organe terminal du robot à l'aide du modèle dynamique appris. Nous décrivons également comment un modèle dynamique inverse appris et soumis à une perturbation peut induire des effets non-désirés appelés *after-effects*.

6.1 Apprentissage de la dynamique inverse

Cette section décrit comment apprendre la dynamique inverse d'un robot plan possédant trois degrés de liberté. Nous décrivons comment régler les paramètres ainsi que la phase d'initialisation. Nous décrivons également comment apprendre la dynamique inverse avec XCSF en soulignant qu'il est facile d'utiliser directement le modèle appris dans une boucle de commande.

6.1.1 Apprentissage de la dynamique inverse avec lwpr

Les entrées étant maintenant des positions, vitesses et accélérations articulaires, le paramètre $norm_{in}$ est réglé en fonction des valeurs possibles des différentes variables. Les positions articulaires sont limitées entre 0 et 2π rad. Les vitesses articulaires sont signées car les articulations peuvent varier dans le sens positif ou dans le sens négatif. Nous avons décidé de les normaliser par 10 qui est approximativement la valeur maximale que peuvent atteindre les liaisons. Les accélérations articulaires sont normalisées par 200 pour la même raison. Nous rappelons ici qu'une mauvaise normalisation va perturber l'apprentissage mais ne sera pas critique si les valeurs maximales des données d'entrée vont au-delà des limites choisies. Par exemple, si la vitesse de 11 rad/sec est atteinte par une articulation, elle sera divisée par 10 et pourra être apprise, même si elle n'est pas dans l'intervalle $[0, 1]$.

Un résumé des valeurs est donné dans le Tableau 2.

Parameters	$norm_{in}$	$init_D$	$update_D$	$init_\alpha$	w_{gen}	γ
Values	$[2\pi, 10, 200]$	0.1	1	10000	0.5	$1e^{-6}$

TABLE 2 – Paramètres utilisés pour apprendre la dynamique inverse avec LWPR

L'apprentissage de la dynamique est donc implémenté de la manière suivante

$$\mathcal{ID}_{\text{LWPR}} = \text{LWPR}_{learn}\left(\left[\boldsymbol{q}\left(t\right), \dot{\boldsymbol{q}}\left(t\right), \ddot{\boldsymbol{q}}\left(t+dt\right)\right], \left[\boldsymbol{\tau}^{con}\left(t\right)\right]\right). \tag{29}$$

Les résultats de l'initialisation apparaissent sur la Figure 12 qui montre l'évolution de l'erreur d'estimation pendant l'apprentissage.

L'erreur utilisée est la *Normalized Mean Absolute Error* (NMAE) calculée par

$$\text{NMAE} = \frac{1}{n_p \|\hat{\tau}^{con}_{max}\|} \sum_{i=1}^{n_p} \|\hat{\tau}^{con} - \tau^{con}\|$$

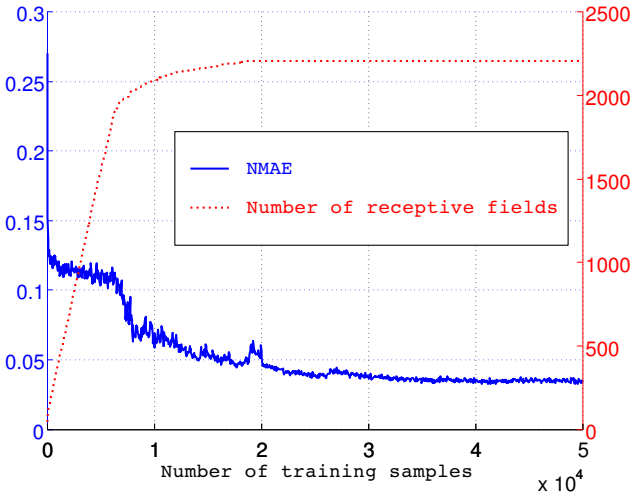


FIGURE 12 – Évolution de la NMAE des couples appris pour l'axe 1 et nombre de champs récepteur en fonction du nombre de données d'entrée.

où $\hat{\tau}^{con}_{max}$ est le couple maximum et $n_p = 1000$ est le nombre de points utilisés pour calculer l'erreur.

Nous avons continué à apprendre tout en réalisant une trajectoire rectiligne avec un modèle dynamique inverse analytique afin d'améliorer le modèle appris.

Notre loi de commande peut être utilisée pour commander un système avec deux tâches hiérarchiques $\boldsymbol{\xi}_1^\dagger = 0.00m$ sur l'axe x_0 et $0.50m$ sur l'axe y_0 tandis que $\boldsymbol{\xi}_2^\dagger$ est fixé alternativement à $-0.15m$ et $0.15m$ sur l'axe x_0. L'alternance des buts a lieu quand l'erreur de position est inférieure à $0.001m$. K_{p1} et K_{p2} sont choisies respectivement à $40s^{-1}$ et $0.5s^{-1}$.

Dans notre loi de commande, nous apprenons les modèles cinématiques et dynamique à l'aide de LWPR, tout d'abord la cinématique de la manière décrite au Chapitre 4 , puis la dynamique une fois la cinématique apprise.

La Figure 13 montre la distance entre l'organe terminal et la cible avec trois conditions différentes : dynamique et cinématique analytiques, cinématique analytique et dynamique apprise, et le cas ou les deux modèles sont appris.

6.1.2 Apprentissage de la dynamique inverse avec xcsf

Nous utilisons XCSF pour apprendre la dynamique inverse du même robot plan à trois degrés de liberté avec la méthode décrite en Section 3.2.2.

L'état $[\boldsymbol{q}, \dot{\boldsymbol{q}}]$, normalisé par ses valeurs maximales, est fourni comme entrée du *condition space*. Les sorties du *prediction space* sont les couples $\boldsymbol{\tau}^{con}$ calculés à partir du modèle analytique et les entrées sont les

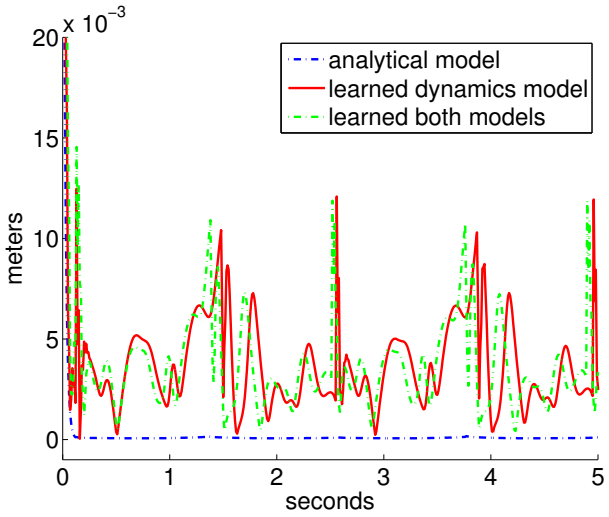


FIGURE 13 – Erreurs de position pour différentes conditions lors de la réalisation de tâches compatibles.

accélérations résultantes au pas de temps suivant

$$\mathcal{ID}_{\text{XCSF}} = \text{XCSF}_{learn}\left(\left[\boldsymbol{q}\left(t\right), \dot{\boldsymbol{q}}\left(t\right)\right], \left[\ddot{\boldsymbol{q}}\left(t+dt\right)\right], \left[\boldsymbol{\tau}^{con}\left(t\right)\right]\right). \tag{30}$$

Les paramètres sont réglés de la manière suivante : le nombre maximum de micro-classeurs est fixé à 1000, la taille minimum des classeurs est fixée à 0.25 dans chaque dimension, il n'y a pas de compaction, le *closest classifier matching* est toujours activé.

Les autres paramètres sont fixés à leur valeur par défaut.

La Figure 14 montre la MAE du couple prédit calculée pendant l'apprentissage. Le nombre de micro-classeurs augmente jusqu'à stabilisation à sa valeur maximale. Contrairement à LWPR, nous n'utilisons ici qu'une seule population de classeurs pour prédire toutes les sorties.

Le modèle dynamique appris avec XCSF peut être utilisé dans la loi de commande décrite en Section 3 sans apprentissage de la cinématique. Les cibles sont toujours positionnées à $[0.20\ 0.50]^T\ m$ et $[0.10\ 1.00]^T\ m$ et sont atteintes avec une précision de 0.01 mètre. K_p est égal à $5s^{-1}$.

Les couples prédits sont calculés de la manière suivante :

$$\boldsymbol{\tau}^{con} = \text{XCSF}_{predict}\left(\mathcal{ID}_{\text{XCSF}}, \left[\boldsymbol{q}, \dot{\boldsymbol{q}}\right], \left[\ddot{\boldsymbol{q}}\right]\right). \tag{31}$$

La trajectoire réalisée évolue jusqu'à atteindre une ligne quasiment droite.

6.2 Compenser des perturbations

Le but de cette section est de montrer que notre architecture peut s'adapter aux perturbations. Les résultats sont présentés avec LWPR comme

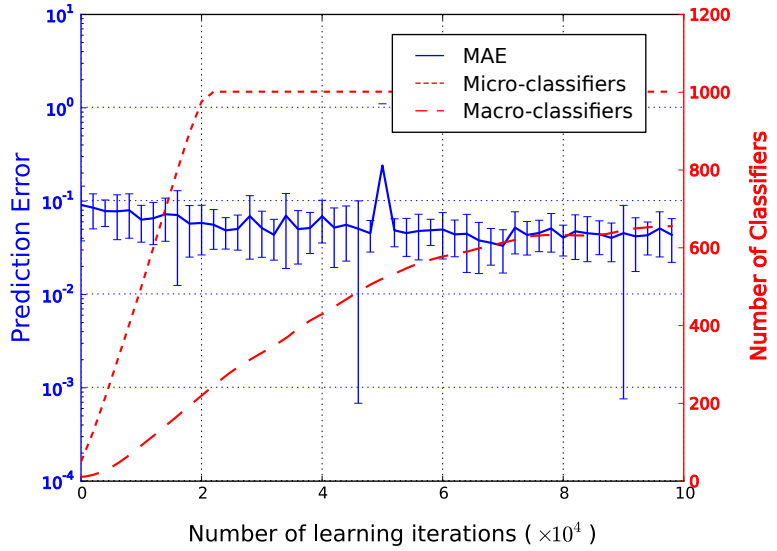


FIGURE 14 – Évolution de la MAE du couple prédit et du nombre de classeurs en fonctions du nombre de données d'entrée en utilisant un modèle appris par XCSF. L'erreur est calculée sur l'axe 1. Les erreurs sur les autres axes sont similaires.

algorithme d'apprentissage. Nous décrivons ensuite pourquoi ces modèles appris sont sujets aux *after-effects* dans le cas où la perturbation est enlevée.

6.2.1 Adaptation aux perturbations aveclwpr

Afin de montrer que notre architecture s'adapte aux perturbations, nous avons commandé un système plan possédant deux degrés de liberté ayant pour unique tâche de positionner $\xi_1^\dagger$ alternativement sur $x = -0.40m$ et $x = 0.40m$ le long de l'axe x_0 avec une précision de $0.05m$ tout en appliquant une force verticale $\boldsymbol{f}$ de $30N$ sur l'organe terminal du robot.

L'apprentissage de la perturbation se fait par l'intermédiaire de l'état du système perturbé. Le modèle dynamique inverse est donc appris comme précédemment :

$$\mathcal{ID}_{\text{LWPR}}^{perturb} = \text{LWPR}_{learn}\left(\left[\boldsymbol{q}\left(t\right), \dot{\boldsymbol{q}}\left(t\right), \ddot{\boldsymbol{q}}\left(t+dt\right)\right], \left[\boldsymbol{\tau}^{con}\left(t\right)\right]\right).$$

À la différence du modèle appris sans perturbation, l'accélération résultante dépend maintenant de la force de perturbation. La dynamique inverse apprise est donc

$$\boldsymbol{\tau}^{con} = A\left(\boldsymbol{q}\left(t\right)\right)\ddot{\boldsymbol{q}}\left(t+dt\right) + \boldsymbol{b}\left(\boldsymbol{q}\left(t\right), \dot{\boldsymbol{q}}\left(t\right)\right) + \boldsymbol{g}\left(\boldsymbol{q}\left(t\right)\right) + \boldsymbol{\epsilon}\left(\boldsymbol{q}\left(t\right), \dot{\boldsymbol{q}}\left(t\right)\right) - \boldsymbol{\tau}^{perturb}. \tag{32}$$

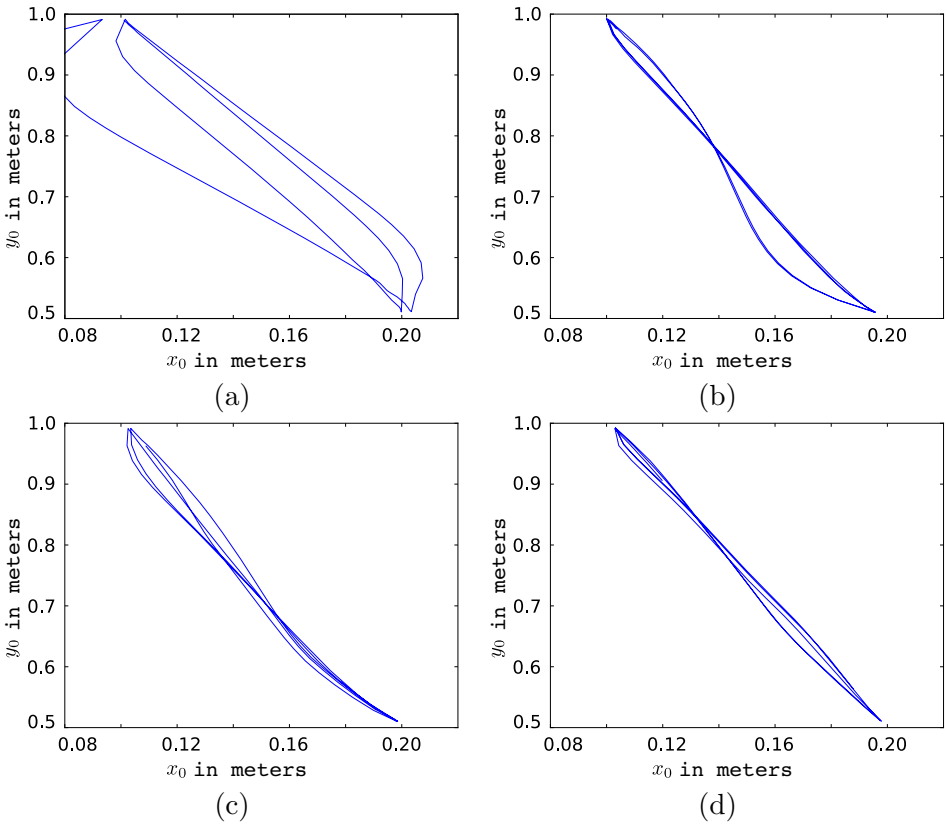


FIGURE 15 – Évolution des trajectoire de l'organe terminal commandé à l'aide d'un modèle appris. Chaque figure représente 4 secondes de simulation.(a) : 0s à 4s.(b) : 24s à 28s.(c) : 52s à 56s.(d) : 76s à 80s.

Le couple $\boldsymbol{\tau}^{con}$ prédit avec la dynamique inverse dépend maintenant de la force appliquée

$$\boldsymbol{\tau}^{con} = \text{LWPR}_{predict}\left(\mathcal{ID}^{perturb}_{\text{LWPR}}, [\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \ddot{\boldsymbol{q}}^{\star}]\right)$$

Pour avoir une meilleure adaptation, le paramètre w_{prune} est fixé à 0.9.

La loi de commande est résumée par l'Algorithme 3.

La Figure 16 représente l'évolution de la déviation induite par la force de perturbation au cours du temps. Lorsque l'apprentissage est mis en route, la perturbation est peu à peu compensée jusqu'à obtenir une ligne droite. Les cinq courbes différentes de la Figure 16 montrent que l'adaptation au cours du temps est possible à l'aide d'un modèle dynamique appris.

Algorithme 3 Boucle de commande avec une force appliquée inconnue.

$\boldsymbol{\nu}_1^\star(t) = K_{p1}\left(\boldsymbol{\xi}_1^\dagger(t) - \boldsymbol{\xi}_1(t)\right)$ *desired operational velocity*

$\dot{\boldsymbol{q}}^\star(t) = \hat{J}_1^+ \boldsymbol{\nu}_1^\star(t)$ *desired joint velocity*

$\ddot{\boldsymbol{q}}^\star(t) = (\dot{\boldsymbol{q}}^\star(t) - \dot{\boldsymbol{q}}(t))/dt$ *desired joint acceleration*

$\boldsymbol{\tau}^{con} = \hat{\mathcal{ID}}(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \ddot{\boldsymbol{q}}^\star(t))$ *desired torques*

$(\boldsymbol{q}(t+dt), \dot{\boldsymbol{q}}(t+dt), \boldsymbol{\xi}(t+dt)) \leftarrow \mathcal{D}\left(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \boldsymbol{\tau}^{con} + \boldsymbol{\tau}^{perturb}\right)$ *integration*

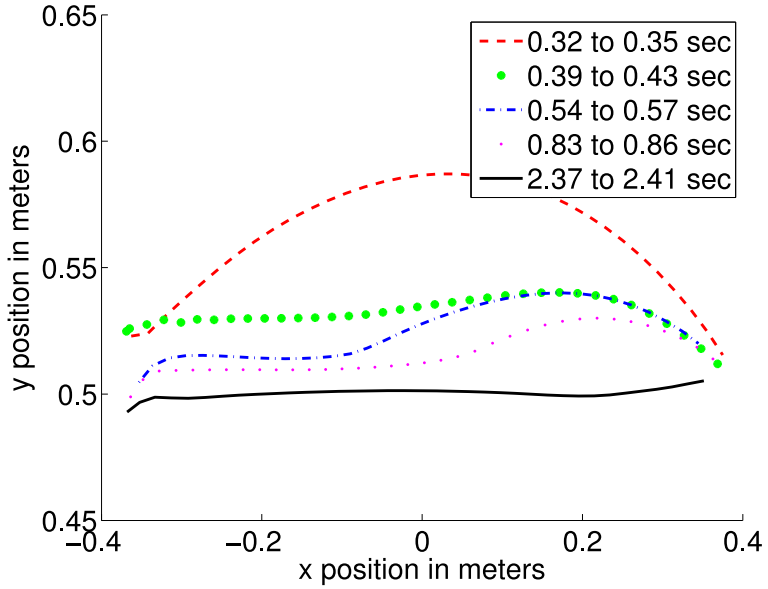


FIGURE 16 – Évolution de la position de l'organe terminal lors de l'application d'une force extérieure, débutant de $x = -0.40m$ et allant à $x = 0.40m$. L'adaptation à la perturbation est bien réalisée. Le correcteur choisi vaut $K_p = 2s^{-1}$.

6.2.2 *After-effects*

Une modèle dynamique incluant la compensation d'une force est perturbé si cette force est supprimée. Le modèle compense toujours la force et la trajectoire est déviée dans le sens inverse de la force qui n'est plus appliquée. Cet effet, présent chez l'être humain, s'appelle *after-effect*. Il est illustré dans [Shadmehr et Mussa-Ivaldi, 1994].

Comme notre loi de commande est fondée sur une modèle dynamique adaptatif, la déviation est présente jusqu'à ce que le modèle apprenne de nouveau le modèle sans perturbation.

La Figure 17 montre l'évolution de la trajectoire utilisant un modèle dynamique appris pendant un mouvement perturbé par une force verticale.

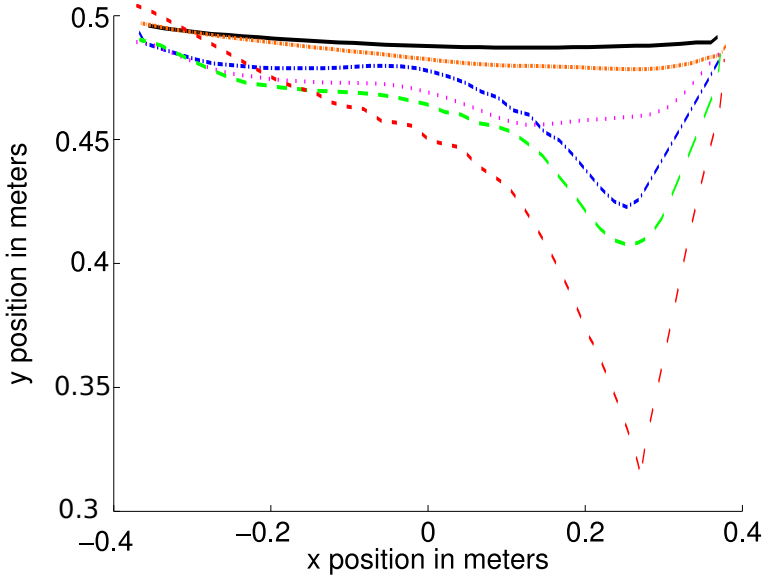


FIGURE 17 – Évolution de l'*after-effect* en utilisant un modèle dynamique inverse appris.

Après retrait de la force, on peut voir que le mouvement est perturbé dans l'autre sens puis s'adapte, petit à petit, jusqu'à retrouver une dynamique non perturbée. La simulation met environ 20 secondes pour retrouver une trajectoire rectiligne.

7 Conclusion

Les robots de service évoluent en interaction avec des utilisateurs. Ces interactions sont rarement modélisées et souvent inconnues par avance. De plus, les robots commandés sont de plus en plus sophistiqués et de plus en plus difficiles à commander. Notre approche consiste à inclure des capacités d'adaptation dans les lois de commande de ces robots pour plus de versatilité.

Nous avons utilisé deux méthodes d'approximation de fonction issues de la littérature afin d'apprendre des modèles géométriques, cinématiques et dynamique de différents systèmes robotiques. Plus précisément, nous avons montré qu'il était possible, à l'aide de modèles appris, de combiner des tâches hiérarchiques même dans le cas où les tâches ne sont pas compatibles. Ceci a été rendu possible par le fait que les modèles sont appris sur l'espace de tous les états possibles et non le long d'une trajectoire comme c'est habituellement le cas. Nous avons expliqué comment obtenir un modèle cinématique sous une forme matricielle en utilisant deux algorithmes d'apprentissage.

Le premier, LWPR, est fondé sur l'apprentissage d'un modèle géométrique qui peut fournir la dérivée de la fonction estimée autrement dit le modèle cinématique. Le second algorithme, XCSF, apprend directement un modèle cinématique direct.

Au niveau expérimental, nous avons validé notre approche sur différents systèmes afin d'étudier différentes propriétés :

- un robot plan à 3 degrés de liberté commandé en couple à l'aide d'une dynamique connue permettant d'illustrer l'apprentissage de la cinématique ainsi que la combinaison de tâches,
- un robot humanoïde commandé en vitesse, nommé ICUB, qui nous a permis de tester notre loi de commande sur un véritable robot,
- un robot plan à 3 degrés de liberté commandé en couple à l'aide d'une dynamique apprise avec LWPR or XCSF,
- un robot plan à 2 degrés de liberté commandé en couple permettant de mettre en évidence des aspects dynamiques tel que l'adaptation à une force de perturbation ainsi que les *after-effects*.

Pour résumé, nous avons atteint notre but qui consistait à proposer une méthode globale d'apprentissage de modèles pour la commande de robots leur donnant ainsi une capacité d'adaptation face à des perturbations inconnues.

Cependant, nous sommes encore loin de résoudre les problèmes exprimés en introduction. Des progrès sur les méthodes d'apprentissage, sur les actionneurs ou les capteurs des robots ainsi que l'utilisation des ces derniers pourraient à l'avenir augmenter considérablement la portée de notre méthode.

Références

BARTHÉLEMY, S. et MICAELLI, A. (2008). Arboris for Matlab.

BEN ISRAEL, A. et GREVILLE, T. (2003). *Generalized inverses : Theory and applications*. Springer, second édition. ISBN 0-387-00293-6.

BUTZ, M., HERBORT, O. et HOFFMANN, J. (2007). Exploiting redundancy for flexible behavior : Unsupervised learning in a modular sensorimotor control architecture. *Psychological Review*, 114(4):1015–1046.

BUTZ, M., PEDERSEN, G. et STALPH, P. (2009). Learning sensorimotor control structures with XCSF : redundancy exploitation and dynamic control. *In Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1171–1178. ACM.

BUTZ, M. V. et HERBORT, . (2008). Context-dependent predictions and cognitive arm control with XCSF. *In Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 1357–1364. ACM New York, NY, USA.

Butz, M. V., Kovacs, T., Lanzi, P. L. et Wilson, S. W. (2004). Toward a theory of generalization and learning in xcs. *IEEE Transactions on Evolutionary Computation*, 8(1):28–46.

Doty, K., Melchiorri, C. et Bonivento, C. (1993). A theory of generalized inverses applied to Robotics. *The International Journal of Robotics Research*, 12(1):1–19.

D'Souza, A., Vijayakumar, S. et Schaal, S. (2001). Learning inverse kinematics. *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 298–303.

Duindam, V. (2006). *Port-Based Modeling and Control for Efficient Bipedal Walking Robots*. Thèse de doctorat, University of Twente.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems : An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press, Ann Arbor, MI.

Jordan, M. I. et Rumelhart, D. E. (1992). Forward models : Supervised learning with a distal teacher. *Cognitive Science*, 16(3):307–354.

Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K. et Hirukawa, H. (2003). Biped walking pattern generation by using preview control of zero-moment point. *In IEEE International Conference on Robotics and Automation, 2003. Proceedings. ICRA*, volume 2, pages 1620–1626.

Kanoun, O., Lamiraux, F., Wieber, P., Kanehiro, F., Yoshida, E. et Laumond, J. (2009). Prioritizing linear equality and inequality systems : application to local motion planning for redundant robots. *In Proceedings of the 2009 IEEE international conference on Robotics and Automation*, pages 724–729. Institute of Electrical and Electronics Engineers Inc., The.

Kavraki, L. et Latombe, J. (1998). Probabilistic roadmaps for robot path planning. *Practical Motion Planning in Robotics : Current Approaches and Future Directions*, 53.

Khalil, W. et Dombre, E. (2002). *Modeling, Identification & Control of Robots*. Butterworth-Heinemann.

Klanke, S., Vijayakumar, S. et Schaal, S. (2008). A library for locally weighted projection regression. *The Journal of Machine Learning Research*, 9:623–626.

Kohonen, T. (2001). *Self-Organizing Maps*. Springer, Berlin.

Landau, I., Lozano, R., M'saad, M., Modestino, J., Fettweis, A., Massey, J., Thoma, M., Sontag, E. et Dickinson, B. (1998). *Adaptive control*. Springer-Verlag New York, Inc. Secaucus, NJ, USA.

Latombe, J. (1991). *Robot Motion Planning*. Kluwer Academic.

Laumond, J. (1998). *Robot motion planning and control*. Springer.

LaValle, S. (2006). *Planning algorithms*. Cambridge Univ Pr.

Liégeois, A. (1977). Automatic supervisory control of the configuration and behavior of multibody mechanisms. *Systems, Man and Cybernetics, IEEE Transactions on*, 7(12):868–871.

Lin, S. et Goldenberg, A. (2001). Neural-network control of mobile manipulators. *IEEE Transactions on Neural Networks*, 12(5):1121–1133.

Ljung, L. (1986). *System identification : theory for the user*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA.

Maciejewski, A. et Klein, C. (1985). Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. *The International Journal of Robotics Research*, 4(3):109–117.

Martinetz, T., Ritter, H. et Schulten, K. (1990). Three-dimensional neural net for learning visuomotor coordination of a robot arm. *IEEE Transactions on Neural Networks*, 1(1):131–136.

Metta, G., Sandini, G., Vernon, D., Natale, L. et Nori, F. (2008). The iCub humanoid robot : an open platform for research in embodied cognition. *In Permis : performance metrics for intelligent systems workshop*, Washington DC, USA.

Mitrovic, D., Klanke, S. et Vijayakumar, S. (2008). Adaptive Optimal Control for Redundantly Actuated Arms. *In From Animals to Animats 10 : 10th International Conference on Simulation of Adaptive Behavior, Sab 2008, Osaka, Japan, July 7-12, 2008, Proceedings*, page 93. Springer.

Narendra, K. et Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1(1):4–27.

Park, K. C., Chang, P.-H. et Lee, S. (2001). Analysis and control of redundant manipulator dynamics based on an extended operational space. *Robotica*, 19(6):649–662.

Ritter, H. J., Martinetz, T. M. et Schulten, K. J. (1989). Topology-conserving maps for learning visuo-motor-coordination. *Neural networks*, 2(3):159–168.

Salaün, C., Padois, V. et Sigaud, O. (2009a). Control of redundant robots using learned models : an operational space control approach. *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 878–885.

Salaün, C., Padois, V. et Sigaud, O. (2009b). A two-level model of anticipation-based motor learning for whole body motion. *In* Pezzulo, G., Butz, M., Sigaud, O. et Baldassarre, G., éditeurs : *Anticipatory Behavior in Adaptive Learning Systems, From Psychological Theories to Artificial Cognitive Systems*, volume 5499 de *Lecture Notes in Computer Science / Lecture Notes in Artificial Intelligence*, pages 229–246. Springer.

Salaün, C., Padois, V. et Sigaud, O. (2010). Learning forward models for the operational space control of redundant robots. *In* Sigaud, O. et Peters, J., éditeurs : *From Motor Learning to Interaction Learning in Robots*, chapitre 8, pages 170–194. Springer.

Sastry, S. et Bodson, M. (1989). *Adaptive control : stability, convergence, and robustness*. Prentice Hall.

Schaal, S., Atkeson, C. G. et Vijayakumar, S. (2002). Scalable techniques from nonparametric statistics for real time robot learning. *Applied Intelligence*, 17(1):49–60.

Selig, J. M. (2005). *Geometric fundamentals of robotics*. Springer Verlag.

Shadmehr, R. et Mussa-Ivaldi, F. (1994). Adaptive representation of dynamics during learning of a motor task. *Journal of Neuroscience*, 14(5): 3208.

Siciliano, B., Sciavicco, L., Villani, L. et Oriolo, G. (2008). *Robotics : modelling, planning and control*. Springer Verlag.

Smits, R., De Laet, T., Claes, K., Soetens, P., De Schutter, J. et Bruyninckx, H. (2008). Orocos : A software framework for complex sensordriven robot tasks. *IEEE Robotics and Automation Magazine*.

Stalph, P., Butz, M. et Pedersen, G. (2009). Controlling a four degree of freedom arm in 3D using the XCSF learning classifier system. *KI 2009 : Advances in Artificial Intelligence*, pages 193–200.

Stramigioli, S. et Bruyninckx, H. (2001). Geometry and Screw Theory for Robotics. *In ICRA 2001*.

Sun, G. et Scassellati, B. (2004). Reaching through learned forward model. *In 4th IEEE/RAS International Conference on Humanoid Robots*, volume 1, pages 93–112, Los Angeles, USA.

Sun, G. et Scassellati, B. (2005). A fast and efficient model for learning to reach. *International Journal of Humanoid Robotics*, 2(4):391–414.

Sutton, R. et Barto, A. (1998). *Reinforcement learning*. MIT Press.

Tikhanoff, V., Fitzpatrick, P., Nori, F., Natale, L., Metta, G. et Cangelosi, A. (2008). The icub humanoid robot simulator. *International Conference on Intelligent RObots and Systems IROS, Nice, France*.

Vijayakumar, S., D'Souza, A. et Schaal, S. (2005). LWPR : A scalable method for incremental online learning in high dimensions. Rapport technique, Edinburgh University Press.

Vijayakumar, S. et Schaal, S. (1997). Local dimensionality reduction for locally weighted learning. *In Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, pages 220–225.

Vijayakumar, S. et Schaal, S. (2000). Locally weighted projection regression : An o (n) algorithm for incremental real time learning in high dimensional space. *In Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford, CA*, pages 1079–1086.

Walter, J. et Schulten, K. (1993). Implementation of self-organizing neural networks for visuo-motor control of an industrial robot. *IEEE Transactions on Neural Networks*, 4(1):86–96.

Whitney, D. E. (1969). Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on Man-Machine Systems*, 10(2): 47–53.

Wieber, P. B. (2006). Trajectory free linear model predictive control for stable walking in the presence of strong perturbations. *In Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 137–142, University of Genova, Genova, Italy.

Wilson, S. W. (2001). Function approximation with a classifier system. *In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 974–981, San Francisco, California, USA. Morgan Kaufmann.