ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
A L'OBTENTION DE LA
MAÎTRISE EN GÉNIE DE LA PRODUCTION AUTOMATISÉE
M. ING.

PAR
BRICE LE BOUDEC

COMMANDE EN TEMPS RÉEL D'UN ROBOT HYPER REDONDANT SOUS ENVIRONNEMENT QNX

MONTRÉAL, LE 9 novembre 2004

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

M. Maarouf Saad, directeur de mémoire
Département du génie électrique à l'école de technologie supérieure

M. Vahé Nerguizian, codirecteur
Département du génie électrique à l'école de technologie supérieure

M. Jean – Pierre Kénné , président du jury
Département du génie mécanique à l'école de technologie supérieure

M. Charles Khairallah, membre du jury
Robotics Design inc.

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 13 octobre 2004

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMMANDE EN TEMPS RÉEL D'UN ROBOT HYPER REDONDANT SOUS ENVIRONNEMENT QNX

Brice Le Boudec

SOMMAIRE

Ce mémoire de maîtrise est la présentation de nos travaux sur la commande adaptative en temps réel du robot hyper redondant ANAT. Nous utilisons la redondance pour l'évitement d'obstacles et la commande sera effectuée sous environnement QNX. Nous présenterons dans ce document la modélisation du robot, la cinématique inverse avec évitement d'obstacles, la commande adaptative, l'interface entre le robot et l'ordinateur et la programmation sous QNX.

La génération de trajectoire avec évitement d'obstacles est réalisée en calculant la cinématique différentielle inverse du robot, basée sur l'inverse généralisé. Une fois cette trajectoire déterminée, nous l'appliquons à un contrôleur adaptatif, basé sur la théorie de Lyapunov, et qui commande le robot ANAT.

Pour ce qui est de l'environnement, nous avons préféré le système d'exploitation QNX à l'utilisation de DSP car nous désirons obtenir une plateforme évolutive pour le développement futur. Bien que le DSP possède une grande capacité de calcul, des limitations sont imposées par son environnement de développement.

Enfin l'interface utilisée entre les moteurs du robot et l'ordinateur sera une interface USB à SPI. Le protocole SPI étant un protocole série, il nous permet de pouvoir modifier le nombre de joints du robot d'une façon pratiquement illimitée et sans avoir à modifier la structure générale de notre système.

REAL TIME CONTROL OF A HYPER REDUNDANT ROBOT USING QNX OS

Brice Le Boudec

ABSTRACT

This document is the presentation of our works on the adaptive control in real time of the hyper redundant robot ANAT. We use the redundancy for the obstacle avoidance and the control will be made under QNX environment. We will see in this document the modeling of the robot, the inverse kinematics with obstacle avoidance, the adaptive control, the interface between the robot and the computer, and the programming under QNX.

The generation of trajectory with obstacle avoidance is realized by calculating the differential inverse kinematics of the robot, based on the generalized inverse. When this trajectory is definite, we apply it to an adaptive controller, based on Lyapunov theory, and who control the ANAT robot.

For the environment system, we preferred operating system QNX to DSP use because we wish to obtain an evolutionary platform for future development. Although the DSP possesses a big capacity of calculation, limitations are imposed by his environment of development. Finally the interface used between the motors of the robot and the computer will be an interface USB to SPI. SPI protocol being a serial protocol, we are able to modify the number of joints of the robot in a practically unlimited way and without having to modify the general structure of our system.

REMERCIEMENTS

Ce document est le premier mémoire de maîtrise issu de l'échange entre l'ESME Sudria, l'école où j'ai reçu ma formation d'ingénieur en France, et l'ETS. Je remercie donc les bureaux des relations internationales de ces deux institutions. Je remercie également Monsieur Laborne directeur de l'ESME Sudria.

Mes sentiments les meilleurs vont à l'égard du Monsieur Maarouf Saad mon directeur de recherche, qui m'a fait part de ses conseils tout au long de ce projet. Monsieur Vahé Nerguizian, mon codirecteur pour ses conseils avisés.

Je remercie également Monsieur Charles Khairallah, PDG de Robotics Design inc, pour avoir mis à disposition le robot ANAT.

Sans oublier Messieurs Handy Blanchette, Guillaume Latombe et Stéphane Bricault que je remercie pour leur aide.

Enfin, je remercie ma famille et mes amis pour leur soutien durant ce projet.

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

LISTE DES ABRÉVIATIONS ET DES SIGLES

a_i	longueur du joint i du robot, m
α_i	angle de torsion du joint i., degré
q_i	valeur du joint i, degré ou m
d_i	distance entre les deux repères i et i+1, m
${}^{i}_{j}T$	matrice de transformation homogène du repère i au repère j
${}^{i}_{j}R$	matrice de rotation du repère i au repère j
Γ	angle de tangage ou pitch, degré
B	angle de roulis ou roll, degré
K	énergie cinétique, J
U	énergie potentielle, J
W	travail, J
τ_i	couple appliqué au joint i, N.m
m_i	représente la masse du corps i, Kg
${}^{j}v_{ci}$	vitesse linéaire du corps i exprimée dans le repère j, m/s
${}^{j}w_{ci}$	vitesse angulaire du corps i exprimée dans le repère j, m/s^2
Q_{nc}	matrice des forces non conservatrices, N
I_i	le tenseur des inerties du corps i
g	accélération gravitationnelle, m/s^2
c_i	centre de masse du corps i
M	matrice de masse du robot
F	vecteur des forces centrifuges, de Coriolis, de frottement et de gravité
${}^{i}J_{j}$	matrice jacobienne du repère i au repère j
$\dot{x}$	le vecteur des vitesses, m/s^2
${}^{i}J_{j}^{+}$	inverse généralisé du repère i au repère j
N	espace nul de ${}^{0}J_{7}$

ξ	vecteur responsable du mouvement interne du robot
V(q)	fonction objective
S(x,y,z)	fonction de surface
d_s	distance de sécurité, m
l	nombre de contraintes associées à la fonction objective
P	fonction de pénalité
f_{obj1}	partie de la fonction objective responsable de l'évitement d'obstacle
f_{obj2}	partie de la fonction objective imposant les limitations physiques du robot
f_{obj3}	partie de la fonction objective évitant au robot de heurter sa partie prismatique
n	nombre d'obstacles
m	nombre de points placés sur le robot et à partir desquelles l'algorithme réalise l'évitement d'obstacle
(x_j, y_j, z_j)	coordonnées de l'articulation j dans le repère de la base
$(x_{ci}, y_{ci}, r_i, h_i)$	coordonnées du cylindre j dans le repère de base
r_{si}, h_{si}	représentent respectivement le rayon et la hauteur de sécurité entre les articulations i et le cylindre j
a	vecteur des paramètres dynamiques
â	vecteur des paramètres adaptatifs
F_f	vecteur de frottement visqueux
Y_r	matrice d'adaptation
Kp	gain proportionnel du contrôleur adaptatif
Kd	gain dérivatif du contrôleur adaptatif
Ga	gain adaptatif
V	tension appliquée aux bornes du moteur, V
E	représente la force électromotrice du moteur, V
R	la résistance des armatures du moteur, Ohm

I	le courant absorbé par l'induit
Φ	flux dans l'induit
Ω	vitesse de rotation du moteur, rd/s
ANAT	Articulated Nimble Adaptable Trunk
DSP	processeur de traitement numérique des signaux (Digital Signal Processor)
SPI	Serial Periferal Interface
MISO	Master In Slave Out
MOSI	Master Out Slave In
CLK	Clock
USB	Universal Serial Bus
PWM	Pulse Width Modulation

INTRODUCTION

Le terme robot a été introduit en 1920 par le romancier tchèque Capek, désigne une machine androïde capable de remplacer l'homme au travail. Aujourd'hui, la robotique touche différents domaines de recherche comme : la vie artificielle pour rendre les robots plus autonomes et leur permettre d'évoluer sans intervention de l'homme; l'intelligence collective, où des robots interagissent entre eux ; les nanotechnologies pour concevoir des micro-robots, utilisables à l'intérieur du corps humain par exemple. Comme nous pouvons le voir, les utilisations sont variées et pratiquement illimitées. Mais revenons un instant sur une définition du terme robot : « Conçus pour exécuter certaines tâches à la place de l'homme (intervenir dans une chaîne de fabrication industrielle, se déplacer sur Mars ou dans des endroits dangereux...), les robots sont plus que de simples ordinateurs : ils doivent être capables de percevoir ce qui les entoure et de réagir en conséquence ». Pour ce projet, nous avons choisi un robot modulaire hyper redondant, c'est-à-dire un robot qui possède un mouvement interne. Ce mouvement interne n'influe pas sur la trajectoire de l'effecteur. Il permet ainsi au robot de réaliser une tâche auxiliaire comme : éviter des obstacles, optimiser la dépense d'énergie du robot ou encore accéder à un environnement difficile.

L'autre terme important de ce sujet est « le temps réel ». Reprenons la définition de Luc Aleaume et Laurent Grzybek [Phillipe, 2001] : « Le temps réel est l'aptitude d'une application à répondre à des sollicitations de l'environnement contrôlé, en des temps de réponse en relation avec la constante de temps dominante dans cet environnement ». En fait, l'application contrôlant le système doit être capable de le faire dans un laps de temps garanti, et ainsi rendre le système déterministe. Enfin pour terminer d'illustrer ce concept, citons cette définition [Salvetti, 2002] : « Un système temps réel n'est pas nécessairement rapide, mais il est capable de donner l'illusion de la réalité ».

Nous allons donc, dans ce projet, commander un robot hyper redondant en temps réel et comme nous l'avons précisé sous environnement QNX, choix que nous justifierons plus tard. Enfin la redondance sera utilisée dans notre cas pour réaliser de l'évitement d'obstacle.

Le robot que nous commandons utilise l' « Articulated Nimble Adaptable Trunk » technologie, dite ANAT. Cette technologie permet d'éviter les obstacles et offre également une grande robustesse. Enfin son aspect le plus intéressant est sa modularité, en effet le robot est conçu de manière à ce que nous puissions faire varier le nombre de joints du robot. Il sera donc important de tenir compte de cela pour créer un programme et une interface, entre le robot et l'ordinateur, aussi modulaire que possible.

REVUE DE LITÉRATURE

Dans cette partie, nous présentons le travail préliminaire nécessaire à la réalisation de ce projet. Nous détaillons le choix des méthodes permettant de mener à bien les différentes étapes de ce travail afin de dégager la problématique. Nous insistons sur deux points, la redondance et le temps réel.

1 Algorithmes de commande et redondance

Nous parlons de redondance lorsque le nombre de degrés de liberté du manipulateur est supérieur au nombre de degrés de liberté de la tâche à effectuer. Lorsqu'il y a redondance, il existe une infinité de positions possibles pour un placement fixe de l'outil ; ainsi certains membres du manipulateur peuvent être en mouvement, nous appelons cela communément, le mouvement interne du robot. La redondance permet d'optimiser les performances d'un robot et de s'affranchir de contraintes imposées par les limites articulaires ou les obstacles.

La trajectoire imposée est plus naturellement exprimée dans l'espace cartésien, appelé aussi espace opérationnel, ainsi il est plus simple de concevoir des lois de commande directement dans cet espace. Il faut donc passer de l'espace cartésien vers l'espace articulaire en effectuant des transformations géométriques et cinématiques. Le modèle géométrique permet de déterminer la relation entre les positions articulaires et cartésiennes, le modèle cinématique permet de décrire la relation de vitesse entre ces deux espaces. Comme nous l'avons dit précédemment, pour les robots redondants, une infinité de solutions possibles existe pour une position de l'effecteur donné.

Plusieurs travaux ont déjà été effectués dans ce domaine. Une première méthode est basée sur des approches d'optimisation. Pour exprimer les vitesses dans l'espace des joints en fonction des vitesses dans l'espace cartésien, nous utilisons l'inverse généralisé

auquel est ajouté un terme. Ce terme est composé d'un vecteur multiplié par l'espace nul de la matrice jacobienne. Ce vecteur est en général le gradient d'une fonction de coût permettant d'optimiser le déplacement, (évitement d'obstacle, minimisation de la vitesse des articulations du système...). Cette technique offre un suivi de trajectoire de l'effecteur fiable en lui assurant une priorité plus importante qu'à l'évitement d'obstacle, qui est réalisée avec beaucoup d'efficacité [1, 2]. Un des intérêts de cette méthode est qu'elle génère une trajectoire, dans l'espace des joints, à suivre par le robot et réalise aussi la tâche auxiliaire (dans notre cas l'évitement d'obstacle) dès la cinématique inverse. Il ne reste alors, qu'à appliquer la commande que l'on souhaite.

Une autre méthode permettant de réaliser la cinématique inverse en même temps que l'évitement d'obstacles est basée sur les travaux de John Baillieul [3, 4, 5, 6]. Elle consiste à introduire la projection du gradient de la fonction objective sur l'espace nul, dans un vecteur étendu de l'espace cartésien. Cette méthode permet d'obtenir une matrice Jacobienne carrée et ainsi de pouvoir réaliser une cinématique inverse classique. Cette méthode permet, elle aussi, de réaliser l'évitement d'obstacle dès la cinématique inverse. Mais, on remarque qu'elle ne réalise pas un suivi de trajectoire aussi précis que la méthode précédente. De plus, le temps de calcul de la cinématique inverse est nettement supérieur, c'est pourquoi nous ne choisirons pas cette méthode.

D'autres méthodes d'optimisation sont aussi utilisées, ainsi nous trouvons dans la littérature des articles utilisant des algorithmes génétiques [7], des réseaux de neurones [8] ou encore des systèmes flous [9] pour résoudre ce problème.

La contrainte du temps réel n'est pas favorable à l'utilisation de ces trois dernières méthodes. Bien que moins difficile à implémenter, elles posent souvent des problèmes de stabilité lorsqu'on les intègre dans des boucles de commande, c'est pourquoi nous allons les écarter pour la suite du développement.

La dernière méthode que nous présentons utilise des forces externes généralisées qui agissent sur le manipulateur afin d'éviter des obstacles. En fait, un champ de force potentielle est créé autour des obstacles pour empêcher le robot d'entrer en collision. Là aussi Mr Léon Zlajpah [10] a rencontré certains problèmes lors d'application réelle et notamment une convergence lente lorsque les forces externes sont nulles.

Nous allons donc baser notre travail sur la première méthode présentée, c'est à dire par la méthode de l'inverse généralisée.

2 Système d'exploitation et temps réel

Un système d'exploitation non temps réel ne permet pas d'avoir un délai de réponse fiable pour une application donnée et ce, à cause de toutes les interruptions que subit le système par des programmes qui fonctionnent en même temps que l'application.
Prenons l'exemple suivant [Blanchier, 2001] : Mesurons sur le port parallèle la fréquence d'un signal carré de période souhaité 25 Hz. Pour un système classique la période varie entre 22 et 29 Hz alors que, pour un système temps réel, elle varie entre 24.98 et 25.01 Hz. Le système temps réel parvient à assurer une fréquence beaucoup plus stable. Nous avons alors un laps de temps garanti (pour rendre le système temps déterministe) beaucoup plus court avec un système temps réel.

Plusieurs systèmes d'exploitation temps réel sont disponibles sur le marché tel que Linux temps réel, Windows temps réel ou encore QNX. Nous avons choisi QNX car ce système est « open source » et offre une documentation et un environnement de développement agréable, une excellente fiabilité, un bon temps de réponse. Enfin, c'est un système d'exploitation très populaire dans le monde de la robotique et il supporte de nombreux processeurs.

Fondé en 1980 par Gordon Bell et Dan Dodge, deux étudiants de l'University of Waterloo au Canada, QNX est rapidement devenu le leader parmi les systèmes d'exploitation dédiés à « l'embarqué ». QNX est un système d'exploitation temps réel UNIX ultraperformant, doté d'un micro noyau (Neutrino) et d'une interface graphique (Photon). On le retrouve dans divers produits embarqués de Cisco, Johnson Controls, Siemens, Alcatel, Texaco ou encore Ford.

Pour offrir un temps de réponse déterministe et une excellente fiabilité, QNX s'appuie sur son architecture micro noyau. Ce noyau n'implémente que les services fondamentaux (communication inter tâches, ordonnancement, gestion des interruptions). Les services de haut niveau (fichier, entrées/sorties, réseau) et les pilotes sont implémentés dans des processus séparés.

Ce système d'exploitation nous offre toutes les qualités requises à l'application de notre travail. C'est donc lui que nous choisissons.

CHAPITRE 1

MODÉLISTION DU ROBOT ANAT

Toute commande ou contrôle d'un robot passe par la modélisation de celui-ci. Cette modélisation comprend différentes phases : tout d'abord la détermination des caractéristiques physiques du robot, ensuite nous déterminons la cinématique du robot après avoir calculé les matrices de transformation homogène. Enfin nous allons voir la modélisation dynamique.

La cinématique et la cinématique différentielle nous permettent de traduire une trajectoire exprimée dans l'espace de travail, en une trajectoire exprimée dans l'espace des articulations. Nous entendons par trajectoire, la position, la vitesse et l'accélération désirée pour réaliser une certaine tâche. Ainsi nous serons capables, pour réaliser un triangle par exemple, de déterminer pour chaque articulation la position, la vitesse et l'accélération à suivre. Nous allons voir dans ce chapitre seulement la cinématique, la cinématique différentielle sera vue au chapitre suivant car c'est avec elle que nous réalisons l'évitement d'obstacle en utilisant la redondance du robot. Nous avons donc décidé de la détailler dans le chapitre trois.

La modélisation dynamique permet elle d'associer à cette trajectoire désirée dans l'espace de joints, un couple à appliquer aux moteurs pour effectuer le déplacement. Plusieurs méthodes s'offrent à nous, la méthode de Lagrange, de Newton Euler ou encore celle des graphes de liaison, nous avons choisi celle de Lagrange car elle est classique et nous en sommes familiers.

1.1 Présentation générale du robot ANAT

La figure 1 présentée ci-après, nous montre la schématisation du robot. Il possède sept degrés de liberté composés comme suit : un premier joint prismatique, une partie redondante composée de trois joints rotatifs parallèles. Enfin, un effecteur capable d'orienter l'outil dans une position désirée et composé de trois joints rotatifs dont les axes de rotations sont perpendiculaires les uns par rapport aux autres.

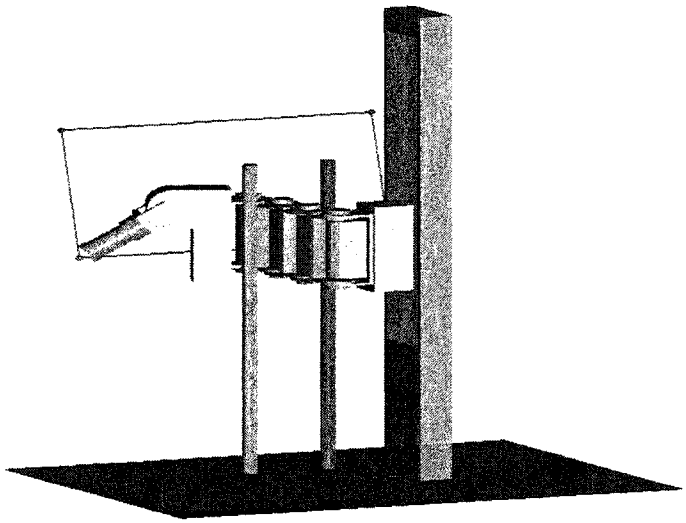


Figure 1 Le robot ANAT

1.1.1 Espace de travail du robot

Le robot est soumis à quelques contraintes de déplacement mécaniques qui sont représentées dans le tableau 1.1.

Tableau I

Espace de travail du robot

Articulation	Type	Espace
1	Prismatique	De L_0 = 0.57m à 1.27m
2	Rotative	De -90° à +90°
3	Rotative	De -90° à +90°
4	Rotative	De -90° à +90°
5	Rotative	De -90° à +90°
6	Rotative	De 0° à 180°
7	Rotative	-nπ à nπ

1.1.2 La redondance du robot

Comme défini dans le chapitre précédent, nous parlons de redondance lorsque le nombre de degrés de liberté du manipulateur est supérieur au nombre de degrés de liberté de la tâche à effectuer. Dans le cas du robot ANAT, les joints rotatifs parallèles, placés entre le joint prismatique et l'effecteur, composent la partie redondante. Cette partie est modulable, c'est-à-dire que l'on peut ajouter ou retirer des joints rotatifs. Nous verrons par la suite, et notamment lorsque nous réaliserons la cinématique différentielle inverse, comment nous utiliserons la redondance du robot pour ce projet.

1.2 Cinématique directe et inverse

Cette partie va permettre de traduire la position du robot de l'espace de travail à celui des joints. Nous allons dans un premier temps attribuer les repères nécessaires au calcul

des matrices de transformation homogène qui permettent elles-mêmes de déterminer la cinématique du robot.

1.2.1 Attribution des repères

Une méthode systématique d'attribution des repères doit être appliquée si l'on désire obtenir les paramètres de Craig permettant de déterminer les matrices de transformations homogènes. Cette méthode est illustrée sur la figure 2. Sur cette figure, les axes z des différents repères sont alignés avec leurs articulations respectives. De plus l'axe x est choisi de façon à être perpendiculaire à l'axe z du repère auquel il appartient et à l'axe z du repère suivant. Enfin l'axe y est déterminé en respectant la règle de la main droite.

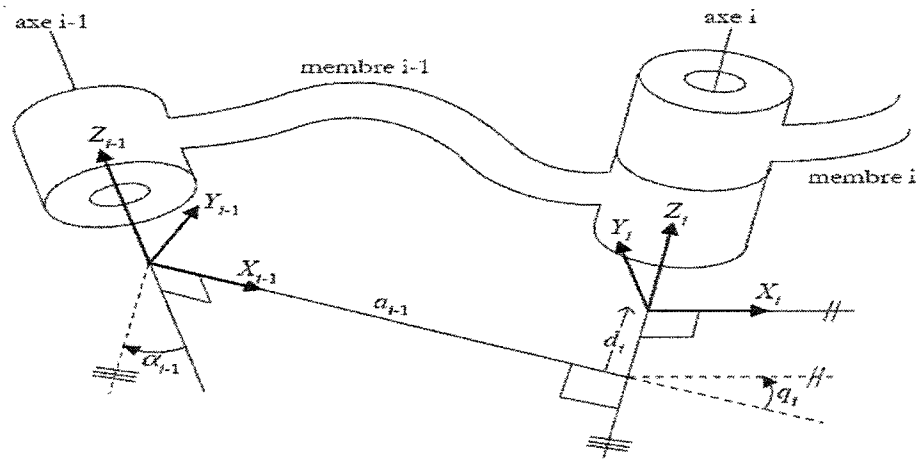


Figure 2 Paramètres de transformation du membre i-1 au membre i

Dans le cas de notre robot, nous fixons les repères suivant cette méthode. La figure 3 indique les systèmes d'axes choisis.

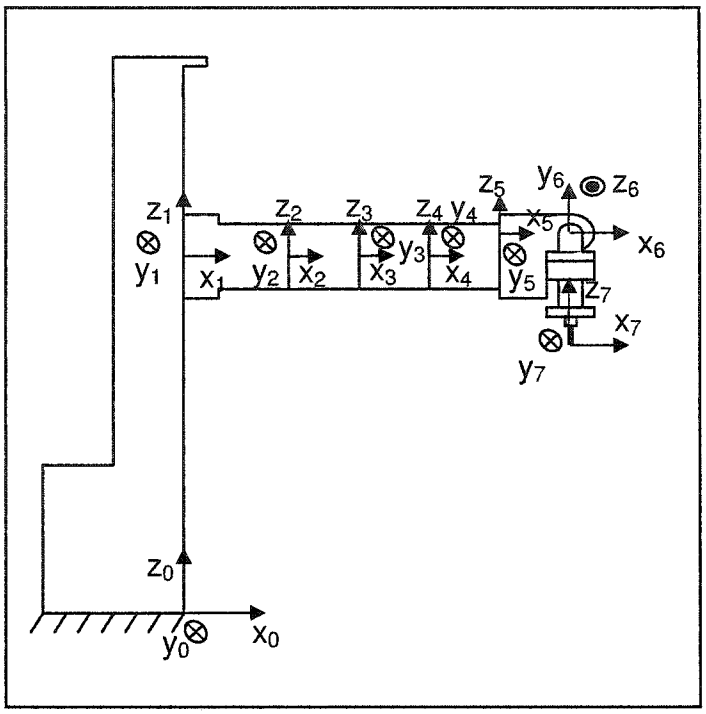


Figure 3 Positionnement des systèmes d'axes du robot ANAT

1.2.2 Matrices de transformations homogènes

La transformation homogène d'un membre au précédent peut être entièrement caractérisée par quatre paramètres appelés paramètres de Denavit-Hartenberg. Ils permettent d'obtenir de façon systématique la matrice de transformation d'un membre à l'autre. Ces paramètres sont attribués selon la convention de Craig et illustrée par la figure 2. Ces paramètres sont définis de la façon suivante :

- Le paramètre a_{i-1} est une translation entre le repère i-1 et le repère i le long de l'axe X_{i-1}.
- Le paramètre α_{i-1} est une rotation autour de l'axe X_{i-1} qui permet de rendre l'axe Z_{i-1} parallèle à l'axe Z_i.
- Le paramètre d_i est une translation entre l'intersection des prolongations des axes X_{i-1} et Z_i et le repère i.

- Le paramètre q_i est une rotation autour de l'axe Z_i qui permet de rendre l'axe X_{i-1} parallèle à l'axe X_i.

Tableau II

Paramètres de Denavit-Hartenberg du robot ANAT

articulation	α_{i-1}	a_{i-1}	d_i	q_i
1	0	0	q_1	0
2	0	L_1	0	q_2
3	0	L	0	q_3
4	0	L	0	q_4
5	0	L	L_2	q_5
6	$\pi/2$	L_3	0	q_6
7	$-\pi/2$	0	$-L_4$	q_7

Selon la figure 2, la matrice de transformation homogène du repère i au repère i-1 est donnée par la relation suivante :

$$ {}^{i-1}_{i}T = \begin{bmatrix} cq_i & -sq_i & 0 & a_{i-1} \\ sq_i c\alpha_{i-1} & cq_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ sq_i s\alpha_{i-1} & cq_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Où : $cq_i = \cos(q_i)$ *et* $sq_i = \sin(qi)$

Nous avons calculé les matrices de passages à l'aide du logiciel de calcul Maple et d'une bibliothèque conçue à cette effet par Bigras [12]. Elles sont présentées dans l'annexe 1.

Voici la matrice de passage du repère de base 0 au dernier repère 7 qui va nous permettre de déterminer la cinématique du robot :

$$ {}^{0}_{7}T = \begin{bmatrix} {}^{0}_{7}T_{11} & {}^{0}_{7}T_{12} & {}^{0}_{7}T_{13} & {}^{0}_{7}T_{14} \\ {}^{0}_{7}T_{21} & {}^{0}_{7}T_{22} & {}^{0}_{7}T_{23} & {}^{0}_{7}T_{24} \\ {}^{0}_{7}T_{31} & {}^{0}_{7}T_{32} & {}^{0}_{7}T_{33} & {}^{0}_{7}T_{34} \\ {}^{0}_{7}T_{41} & {}^{0}_{7}T_{42} & {}^{0}_{7}T_{43} & {}^{0}_{7}T_{44} \end{bmatrix} $$

$$ {}^{0}_{7}T_{11} = \frac{1}{4}\left[c_{234567} + c_{23457-6} + c_{23456-7} + c_{2345-6-7}\right] + \frac{1}{2}\left[-c_{2345-7} + c_{23457}\right] $$

$$ {}^{0}_{7}T_{12} = \frac{1}{4}\left[-s_{234567} - s_{23457-6} + s_{23456-7} + s_{2345-6-7}\right] - \frac{1}{2}\left[s_{23457} + s_{2345-7}\right] $$

$$ {}^{0}_{7}T_{13} = \frac{1}{2}\left[-s_{23456} + s_{2345-6}\right] $$

$$ {}^{0}_{7}T_{14} = \frac{1}{2}L_4\left[s_{23456} - s_{2345-6}\right] + L_3 c_{2345} + L\left[c_{234} + c_{23} + c_2\right] + L_1 $$

$$ {}^{0}_{7}T_{21} = \frac{1}{4}\left[s_{234567} + s_{23457-6} + s_{23456-7} + s_{2345-6-7}\right] + \frac{1}{2}\left[-s_{2345-7} + s_{23457}\right] $$

$$ {}^{0}_{7}T_{22} = \frac{1}{4}\left[c_{234567} + c_{23457-6} - c_{23456-7} - c_{2345-6-7}\right] + \frac{1}{2}\left[c_{23457} + c_{2345-7}\right] $$

$$ {}^{0}_{7}T_{23} = \frac{1}{2}\left[c_{23456} - c_{2345-6}\right] \qquad (1.1) $$

$$ {}^{0}_{7}T_{24} = \frac{1}{2}L_4\left[-c_{23456} + c_{2345-6}\right] + L_3 s_{2345} + L\left[s_{234} + s_{23} + s_2\right] $$

$$ {}^{0}_{7}T_{31} = \frac{1}{2}\left[s_{67} + s_{6-7}\right] $$

$$ {}^{0}_{7}T_{32} = \frac{1}{2}\left[c_{67} - c_{6-7}\right] $$

$$ {}^{0}_{7}T_{33} = c_6 $$

$$ {}^{0}_{7}T_{34} = -L_4 c_6 + L_2 + q_1 $$

$$ {}^{0}_{7}T_{41} = 0 $$

$$ {}^{0}_{7}T_{42} = 0 $$

$$ {}^{0}_{7}T_{43} = 0 $$

$$ {}^{0}_{7}T_{44} = 1 $$

1.2.3 Expression de la cinématique directe et inverse

Nous obtenons donc, grâce à la matrice de transformation homogène ${}^{0}_{7}T$, la position de l'effecteur dans le repère de la base :

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} L_4 c_{2345} s_6 + L_3 c_{2345} + L\left[c_{234} + c_{23} + c_2\right] + L_1 \\ L_4 s_{2345} s_6 + L_3 s_{2345} + L\left[s_{234} + s_{23} + s_2\right] \\ -L_4 c_6 + L_2 + q_1 \end{bmatrix} \tag{1.2}$$

De plus, en identifiant la matrice de rotation ${}^{0}_{7}R$, sous matrice de la matrice de transformation homogène ${}^{0}_{7}T$, avec celle des angles d'Euler, $Rot(Z,\gamma)Rot(Y,\beta)Rot(Y,\alpha)$, nous pouvons obtenir les angles d'Euler. Ces angles sont en fait la rotation entre les repères 0 et 7 suivant les axes X, Y et Z.

$${}^{0}_{7}R = \begin{bmatrix} {}^{0}_{7}T_{11} & {}^{0}_{7}T_{12} & {}^{0}_{7}T_{13} \\ {}^{0}_{7}T_{21} & {}^{0}_{7}T_{22} & {}^{0}_{7}T_{23} \\ {}^{0}_{7}T_{31} & {}^{0}_{7}T_{32} & {}^{0}_{7}T_{33} \end{bmatrix} = \begin{bmatrix} c_\gamma c_\beta c_\alpha - s_\gamma s_\alpha & -c_\gamma c_\beta s_\alpha - s_\gamma s_\alpha & -c_\gamma s_\beta \\ c_\gamma c_\beta c_\alpha - s_\gamma s_\alpha & -s_\gamma c_\beta s_\alpha + c_\gamma c_\alpha & -s_\gamma s_\beta \\ s_\beta c_\alpha & -s_\beta s_\alpha & c_\beta \end{bmatrix} \tag{1.3}$$

Ainsi, nous obtenons les angles d'Euler :

$$\alpha = q_7 \qquad \beta = q_6 \qquad \gamma = q_2 + q_3 + q_4 + q_5 \tag{1.4}$$

À partir des formules (1.2) et (1.4) nous obtenons les relations de cinématique inverse suivantes :

$$\begin{bmatrix} q_6 \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_7 \end{bmatrix} = \begin{bmatrix} \beta \\ z + L_4 c_6 - L_2 \\ \gamma - q_3 - q_4 - q_5 \\ \gamma - q_2 - q_4 - q_5 \\ \gamma - q_2 - q_3 - q_5 \\ \gamma - q_2 - q_3 - q_4 \\ \alpha \end{bmatrix} \tag{1.5}$$

1.3 Modélisation dynamique du robot ANAT selon la méthode de Lagrange

Dans cette partie, nous faisons un bref rappel théorique des équations de Lagrange ainsi que des énergies potentielles et cinétiques nécessaires à l'application de cette méthode. Ensuite, nous donnons la forme générale de la modélisation dynamique d'un robot, et enfin, nous l'appliquons, pour réaliser la modélisation dynamique du robot ANAT.

1.3.1 Équations de Lagrange

L'énergie cinétique K dépendant des vitesses des articulations q_i. La variation de l'énergie cinétique prend la forme suivante :

$$\delta K = \left(\frac{\partial K}{\partial q} - \frac{d}{dt} \frac{\partial K}{\partial \dot{q}} \right) \delta q \tag{1.6}$$

La variation de travail prend la forme suivante :

$$\delta W = \sum F^T \delta x = \sum F_i^T \frac{\partial f}{\partial q} \delta q = Q^T \delta q \tag{1.7}$$

En sommant ces deux variations et en les intégrant entre les temps t_1 et t_2, nous obtenons l'expression suivante :

$$\int_{t1}^{t2} (\delta K + \delta W) dt = \int_{t1}^{t2} \left(\frac{\partial K}{\partial q} - \frac{d}{dt} \frac{\partial K}{\partial \dot{q}} + Q^T \right) \delta q dt \tag{1.8}$$

Selon le principe d'Hamilton, en annulant cette variation, on obtient la dynamique du système. La variation δq étant arbitraire, l'intégrale s'annule si et seulement si l'équation de Lagrange est respectée, c'est-à-dire si :

$$\frac{d}{dt} \frac{\partial K^T}{\partial \dot{q}} - \frac{\partial K^T}{\partial q} - Q = 0 \tag{1.9}$$

Où : $Q = -\frac{dU}{dq} + \tau + Q_{nc}$ avec :

- U représentant l'énergie potentielle.
- τ représentant le couple appliqué au robot.
- Q_{nc} représentant les forces non conservatrices (ex : les frottements).

1.3.2 Représentation de l'énergie cinétique

L'énergie cinétique d'un corps i est donnée par la relation suivante :

$$K_i = \frac{1}{2} \int {}^{ci}v_x^T \, {}^{ci}v_x \, dm_i \tag{1.10}$$

Avec : ${}^{ci}v_x$ la vitesse d'un élément de masse dm_i situé à une distance x du centre d'inertie c_i du corps i où est situé le repère de l'objet.

Après intégration, nous obtenons l'énergie cinétique d'un corps i :

$$K_i = \frac{1}{2}\left(m_i v_{ci}^T v_{ci} + {}^{ci}w_{ci}^T I_i \, {}^{ci}w_{ci}\right) \tag{1.11}$$

Où :

- m_i représente la masse du corps i.
- v_{ci} représente la vitesse linéaire du corps i exprimée dans le repère 0.
- ${}^{ci}w_{ci}$ représente la vitesse angulaire du corps i exprimée dans le repère i.
- I_i représente le tenseur des inerties. Si l'orientation du repère du corps est choisie de façon à ce qu'il corresponde avec les axes principaux de ce même corps, alors le tenseur d'inertie a pour valeur :

$$I_i = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$$

1.3.3 Représentation de l'énergie potentielle

L'énergie potentielle peut être exprimée sous la forme suivante :

$$U_i = -\int {}^0 g^T \, {}^0 x dm_i \tag{1.12}$$

Où :

- ${}^0 g^T$ représente le vecteur gravité exprimé dans le repère de base.
- ${}^0 x$ représente la position d'une unité de masse dm_i.

Si le repère est situé au centre de masse c_i, U_i à pour valeur :

$$U_i = -{}^0 g^T \, {}^0 c_i m_i$$

1.3.4 Forme générale du modèle dynamique

La forme générale d'une modélisation dynamique obtenue par la méthode de Lagrange s'écrit comme suit :

$$M(q)\ddot{q} + F(q,\dot{q}) = \tau \tag{1.13}$$

Où :

- M représente la matrice de masse.
- F le vecteur des forces centrifuges, de Coriolis, de frottement et de gravité.
- τ le couple appliqué aux joints du robot.

1.3.5 Application au robot ANAT

1.3.5.1 Définition du vecteur de gravité et du tableau des spécifications dynamiques

Pour notre robot, nous considérons que le centre de masse des différents membres est situé à l'extrémité de ceux-ci. Ainsi nous obtenons le tableau des spécifications dynamiques sur la page suivante.

Tableau III

Tableau des spécifications dynamiques

articulation	cm_x	cm_y	cm_z	m	I_{xx}	I_{yy}	I_{zz}
1	L_1	0	0	m_1	I_{xx1}	I_{yy1}	I_{zz1}
2	L	0	0	m_2	I_{xx2}	I_{yy2}	I_{zz2}
3	L	0	0	m_3	I_{xx3}	I_{yy3}	I_{zz3}
4	L	0	0	m_4	I_{xx4}	I_{yy4}	I_{zz4}
5	L_3	0	L_2	m_5	I_{xx5}	I_{yy5}	I_{zz5}
6	0	$-L_4$	0	m_6	I_{xx6}	I_{yy6}	I_{zz6}
7	0	0	$-L_5$	m_7	I_{xx7}	I_{yy7}	I_{zz7}

L'expression du vecteur de gravité, dans le repère de la base, prend la forme suivante :

$$g = [0 \quad 0 \quad -g]^T$$

1.3.5.2 Énergie cinétique du robot ANAT

L'énergie cinétique du robot ANAT est égale à la somme de l'énergie cinétique des différents membres du robot.

$$K = \sum_{i=1}^{7} \frac{1}{2} \left(m_i v_{ci}^T v_{ci} + {}^{ci}w_{ci}^T I_i {}^{ci}w_{ci} \right) \tag{1.14}$$

1.3.5.3 Énergie potentielle du robot ANAT

L'énergie potentielle du robot est égale à la somme de l'énergie potentielle des différents membres.

$$U = -\sum_{i=1}^{7} {}^0g^T {}^0c_i m_i$$

Après calcul nous trouvons l'expression suivante de l'énergie potentielle :

$$U = g[q_1(m_1+m_2+m_3+m_4+m_5+m_6+m_7)+c_6(m_6L_4-m_7(L_4+L_5)) + L_2(2m_5+m_6+m_7)] \quad (1.15)$$

1.3.5.4 Forme générale du modèle dynamique du robot ANAT

Après avoir appliqué le principe d'Hamilton en utilisant les expressions des énergies potentielle et cinétique, et après avoir mis cette équation sous la forme générale, nous obtenons les expressions de M et F.

$$M(q)=\begin{bmatrix} M_{11} & M_{21} & M_{31} & M_{41} & M_{51} & M_{61} & M_{71} \\ M_{21} & M_{22} & M_{32} & M_{42} & M_{52} & M_{62} & M_{72} \\ M_{31} & M_{32} & M_{33} & M_{43} & M_{53} & M_{63} & M_{73} \\ M_{41} & M_{42} & M_{43} & M_{44} & M_{54} & M_{64} & M_{74} \\ M_{51} & M_{52} & M_{53} & M_{54} & M_{55} & M_{65} & M_{75} \\ M_{61} & M_{62} & M_{63} & M_{64} & M_{65} & M_{66} & M_{76} \\ M_{71} & M_{72} & M_{73} & M_{74} & M_{75} & M_{76} & M_{77} \end{bmatrix} \qquad F(q,\dot{q})=\begin{bmatrix} F_{11} \\ F_{21} \\ F_{31} \\ F_{41} \\ F_{51} \\ F_{61} \\ F_{71} \end{bmatrix}$$

Maintenant que nous avons réalisé la modélisation du robot, nous allons générer la trajectoire avec évitement d'obstacle. Puis, calculer la commande à appliquer au robot afin d'effectuer des simulations.

CHAPITRE 2

CINÉMATIQUE DIFFÉRENTIELLE INVERSE ET ÉVITEMENT D'OBSTACLE

2.1 Cinématique différentielle inverse

Le robot ANAT étant redondant, la matrice jacobienne est rectangulaire, nous ne pouvons donc pas calculer la cinématique différentielle inverse car il nous est impossible d'inverser la matrice jacobienne. Aussi, deux solutions s'offrent à nous :

- Augmenter l'espace de tâche afin de rendre la matrice jacobienne carrée.
- Obtenir la solution à l'aide d'un inverse généralisé de la matrice jacobienne

La méthode d'augmentation de la tâche permet de résoudre facilement le problème d'inversion de la cinématique différentielle pour un manipulateur redondant. Cependant le choix du vecteur des vitesses peut être compliqué et peut être paramétré d'une façon plus générale. C'est pourquoi nous choisissons la méthode de l'inverse généralisé comme convenue dans la partie 1.1 de ce document.

2.1.1 Inverse généralisé

L'inverse de la relation des vitesses pour un robot non redondant est donné par la relation suivante :

$$\dot{q} = {}^{0}J_{7}^{-1}\dot{x} \tag{2.1}$$

Avec : ${}^{0}J_{7}$ représentant la matrice jacobienne et $\dot{x}$ le vecteur des vitesses.

Malheureusement pour nous, cette expression ne peut s'appliquer pour un robot redondant dont le nombre de degrés de liberté est supérieur au nombre de degré de liberté de la tâche à effectuer, étant donné que la matrice jacobienne est rectangulaire et par conséquent non inversible. La méthode de l'inverse généralisé permet de parcourir

tout l'espace des solutions de la cinématique différentielle inverse pour un robot redondant. Dans notre cas, c'est beaucoup plus intéressant compte tenu que pour éviter les obstacles, nous désirons offrir toutes les possibilités de le faire au robot et ainsi lui laisser le choix de la trajectoire optimale, Bigras [12].

L'inverse de la relation des vitesses est alors donné par :

$$\dot{q} = {}^0J_7^+\dot{x} - (I - {}^0J_7^+ {}^0J_7)\xi \tag{2.2}$$

Où :

- ${}^0J_7^+ = {}^0J_7^T({}^0J_7 {}^0J_7^T)^{-1}$ représente l'expression de l'inverse généralisé (2.3)
- $N = (I - {}^0J_7^+ {}^0J_7)$ représente l'espace nul de ${}^0J_7$.
- $\dot{x}$ représente la vitesse.
- Et ξ représente le vecteur responsable du mouvement interne du robot.

Étant donné que le vecteur ξ est multiplié par l'espace nul de la matrice jacobienne, il n'influe pas sur le déplacement de l'effecteur. Il ne fait qu'exploiter la redondance du robot pour réaliser une tâche auxiliaire : l'évitement d'obstacles. De plus, si nous établissons un ordre de priorité, l'équation 2.2 obligera d'abord l'effecteur du robot à suivre la vitesse imposée par le vecteur $\dot{x}$, c'est-à-dire que si le vecteur ξ est conçu de telle manière à éviter des obstacles et que la trajectoire désirée ne le permet pas physiquement, le robot ne décrochera pas de la trajectoire et rentrera dans l'obstacle.

2.1.2 Détermination du vecteur responsable du mouvement interne du robot redondant ANAT

Il nous reste maintenant à réécrire l'équation 2.2 afin que le robot évite les obstacles, Il nous faut donc déterminer l'expression correcte du vecteur ξ. Zlajpah [1] l'a réécrit sous la forme suivante :

$$\dot{q} = {}^0J_7^+\dot{x}_c + (I - {}^0J_7^+ {}^0J_7)\nabla V(q) \tag{2.4}$$

Où :

- $\dot{x}_c = \dot{x}_d + K_p e = \dot{x}_d + K_p(x_d - x)$ avec x_d et $\dot{x}_d$ représentant respectivement la position et la vitesse désirée et K_p une matrice de gain constante.
- $\nabla V(q)$ représente le vecteur ξ. C'est le gradient de la fonction objective $V(q)$ grâce à laquelle le robot réalise l'évitement d'obstacle.

Zlajpah [1] réécrit aussi le vecteur des vitesses en introduisant un correcteur proportionnel intégral, l'erreur s'en trouve réduite et nous pouvons adapter le temps de réponse, le dépassement en fonction de la tâche à réaliser.

Essayons d'expliciter l'expression (2.4) en détaillant un déplacement Δx, Wang [2]. Considérons que l'effecteur du robot est positionné en $x = f(q)$ et supposons que l'on impose une nouvelle position désirée $x_d = x + \Delta x = f(q + \Delta q)$.

Afin de bien identifier le rôle des deux différents termes de l'équation 2.4, détaillons le processus en deux étapes.

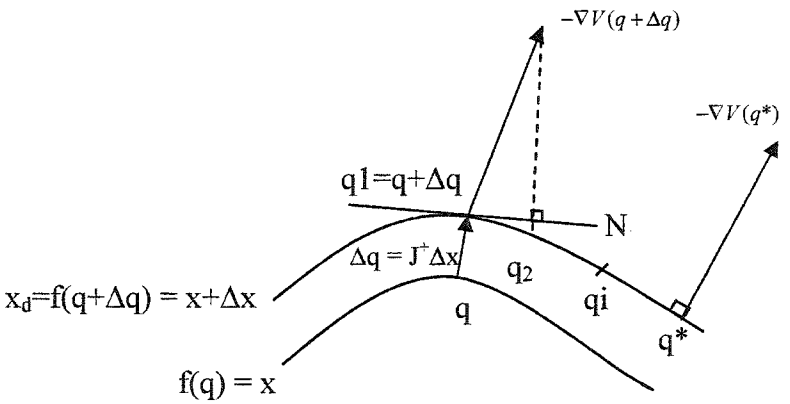


Figure 4 Projection du gradient

Pour commencer, nous allons regarder le premier terme représentant une simple pseudo inverse pour effectuer le déplacement de l'effecteur de fin : $\Delta q = J^+ \Delta x$ .Nous basculons ainsi dans l'espace des solutions $f(q + \Delta q) = x + \Delta x$.

Le deuxième terme impose une descente de gradient dans l'espace des solutions x_d afin de minimiser l'énergie potentielle : $N\nabla V(q+\Delta q)$. Cette opération se répète jusqu'à ce que l'on aboutisse à un q* optimal qui apparaît lorsque le gradient de l'énergie potentielle est normal à l'espace nul. C'est une minimisation de la fonction objective $V(q)$, voir figure 4.

Enfin, si on diminue suffisamment le pas de temps Δt et que l'on ajoute les deux termes précédents, nous retrouvons bien l'équation 2.4 :

$$\dot{q} = {}^0J_7^+ \dot{x}_c - (I - {}^0J_7^+ {}^0J_7)\nabla V(q)$$

Suivant l'expression de la fonction $V(q)$, nous pouvons minimiser la vitesse des articulations du système, minimiser la consommation d'énergie ou encore éviter des obstacles. Nous nous sommes fixés comme objectif l'évitement d'obstacle et c'est ce à quoi nous servira cette fonction de coût.

2.1.3 Modélisation de l'obstacle

Pour déterminer la fonction objective permettant d'éviter les obstacles, il faut d'abord commencer par modéliser ceux-ci. Nous travaillons dans l'espace et une modélisation d'objets en trois dimensions, dans l'espace cartésien, peut s'avérer parfois compliquée. De plus, le calcul de distances minimales entre les différentes parties du robot et les obstacles peuvent alourdir considérablement les calculs. Ainsi, nous avons choisi une modélisation par des fonctions de surfaces super quadratiques, méthode utilisée par Perdereau [6] et Khosla [11].

Cette méthode nous permet de décrire la surface d'un objet ou de l'envelopper dans une hyper surface dont l'expression analytique est connue. De ce fait, le problème de fonctions de surface est résolu et nous pouvons approximer d'une manière très simple un grand nombre d'obstacles.

Le principe de ces expressions est le suivant, en utilisant des formules d'ellipses et d'ellipsoïdes et en modifiant les puissances présentes dans ces formules, nous pouvons approximer des formes comme des cylindres, des trapèzes ou encore des cônes. La formule générale est la suivante :

$$S(x,y,z)=\left(\frac{x}{f_1(x,y,z)}\right)^{2n}+\left(\frac{y}{f_2(x,y,z)}\right)^{2n}+\left(\frac{z}{f_3(x,y,z)}\right)^{2p}-1 \qquad (2.5)$$

Où :

- (x,y,z) représente les coordonnées du point à partir duquel on désire calculer la pseudo distance de l'ellipsoïde
- f_1, f_2, f_3 sont des fonctions qui permettent de donner la forme désirée à l'ellipsoïde (cylindre, cône…).
- n et p permettent de rapprocher l'enveloppe de l'ellipsoïde de la forme désirée. Plus l'enveloppe est proche, plus l'obstacle est décrit précisément, mais le temps de calcul se trouve augmenté.

À titre d'exemple, certaines figures sont présentes en annexe 3 ainsi que leurs équations associées.

Analysons maintenant les différentes valeurs que peut prendre la fonction $S(x,y,z)$ et voyons quel rôle elle apporte dans l'évaluation de pseudo distances entre le robot et l'obstacle.

En fait pour comprendre cette idée, imaginons un solide dont le volume est décrit par l'équation de surface $S(X) = 0$.

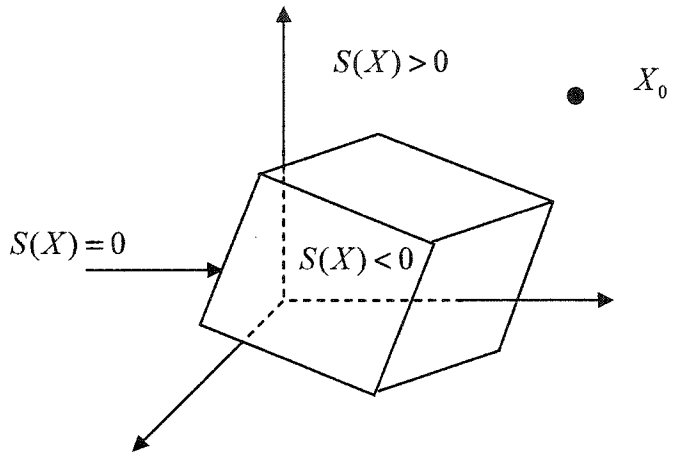


Figure 5 Illustration d'une fonction de surface

L'espace des solutions est décrit telle que suit :

- $S(X_0) < 0$ si X_0 est à l'intérieur du solide.
- $S(X_0) = 0$ si X_0 est sur la surface du solide.
- $S(X_0) > 0$ si X_0 est à l'extérieur du solide.

On remarque que $S(X_0)$ exprime une pseudo distance entre le point X_0 et la surface de l'objet. Bien sûr, cette distance n'est pas une distance euclidienne mais elle permet tout de même de faire croître ou décroître la fonction objective en fonction de l'éloignement du point X_0 de la surface du solide.

Enfin, il nous a paru utile de pouvoir donner une marge de sécurité d_s entre le robot et l'obstacle. Ceci se résume à écrire l'inéquation suivante :

$$h(q) = -S(X_0) + d_s^2 \leq 0 \qquad (2.6)$$

La distance de sécurité d_s étant elle-même une pseudo distance, est difficile à représenter dans l'espace cartésien. Nous verrons, dans la partie 2.3, comment s'affranchir de ce problème.

Il nous reste maintenant à choisir la localisation des points X_0 sur le robot afin d'éviter toute collision avec un obstacle. Il est clair que plus nous prenons de points plus les calculs sont alourdis et ainsi le temps de calcul augmenté. Il nous a donc paru judicieux, suivant la structure du robot ANAT et pour ne pas alourdir les calculs de prendre comme X_0 les origines des repères des différents joints ainsi que les points médians entre ces origines.

2.1.4 Détermination de la fonction objective

Comme nous l'avons vu dans la partie 2.1.2, la fonction objective doit décroître si les contraintes sont respectées. Nous allons tout d'abord présenter les types de contraintes associées à notre problème.

Pour réaliser l'évitement d'obstacle, n points seront placés sur le robot et l'évitement sera calculé à partir de ceux-ci. L'espace sera composé quant à lui de m obstacles. Nous avons donc déjà $n \times m$ contraintes associées à l'évitement d'obstacle. A ces contraintes, nous ajoutons celles dues aux limitations physiques du robot, par exemple empêcher le robot de se heurter à lui-même, ou encore qu'il aille au delà de son espace de travail. Ceci est détaillé, pour le robot ANAT, dans la partie 2.3.2. Voici la forme générale de la fonction objective.

$$V(q) = \sum_{i=1}^{l} \alpha_i p(h_i(q)) \tag{2.7}$$

Où :

- l représente le nombre de contraintes.
- les α_i représentent les poids assignés à ces contraintes suivant l'importance que l'on désire leur donner
- $p(h(q))$ représente une fonction de pénalité, que nous présenterons ci-après, et qui tend vers zéro lorsque la contrainte $h(q)$ est satisfaite.
- $h(q)$ représente la fonction associée aux différentes contraintes, pour l'évitement d'obstacle, c'est la fonction 2.6.

Étant donné les caractéristiques de la fonction $h(q)$, il nous faut une fonction de pénalité qui respecte les conditions suivantes :

$$p(h) > 0 \text{ si } h(q) < 0 \quad \text{et} \quad p(h) \to \infty \text{ si } h(q) \to 0$$

Nous avons donc choisi la fonction $p(h) = -\frac{1}{h(q)}$ illustrée par la figure 6.

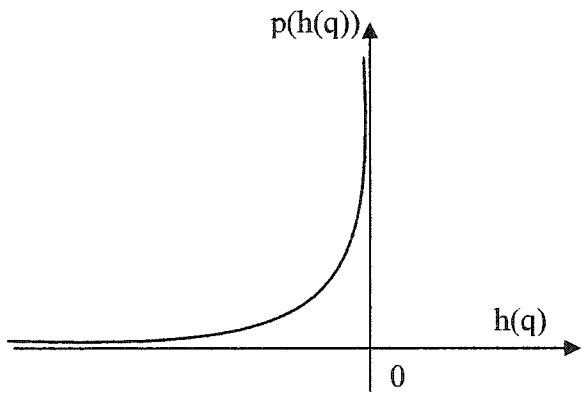


Figure 6 Fonction de pénalité

Nous avons ainsi toutes les données nécessaires pour appliquer l'équation 2.2 et réaliser la cinématique différentielle inverse tout en permettant au robot d'éviter les obstacles. Il nous reste simplement à générer la trajectoire dans l'espace cartésien.

2.2 Génération de trajectoire dans l'espace cartésien

Pour générer la trajectoire du robot dans l'espace de la tâche, nous utilisons un polynôme d'ordre cinq afin d'obtenir une trajectoire douce. Les polynômes d'ordre cinq prennent la forme suivante :

$$p_{xi}(t) = a_{xi5}t^5 + a_{xi4}t^4 + a_{xi3}t^3 + a_{xi2}t^2 + a_{xi1}t + a_{xi0}$$

Où :

$$a_{xi5} = \frac{12(x_{i+1} - x_i)}{2t_1^5} \quad , \quad a_{xi4} = \frac{30(x_i - x_{i+1})}{2t_1^4} \quad , \quad a_{xi3} = \frac{20(x_{i+1} - x_i)}{2t_1^3}$$

$$a_{xi2} = a_{xi1} = 0 \quad , \quad a_{xi0} = x_i$$

Et avec :

- x_i la coordonnée de départ et x_{i+1} la coordonnée d'arrivée.
- t_1 le temps nécessaire pour aller de la coordonnée x_i à celle de x_{i+1}.

La figure 7 présente les résultats pour un segment de droite parcouru entre les points de coordonnées (x_1,y_1) et (x_2,y_2) sur une période de temps t_1 :

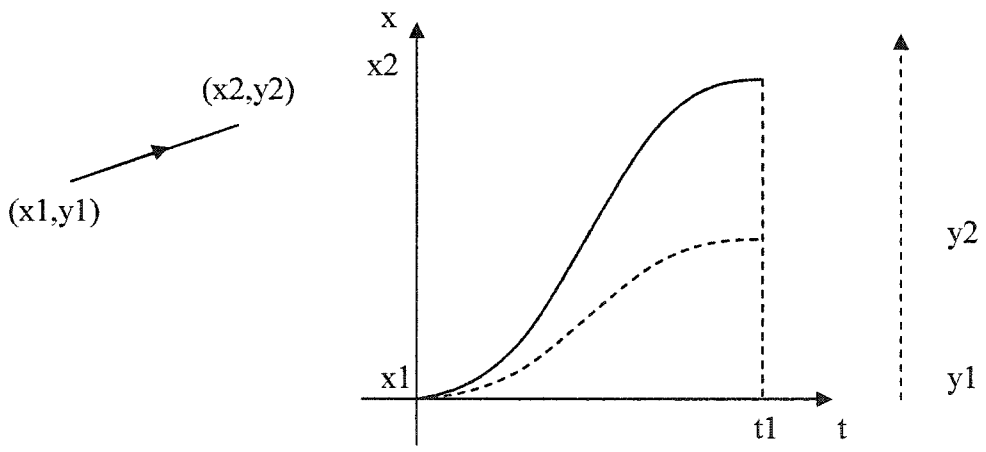


Figure 7 Génération de trajectoire à l'aide d'un polynôme d'ordre cinq

2.3 Programmation de l'algorithme en utilisant le logiciel SIMULINK de MATLAB

Nous allons maintenant utiliser les informations développées précédemment pour réaliser une simulation du problème. Le logiciel utilisé est SIMULINK, mais certaines fonctions sont générées à l'aide de MAPLE qui permet le calcul littéral. Une bibliothèque de robotique, spécialement conçue pour ce logiciel par Bigras [12], facilite les calculs et le temps associé à celui-ci. L'équation 2.2 nous permet de réaliser le schéma bloc de la figure 8.

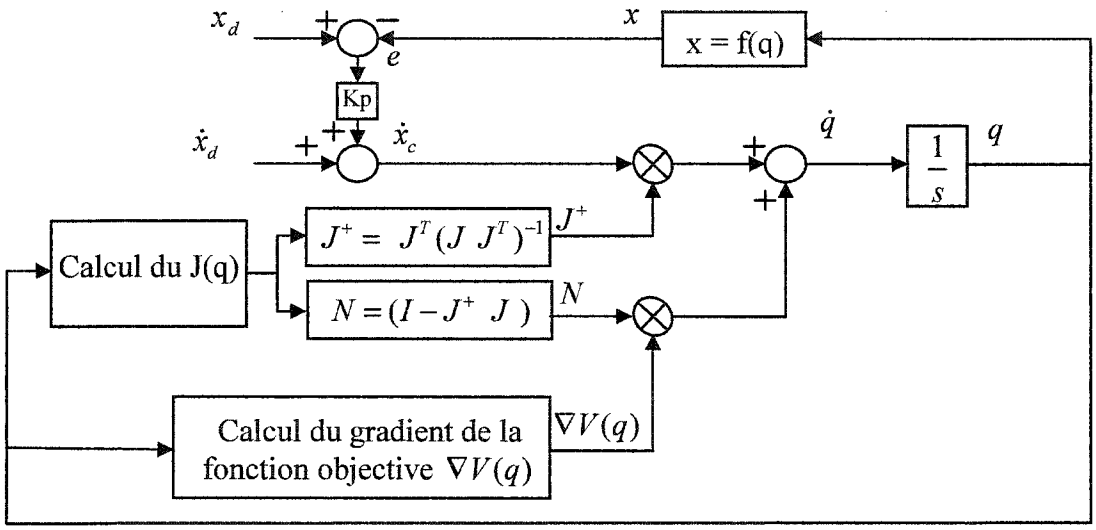


Figure 8 Schéma bloc de la cinématique différentielle inverse

Nous allons détailler ce schéma et notamment les trois fonctions importantes calculées avec le logiciel maple : le jacobien, la fonction objective et enfin la fonction f(q).

2.3.1 Calcul du jacobien et de la fonction x=f(q)

Nous utilisons une librairie pour la modélisation cinématique de robot créé par Bigras [12] pour le logiciel maple. Pour le calcul des jacobiens linéaire et angulaire, il nous suffit de rentrer le tableau cinématique de Craig ainsi que les coordonnées généralisées.

Nous appelons alors les fonctions prévues pour le calcul des deux jacobiens et nous concaténons les deux matrices pour obtenir la matrice Jacobienne. Une ligne de commande permet ensuite de générer cette matrice pour qu'elle soit utilisable sous Simulink, en fait elle génère un fichier *ssm.c que l'on compile ensuite. Une liste de paramètres est associée à ce fichier, ceux-ci seront initialisés au début de la simulation.

Pour ce qui est de la fonction f(q), le principe d'utilisation de maple reste le même. La bibliothèque nous permet de réaliser la cinématique inverse du robot en suivant l'équation 2.5, celle-ci est ensuite transformée pour être utilisable sous matlab.

2.3.2 Calcul de la fonction objective

Avant de calculer la fonction objective, il nous faut bien sûr fixer le nombre d'obstacles, leurs types ainsi que leurs emplacements dans l'espace de travail. Mais ce n'est pas tout, car il faut aussi tenir compte des spécificités physiques du robot contenues dans le tableau 2.1. Le robot ne doit pas par exemple se rétracter sur lui-même! Enfin, à cause de sa physionomie et de sa redondance, l'effecteur peut percuter la partie prismatique du robot. La fonction objective doit donc tenir compte de tous ces paramètres et ne pas simplement réaliser l'évitement d'obstacle. C'est pourquoi nous avons décidé de diviser la fonction objective en trois parties.

- f_{obj1} représente la partie de la fonction objective responsable de l'évitement d'obstacle.
- f_{obj2} représente la partie de la fonction objective imposant les limitations physiques du robot
- f_{obj3} représente la partie de la fonction objective évitant au robot de heurter sa partie prismatique.

Ces trois fonctions sont inspirées de Perdereau [6], mais ont été modifiées pour s'appliquer à notre problème. Nous sommons ces trois fonctions pour obtenir la fonction objective globale.

2.3.2.1 Fonction objective responsable de l'évitement d'obstacle

Nous nous sommes fixés un environnement le plus révélateur possible afin de tester l'algorithme de commande utilisé sur le robot ANAT. Nous proposons un environnement composé de trois obstacles maximum : des cylindres de hauteur variable et d'un diamètre de 35 mm. Voici les équations représentant l'évitement d'obstacle et qui seront présentes dans la fonction objective :

$$f_{obj1} = \sum_{i=1}^{n}\sum_{j=2}^{m} \frac{\alpha_{ji}}{-\left(\frac{x_j - x_{ci}}{r_i + r_{si}}\right)^2 - \left(\frac{y_j - y_{ci}}{r_i + r_{si}}\right)^2 - \left(\frac{z_j}{h_i + h_{si}}\right)^2 + 1} \tag{2.8}$$

Où:

- n représente les différents obstacles.
- m représente les points que nous avons placés sur le robot et à partir desquels l'algorithme réalise l'évitement d'obstacle, ici ce sont les origines des repères 2 à 7 associées aux joints ainsi que les points médians entre ces origines.
- Les α représentent les poids pour donner plus ou moins d'importance à une contrainte. Par exemple, si l'on désire éloigner l'articulation 3 de l'obstacle 2 on augmente le α_{23}.
- (x_j, y_j, z_j) représentent les coordonnées de l'articulation j dans le repère de la base.
- $(x_{ci}, y_{ci}, r_i, h_i)$ représentent les coordonnées du cylindre j dans le repère de base.
- r_{si}, h_{si} représentent respectivement le rayon et la hauteur de sécurité entre les articulations et le cylindre j. C'est en fait la distance de sécurité d_s présenté dans

l'équation 2.6. Toutefois cette distance de sécurité n'est plus une pseudo distance mais une distance cartésienne.

2.3.2.2 Fonction objective imposant les limitations physiques du robot

Cette fonction objective permet que le robot évite de dépasser ses limites physiques. En fait, le joint prismatique doit varier entre L_0 = 57 cm et L_0+70 cm, les autres joints doivent eux varier entre -90°et +90°sauf le joint 6 entre 0° et 180° et le joint 7 qui n'a pas de limite.

Détaillons maintenant la partie de la fonction objective évitant que le robot se rétracte sur lui-même :

$$f_{obj2} = \sum_{i=1}^{2} \sum_{j=1}^{7} \frac{\lambda_{ji}}{(-1)^i \left(q_j - l_{ji}\right)} \tag{2.9}$$

Où :

- Les λ_{ji} sont des poids donnant plus ou moins d'importance à certaines contraintes, le premier indice correspond au numéro du joint et le deuxième à la contrainte physique minimale (1) où maximale (2) associée.
- Les l_{ji} représentent les contrainte physique minimale (1) où maximale (2) associée au joint j.
- Les fonctions composant le dénominateur de toutes les fractions possèdent les mêmes propriétés que la fonction h(q) présentée en 2.1.4, elles sont négatives et tendent vers zéro lorsque le robot s'approche de la limite physique à respecter.

2.3.2.3 Fonction objective empêchant le robot de heurter sa partie prismatique

Il nous reste maintenant à détailler la dernière partie de la fonction objective, celle qui permet d'éviter au robot de percuter sa base. Nous considérons que la base du robot, en fait tout le bloc prismatique, est un obstacle. Il nous suffit donc de décomposer ce bloc

en deux parallélépipèdes. Le premier est centré en $x_{p1} = 0.0835$ et $y_{p1} = 0$ avec une hauteur $h_{p1} = L_0 + 0.7$, une longueur suivant x de $L_{xp1} = 0.0835$ m et une largeur de $L_{yp1} = 0.05$ m. Le deuxième est centré en $x_{p2} = 0.08$ et $y_{p2} = 0$ avec une hauteur $h_{p2} = L_0 + 0.7$, une longueur suivant x de $L_{xp2} = 0.08$ m et une largeur de $L_{yp2} = 0.1$ m.

$$f_{obj3} = \sum_{i=1}^{2}\sum_{j=2}^{m} \frac{\beta_{ji}}{-\left(\frac{x_j - x_{pi}}{L_{xpi} + x_{si}}\right)^2 - \left(\frac{y_j - y_{pi}}{L_{ypi} + y_{si}}\right)^2 - \left(\frac{z_j}{h_{pi} + h_{si}}\right)^2 + 1} \tag{2.10}$$

Là aussi m représente le nombre de points placés sur le robot à partir desquelles on réalise l'évitement d'obstacle, (x_{si}, y_{si}, h_{si}) sont les distance de sécurité suivant les différents cotés et enfin les β_{ji} les poids associés aux contraintes.

La fonction objective totale a donc pour valeur :

$$V(q) = f_{obj1} + f_{obj2} + f_{obj3} \tag{2.11}$$

2.4 Résultats de simulation

Pour cette simulation nous plaçons deux cylindres d'un mètre de hauteur et de 35 millimètres de diamètre aux coordonnées [0.1,-0.2] et [0.3,-0.2]. Nous demandons ensuite au robot de répéter une trajectoire rectangulaire qui suit dans l'ordre les quatre coins suivant : [0.6 ; -0.5 ; 0.1+L0], [0.6 ; -0.6 ; 0.5+L0], [-0.2 ; -0.6; 0.5+L0], [0.2 ; -0.5 ; 0.1+L0]. L'intérêt de cet environnement de travail est que l'on voit le robot travailler aux limites physiques de son espace de travail. En effet l'articulation numéro deux va pratiquement toujours travailler à +90°. De plus, le robot va devoir s'étirer de tout son long pour pouvoir accéder aux quatre coins de la trajectoire désirée sans toucher les obstacles.

Les coefficients de la fonction objective sont tous à 0.01 sauf ceux pour l'évitement d'obstacle des membres trois, quatre et cinq pour les cylindres 1 et 2 qui sont à 150. En effet, compte tenu de la tâche à réaliser, ils ont beaucoup plus de risque de heurter les cylindres. Il faut donc augmenter considérablement ces coefficients afin que l'évitement d'obstacle soit réalisé correctement.

2.4.1 Visualisation des résultats grâce au logiciel d'animation 3D

Un programme, que nous avons développé en open GL, nous permet de visualiser le robot ainsi que les différents obstacles et la trajectoire désirée. Voici un exemple (figure 9) pour l'application précédente.

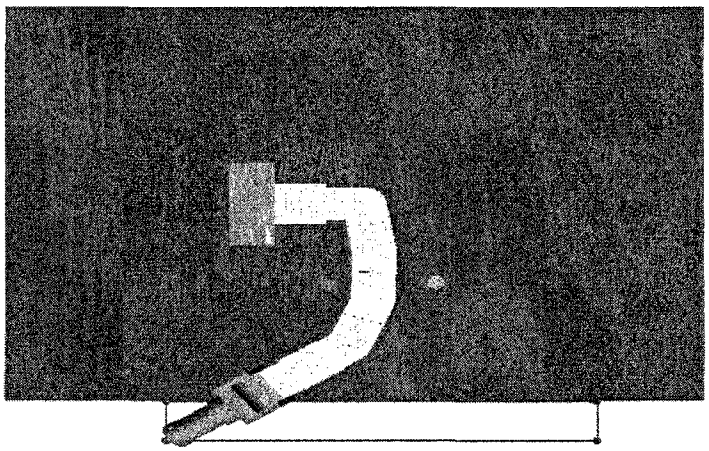


Figure 9 Résultat de la cinématique inverse avec évitement (vue de dessus)

Cette vue de dessus nous permet de voir que le robot s'étire (notamment le joint 6) afin de ne pas percuter le cylindre de gauche.

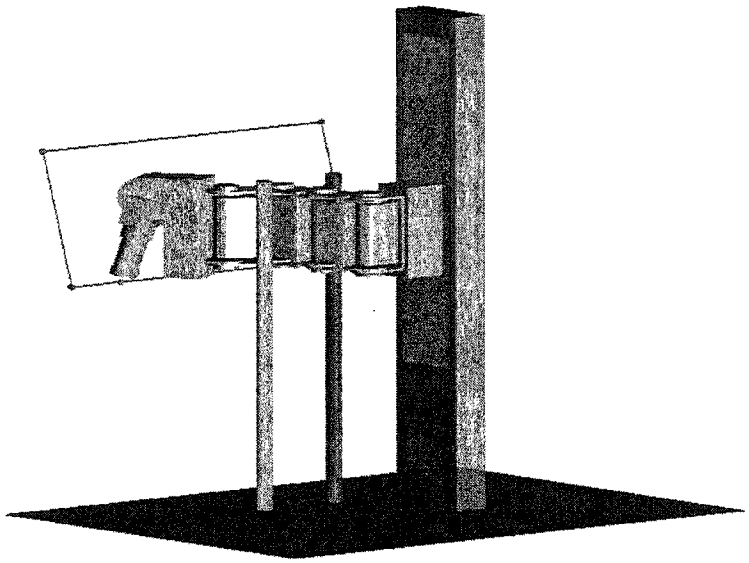


Figure 10 Résultat de la cinématique inverse avec évitement (vue de côté)

Nous remarquons, grâce à ce logiciel que la cinématique ainsi que l'évitement d'obstacle est correctement réalisé. Cet outil est très appréciable car il est difficile de voir si le robot réalise correctement l'évitement d'obstacle en étudiant simplement les positions de tous les joints.

2.4.2 Étude des courbes représentant les trajectoires désirées appliquées au contrôleur adaptatif et l'erreur de position

Les résultats de cette simulation sont les graphiques de la trajectoire désirée avec évitement d'obstacle et dans l'espace des articulations que nous allons appliquer à l'entrée du contrôleur adaptatif. Ils sont présentés dans l'annexe 4. Nous en avons choisi quelques uns pour analyser les résultats.

Regardons tout d'abord l'articulation 2, sur la figure 11, qui est une des plus intéressante pour cette simulation.

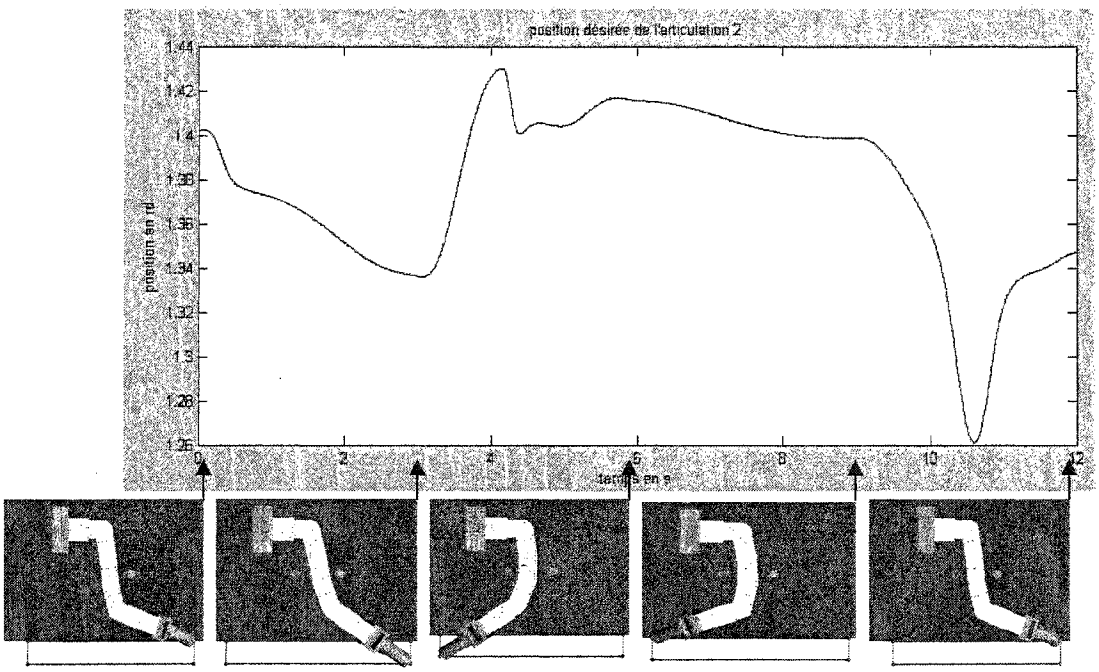


Figure 11 Position désirée de l'articulation 2

On remarque entre les temps 3 et 6 secondes que l'effecteur du robot est à la gauche du rectangle, c'est là où q_2 pourrait valoir plus de $\pi/2$, il est arrêté par la contrainte physique associée à cette articulation dans la fonction objective et aussi à cause du cylindre de gauche que pourrait rencontrer le robot. Nous remarquons, entre les temps 9 et 12 secondes, que le robot s'éloigne de $\pi/2$ rapidement mais est à nouveau arrêté par le cylindre de droite.

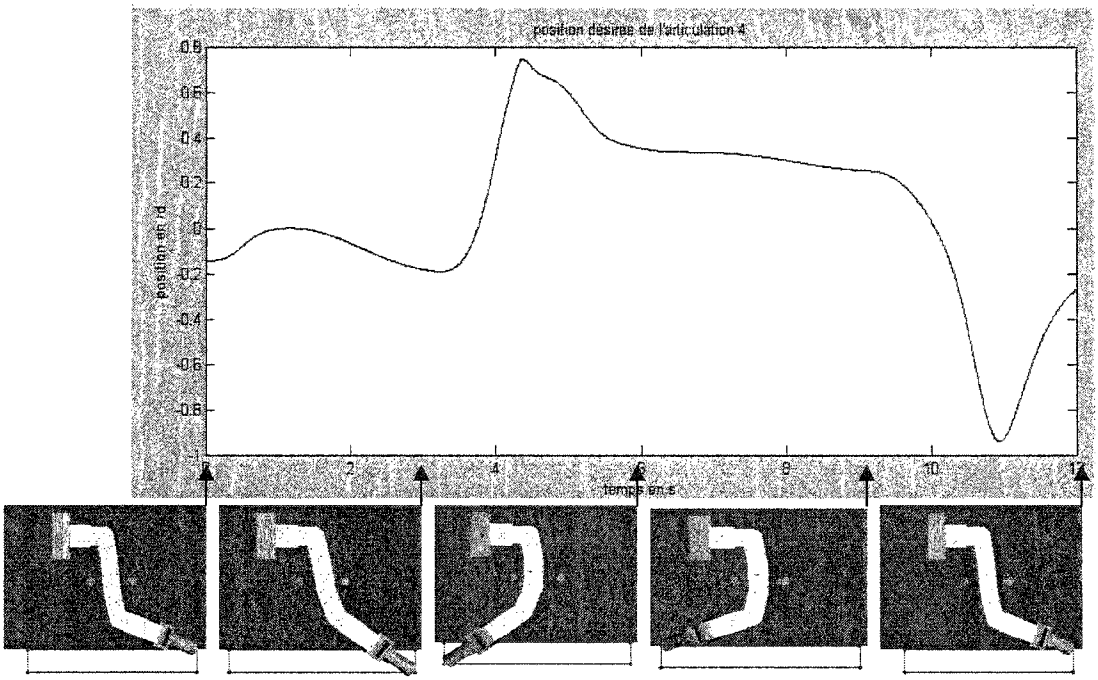


Figure 12 Position désirée de l'articulation 4

L'articulation 4 est celle qui a le plus de chance de heurter les cylindres, on remarque là aussi entre les temps 3 et 6 que q_4 part pour aller plus que 0.8 radians, elle est arrêtée par la fonction objective pour qu'elle ne heurte pas le cylindre de gauche. Il en est de même entre les temps 9 et 12, q_4 part pour aller plus que -1 radian mais là aussi, elle est arrêtée par la fonction objective pour qu'elle ne heurte pas le cylindre de droite.

Regardons maintenant l'erreur de position. L'erreur de position amenée par l'algorithme d'évitement d'obstacle est négligeable. En effet, celle-ci varie entre -1 et 1 millimètre pour l'erreur en x et y et est pratiquement négligeable pour l'erreur en z. Nous pouvons le voir sur la figure 13.

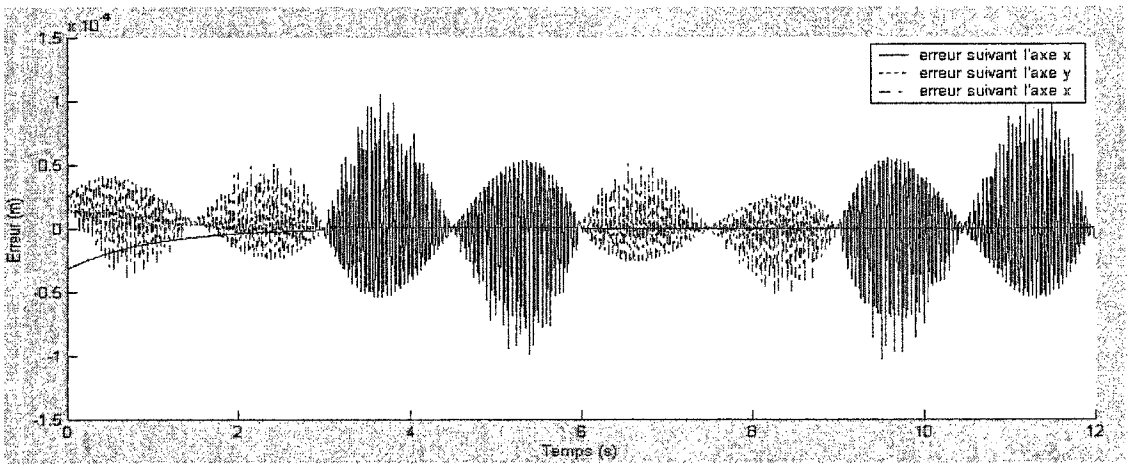


Figure 13 Erreur entre la trajectoire désirée avant et après l'évitement obstacle

Il nous reste maintenant à appliquer ces trajectoires désirées au contrôleur adaptatif, pour terminer la commande du robot. C'est ce que nous allons voir dans le chapitre trois.

CHAPITRE 3

COMMANDE ADAPTATIVE DU ROBOT ANAT

Tout modèle utile à des fins de contrôle de systèmes réels possède divers paramètres en rapport avec les caractéristiques physiques et mécaniques de celui-ci. Cependant certains de ces paramètres peuvent être incertains. La commande adaptative a été créée pour pouvoir commander un système sans avoir connaissance de tous ses paramètres dynamiques. Grâce à celle-ci, il est possible d'assurer la stabilité de la commande de positionnement d'un robot tout en sachant que les paramètres suivants sont inconnus:

- la masse des différents membres du robot;
- les inerties des membres;
- la position des centres de masse.

Nous allons présenter dans ce chapitre le type de commande adaptative utilisée, prouver la stabilité de cette commande et enfin présenter les résultats de simulation.

3.1 Modélisation et paramètres dynamiques

Nous avons réalisé dans la partie 2.3 la modélisation du robot ANAT par la méthode de Lagrange. Nous obtenons un modèle sous la forme suivante :

$$M(q)\ddot{q} + F(q,\dot{q}) = \tau$$

Une des caractéristiques des robots manipulateurs, qui permet la synthèse des contrôleurs adaptatifs, est que leurs modèles dynamiques peuvent s'exprimer en termes linéaires d'un ensemble de paramètres correctement choisis appelés paramètres dynamiques. Le modèle peut donc se réécrire sous la forme :

$$Y_r(q,\dot{q},\ddot{q})a + F_f(\dot{q}) = \tau \qquad (3.1)$$

Où :

- a représente le vecteur des paramètres dynamiques;
- $F_f(\dot{q})$ représente le vecteur de frottement visqueux.

Dans le cas de notre robot ANAT, le vecteur des paramètres dynamiques prend la forme suivante :

$$a = [m_1 \quad m_2 \quad m_3 \quad m_4 \quad m_5 \quad m_6 \quad m_7 \quad m_5 L_3 \quad -m_6 L_4 \quad -m_7 L_5 \quad I_{y6} \quad m_7 L_5^{\,2} + I_{y7} \quad m_4 L^2 + I_{z4} \quad m_5 L_3^{\,2} + I_{z5} \quad m_6 L_4^{\,2} + I_{z6} \quad I_{z7}]^T \tag{3.2}$$

Nous déterminons ce vecteur en utilisant les données du Tableau III des spécificités dynamiques du robot ANAT. Les m_i sont identifiées dans un premier temps puis les termes $m_i x_{cj}$, $m_i y_{cj}$ et $m_i z_{cj}$ et enfin les termes $m_i(y_{ci}^2+z_{ci}^2)+I_{xx}$, $m_i(x_{ci}^2+z_{ci}^2)+I_{yy}$ et $m_i(x_{ci}^2+y_{ci}^2)+I_{zz}$. Une fois ce vecteur calculé, on détermine par identification le produit $Y_r(q,\dot{q},\ddot{q})a$, on le linéarise par rapport au vecteur a et enfin on détermine $Y_r(q,\dot{q},\ddot{q})$.

3.2 Loi de commande et d'adaptation

Maintenant que nous avons le modèle dynamique du robot, linéaire par rapport au vecteur des paramètres dynamiques, et, en considérant un ensemble de fonctions vectorielles bornées $q_d, \dot{q}_d$ et $\ddot{q}_d$ rapportées par l'algorithme d'évitement d'obstacle comme positions, vitesses et accélérations articulaires. Il s'agit de faire la synthèse des contrôles pour satisfaire l'objectif de contrôle du mouvement établi comme :

$$\lim_{t \to \infty} \tilde{q}(t) = 0$$

Où $\tilde{q}$ représente l'erreur de position articulaire définie comme $\tilde{q} = q_d - q$

Un contrôleur adaptatif est généralement composé de deux parties :

- la loi de commande ou contrôleur,
- la loi d'adaptation.

La loi de commande est, en général, une équation algébrique qui permet de calculer une action de contrôle s'exprimant de la manière suivante :

$$\tau = \tau(t, q, \dot{q}, q_d, \dot{q}_d, \ddot{q}_d, \hat{a})$$

Avec $\hat{a}$ représentant les paramètres adaptatifs.

La loi de commande précédente permet, lors du remplacement des paramètres adaptatifs par les paramètres dynamiques (le contrôleur n'est plus adaptatif), au système de vérifier le contrôle du mouvement.

La loi d'adaptation permet elle de déterminer le vecteur de paramètres dynamiques $\hat{a}(t)$. Elle est souvent exprimée comme une équation différentielle en $\hat{a}$.Une des plus communément employées est appelée loi intégrale ou du type gradient :

$$\hat{a}(t) = \Lambda \int_0^t \psi(t, q, \dot{q}, q_d, \dot{q}_d, \ddot{q}_d) dt$$

Λ ou $\Lambda^T \in R^{m \times n}$ est normalement diagonale et définie positive et représente le gain d'adaptation. L'ampleur de ce gain est en rapport avec la vitesse d'adaptation du système. Généralement ce gain prend une petite valeur. Observons la loi de commande sur la figure 14.

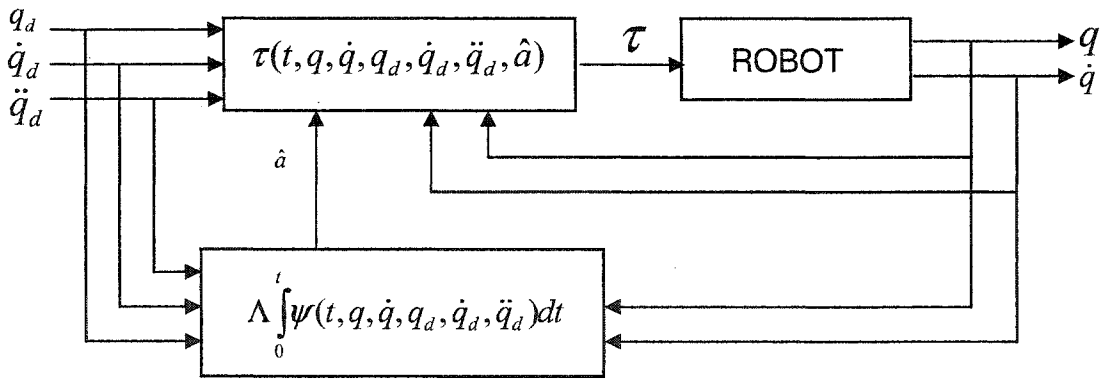


Figure 14 Commande adaptative de robot

Cette figure 14 nous montre les interactions entre la loi de commande et d'adaptation ainsi que les paramètres utilisés pour chacune d'elles.

3.3 Loi de commande et d'adaptation utilisées et preuve de stabilité

Nous avons vu dans le chapitre 3 portant sur la modélisation dynamique du robot que le modèle d'un robot non contraint peut s'écrire sous la forme suivante :

$$M(q)\ddot{q} + Vm(q,\dot{q})\dot{q} + F_f(\dot{q}) + F_g(q) = \tau \tag{3.3}$$

La loi de commande que nous utilisons est une commande de position basée sur la théorie de Lyapunov et semblable à un couple pré calculé.

$$\tau = M(q)\ddot{q}_r + V_m(q,\dot{q})\dot{q}_r + F_f(\dot{q}) + F_g(q) + K_d s = Y_r(q,\dot{q},\dot{q}_r,\ddot{q}_r)a + F_f(\dot{q}) + K_d s \tag{3.4}$$

Où :

$$\dot{q}_r = \dot{q}_d + K_p(q_d - q) = \dot{q}_d + K_p\tilde{q}$$
$$\ddot{q}_r = \ddot{q}_d + K_p(\dot{q}_d - \dot{q}) = \ddot{q}_d + K_p\dot{\tilde{q}}$$
$$s = \dot{\tilde{q}} + K_p\tilde{q} = \dot{q}_r - \dot{q}$$

Avec K_p et K_d deux matrices de gains symétriques définies positives.

Dans le cas d'une commande adaptative, nous pouvons réécrire l'équation 3.4 sous la forme suivante :

$$\tau = Y_r(q,\dot{q},\dot{q}_r,\ddot{q}_r)\hat{a} + F_f(\dot{q}) + K_d s \tag{3.5}$$

Avec $\hat{a}$ représentant l'estimation du vecteur des paramètres dynamiques inconnus a.

Nous considérons maintenant la loi d'adaptation suivante :

$$\dot{\hat{a}} = \Lambda Y_r(q,\dot{q},\dot{q}_r,\ddot{q}_r)s \tag{3.6}$$

Maintenant que nous avons choisi les lois de commande et d'adaptation, nous allons prouver la stabilité du système en étudiant la dynamique de s et de l'erreur d'estimation. Nous introduisons l'erreur d'estimation ($\tilde{a} = a - \hat{a}$) grâce aux équations 3.4 et 3.5 :

$$\begin{aligned}\tau &= Y_r(q,\dot{q},\dot{q}_r,\ddot{q}_r)a - Y_r(q,\dot{q},\dot{q}_r,\ddot{q}_r)\tilde{a} + F_f(\dot{q}) + K_d s \\ &= M(q)\ddot{q}_r + V_m(q,\dot{q})\dot{q}_r + F_f(\dot{q}) + F_g(q) + K_d s - Y_r(q,\dot{q},\dot{q}_r,\ddot{q}_r)\tilde{a}\end{aligned} \tag{3.7}$$

En appliquant la loi de commande 3.7 à la modélisation dynamique 3.3, nous obtenons la dynamique de s suivante :

$$M(q)\dot{s} + V_m(q,\dot{q})s + K_d s = Y_r(q,\dot{q},\dot{q}_r,\ddot{q}_r)\tilde{a} \tag{3.8}$$

Afin de prouver la stabilité du système, nous choisissons la fonction de Lyapunov définie positive suivante:

$$v = \frac{1}{2}\left(s^T M(q)s + \tilde{a}^T \Lambda \tilde{a}\right)$$

Sa dérivée en fonction du temps nous donne l'expression suivante :

$$\dot{v} = -s^T K_d s$$

Cette fonction étant définie semi négative, d'après le Théorème de Lyapunov le système étudié est stable. De plus, étant donné que la trajectoire désirée est uniformément bornée et que nous venons de le voir, $\dot{s}$ l'est aussi, $\ddot{v} = -2s^T K_d \dot{s}$ est uniformément bornée. Le lemme de Barbalat permet d'affirmer que $\dot{v} \to 0$ quand $t \to \infty$. Ainsi les erreurs de suivi convergent à zéro.

3.4 Programmation de l'algorithme en utilisant le logiciel SIMULINK de MATLAB

Comme pour la programmation de la cinématique différentielle inverse, nous avons utilisé Simulink. Maple est, lui aussi, utilisé pour le calcul des matrices $M(q)$, $Y_r(q, \dot{q}, \dot{q}_r, \ddot{q}_r)$ et $F(q, \dot{q})$. Nous avons simplement à mettre en entrée de ces S fonctions les variables dont dépendent ces trois matrices et leurs valeurs sont actualisées. Le schéma bloc de la commande adaptative, basée sur Lyapunov, est présenté ci-après .

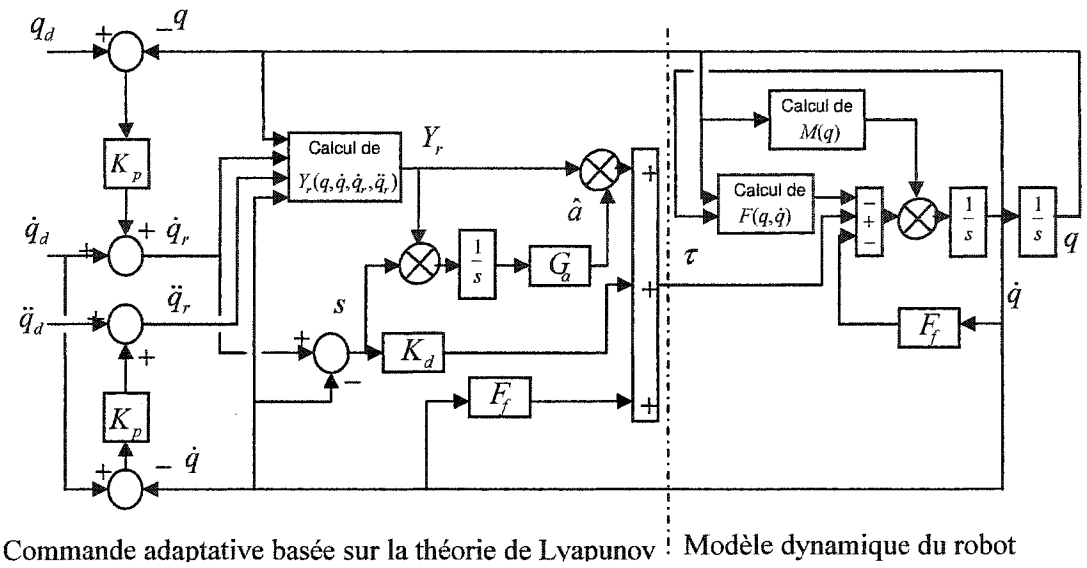


Figure 15 Contrôleur basé sur la théorie de Lyapunov

Si nous connectons les schémas bloc de la cinématique différentielle inverse avec évitement d'obstacles et le contrôleur adaptatif basé sur la théorie de Lyapunov, nous obtenons le schéma bloc présenté sur la figure 16.

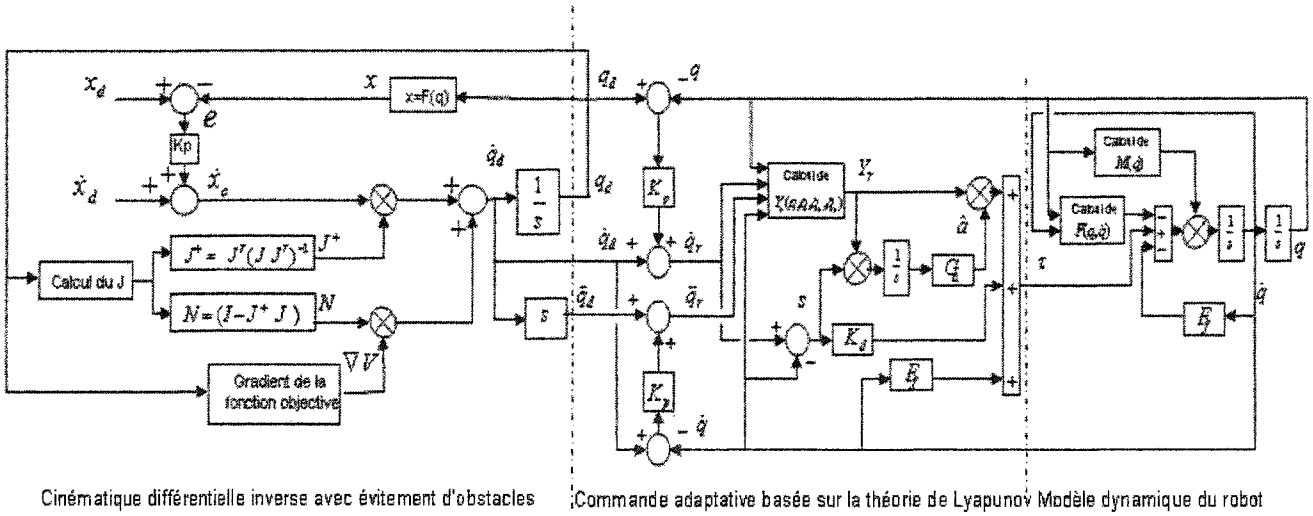


Figure 16 Contrôleur basé sur la théorie de Lyapunov avec évitement d'obstacles

3.5 Résultats de simulation

Pour la simulation, il nous a fallu déterminer les gains K_d, K_p et G_a. Il n'y a pas réellement de méthode pour obtenir les valeurs exactes de ces gains, et ce, compte tenu de la non linéarité du problème. Toutefois, nous pouvons déterminer une première valeur de K_d et K_p en les calculant de façon à imposer les pôles et par conséquent le temps de réponse et le dépassement. Nous avons procédé de cette manière pour ensuite lancer un protocole d'expérience et optimiser ces gains. Si nous désirons un temps de réponse d'environ $T_r = 0.5$ seconde, les gains ont pour valeur :

$$K_p = I * \frac{3}{T_r}, \qquad K_d = 12 * K_p, \qquad G_a = I * 5$$

Nous pouvons maintenant visualiser, sur les figure 17 et 19, les couples appliqués respectivement aux axes 2 et 4 que nous avons précédemment étudiés pour la cinématique différentielle inverse.

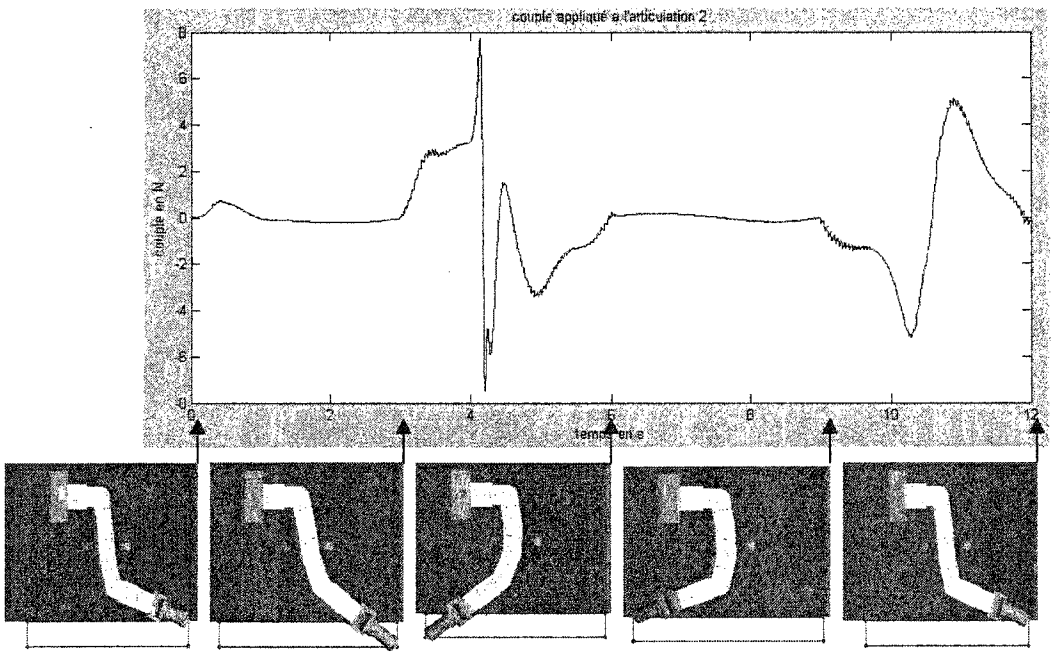


Figure 17 Couple appliqué à l'articulation 2

Nous remarquons, sur la figure 17, qu'entre les temps 3 et 6 secondes l'effecteur du robot est à la gauche du rectangle, c'est là où q_2 pourrait valoir plus de $\pi/2$, il est arrêté par la contrainte physique associée à cette articulation dans la fonction objective et aussi à cause du cylindre de gauche que pourrait rencontrer le robot. Cet arrêt brutal impose une rapide variation du couple appliqué. On remarque le même phénomène mais avec une variation moins importante entre 9 et 12 secondes. La variation est moins importante du fait que la contrainte physique associée à l'articulation 2 n'intervient pas de ce côté du rectangle.

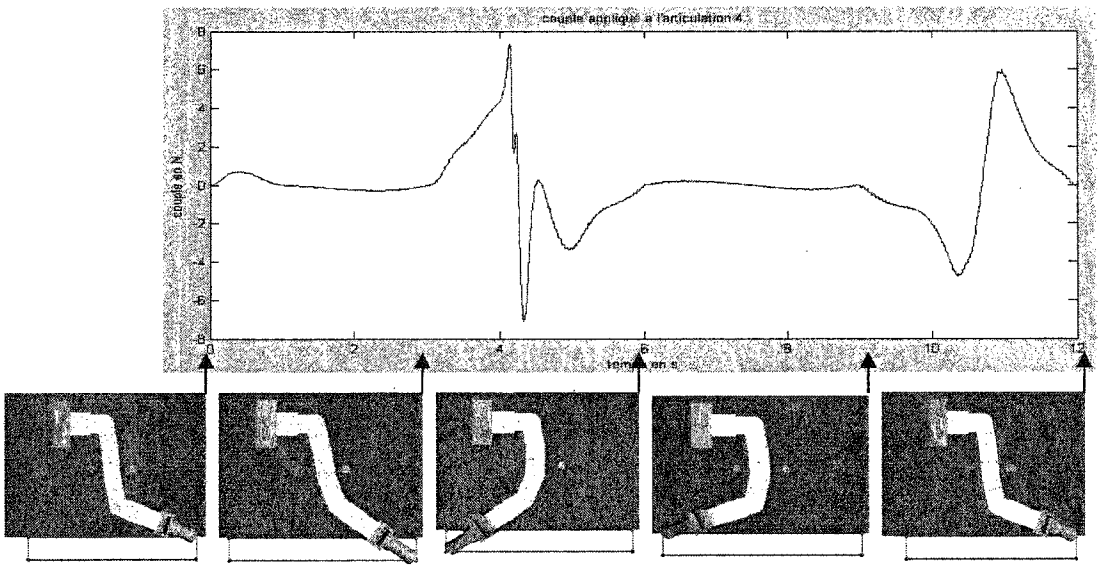


Figure 18 Couple appliqué à l'articulation 4

Pour ce qui est du joint 2, représenté sur la figure 18, nous remarquons entre les temps 3 et 6, que q_4 part pour aller plus que 0.8 radians et ainsi heurter le cylindre de gauche. Elle est arrêtée par la fonction objective d'une façon assez abrupte et ainsi provoque une variation importante du couple appliqué à cette articulation. Il en est de même entre les temps 9 et 12.

Pour ce qui est de l'erreur due à la commande adaptative, représentée sur la figure 19, elle est pratiquement négligeable, presque nulle sauf aux instants t =0, 4 et 10 où la variation rapide des couples appliqués entraîne une erreur d'environ $6x10^{-3}$ radians.

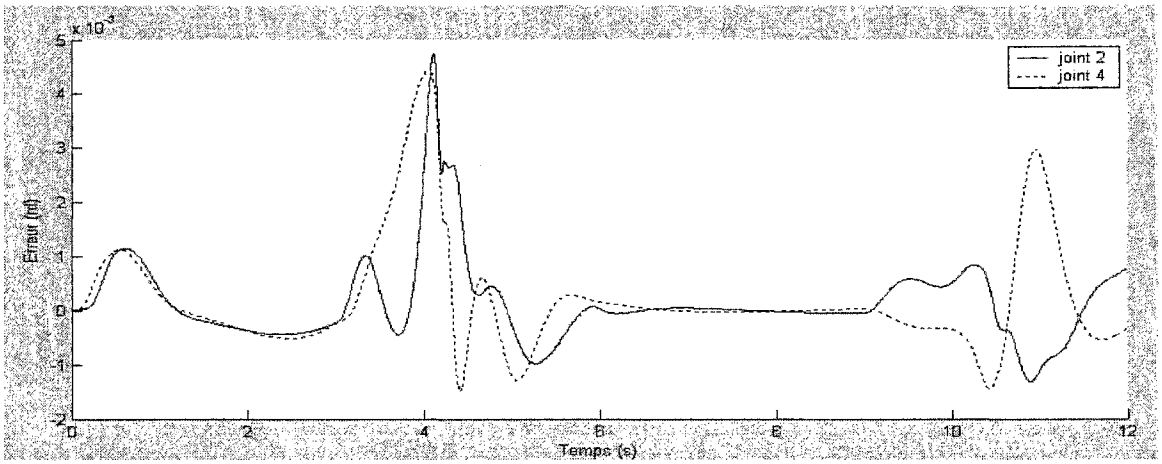


Figure 19 Erreur entre la trajectoire désirée avec évitement obstacle et celle du robot

CHAPITRE 4

IMPÉMENTATION DU CONTROLEUR EN TEMPS RÉEL

Nous avons maintenant l'algorithme permettant de commander le robot. Nos résultats de simulation sont satisfaisants, il nous reste donc à implanter cet algorithme en temps réel. Comme nous l'avons dit dans l'introduction, nous avons choisi le système d'exploitation QNX afin de commander le robot.

Nous avons dans un premier temps reprogrammé le code en C, afin de faire fonctionner l'algorithme sous QNX. Nous avons aussi créé une interface graphique, grâce au logiciel PhAB, permettant de rentrer les paramètres initiaux, de lancer une simulation ou une expérience et enfin de visualiser les résultats.

La partie développement sous QNX étant terminée, nous avons dû créer une interface entre les cartes de contrôle des moteurs, et l'ordinateur. Le robot étant modulaire, nous avons choisi d'associer une carte de contrôle à chaque moteur. Ces cartes récupèrent la position des moteurs grâce aux encodeurs et traduit le couple désiré en PWM afin de l'envoyer aux cartes de puissance.

Nous allons donc détailler dans ce chapitre les différentes étapes de l'implantation de l'algorithme.

4.1 Comparaison entre une architecture système hôte/DSP et hôte/QNX

Pour une architecture hôte/DSP, le programme est généralement développé sous l'hôte et ensuite envoyé sur le DSP où il s'exécute. Toute la partie interface : visualisation des résultats, modification des différents paramètres de simulation, gestion des entrées sorties du DSP, est traitée depuis l'hôte. Toute cette partie peut empêcher un système

d'être en temps réel. En effet, ces options créent des interruptions et ainsi modifient la période de l'horloge du système. En fait, cette gestion de l'interface par l'hôte permet de rendre la réponse du DSP déterministe.

Mais les inconvénients de ce système restent le coût et la flexibilité, en effet, le coût du matériel et des logiciels de développement sont assez élevés. De plus, le fait de développer un programme depuis l'hôte, pour ensuite l'envoyer vers le microcontrôleur, n'est pas optimal. Enfin, les entrées sorties restent limitées contrairement à un ordinateur.

Vous l'aurez compris, le DSP est très attrayant mais son environnement de développement n'offre pas de modularité et d'évolutivité comparable à un système tel que l'environnement QNX.

4.2 Architecture système hôte/QNX

Nous avons donc choisi de travailler sous le système d'exploitation QNX. L'architecture du robot est modulaire. Nous pouvons ainsi ajouter ou supprimer des joints sur la partie redondante du robot, suivant les différentes tâches à effectuer. Nous avons choisi d'associer à chaque axe une carte de contrôle qui reçoit le couple et renvoie la position des moteurs. Il nous reste donc à choisir le mode de communication entre l'ordinateur et ces cartes de contrôle.

4.2.1 Choix de l'interface entre la plateforme QNX et les cartes de contrôle

Pour que l'on puisse commander le robot d'une façon convenable, il faut que la période d'échantillonnage, c'est-à-dire le temps entre le calcul des couples des différents moteurs, l'envoi de ces couples aux cartes de contrôle et le retour des positions des différents moteurs, doit être de maximum 5 ms. En effet pour que les moteurs ne

reçoivent pas des variations de couple trop importantes et ainsi fonctionner sans à coups, nous devons considérer ces 5ms comme une limite.

Nous pouvons communiquer avec les cartes de contrôle suivant le protocole SPI, c'est un protocole proche de l'I2C mais nettement plus rapide, il peut aller jusqu'à 5MHz contre seulement 400 KHz pour l'I2C. Ce protocole permet donc de respecter amplement la limite d'échantillonnage. Nous devons maintenant trouver un moyen de sortir les informations de l'ordinateur selon le protocole SPI.

4.2.1.1 Le protocole de communication SPI

Le protocole SPI est un protocole de communication série. C'est un protocole de liaison entre un composant « maître » et un ou plusieurs composants « esclaves ». Il fonctionne par rotation des registres entre le maître et un esclave. Il est composé de quatre fils :

- un chip select appelé « SS »
- un fil permettant à l'esclave de recevoir la donnée et donc au maître de l'envoyer « MOSI »
- un fil permettant au maître de recevoir la donnée et donc à l'esclave de l'envoyer « MISO »
- un signal d'horloge « CLK »

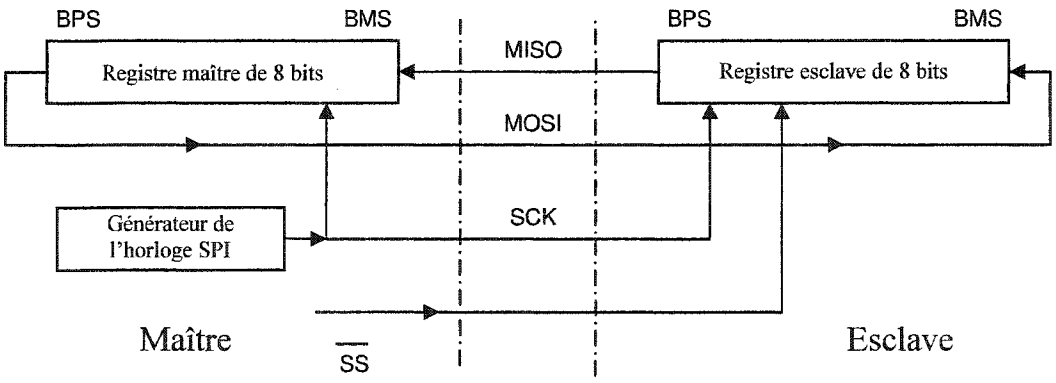


Figure 20 Connexion SPI entre un composant maître et un esclave

La figure 20 représente un échange suivant le protocole SPI. Le maître sélectionne dans un premier temps l'esclave en passant à 0 le bit SS. Après cette action, les deux octets à transférer sont placés dans les registres esclave et maître. Enfin le contrôle de transfert par la ligne d'horloge est effectué. Le transfert étant circulaire, le registre du maître est remplacé par celui de l'esclave et inversement. Le transfert peut s'effectuer en envoyant le bit le plus significatif (BPS) en premier, comme le montre la figure, ou alors en commençant par le bit le moins significatif (BMS).

4.2.1.2 L'interface USB / SPI

Nous avons, dans un premier temps, essayé de trouver une carte d'entrée sortie PCI/SPI mais les seules cartes disponibles étaient hors de prix ou alors ne possédaient pas de driver pour QNX. Nous nous sommes ensuite intéressés au port RS 232 mais sa vitesse n'était pas suffisante pour notre application.

QNX possède une librairie de fonction permettant de faire des envois de donnée suivant le protocole USB. Le protocole USB 2 permet d'envoyer des données à une vitesse de 12 Mo/s, ce qui est amplement suffisant pour notre application. Nous avons donc recherché un microcontrôleur USB possédant aussi une sortie SPI.
Plusieurs fabricants offrent ce genre de produit, mais nous avons choisi ATMEL pour cause de compatibilité. En effet, les cartes de contrôle que nous utilisons possèdent des contrôleurs ATMEL. Par ce fait, il nous est apparu judicieux d'utiliser un microcontrôleur USB de cette marque.

Nous présentons sur la figure 21 l'architecture globale du système avec la sortie du PC par le port USB, la traduction de l'information par le microcontrôleur en SPI et enfin l'arrivée aux cartes de contrôle.

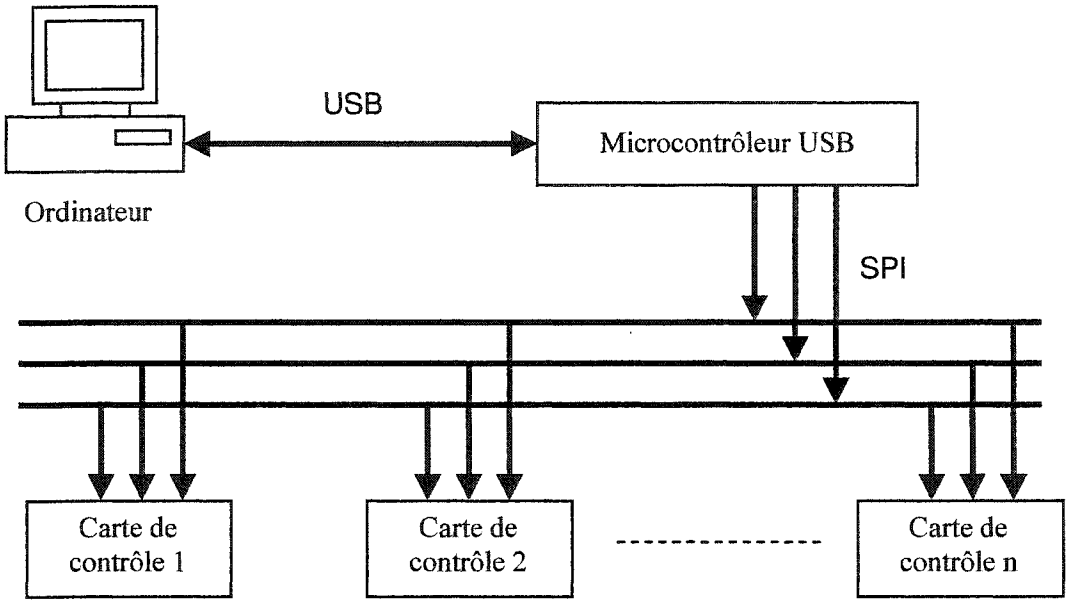


Figure 21 Architecture du système

4.3 la représentation mécanique du système

Le Système, représenté par la figure 22, est composé du robot bien sûr, du module d'alimentation des moteurs, de l'ordinateur et de tout le câblage qui relie ces différents éléments. La modularité de la partie redondante du robot peut faire varier sa taille, toutefois le système au complet occupe environ trois mètres carré.

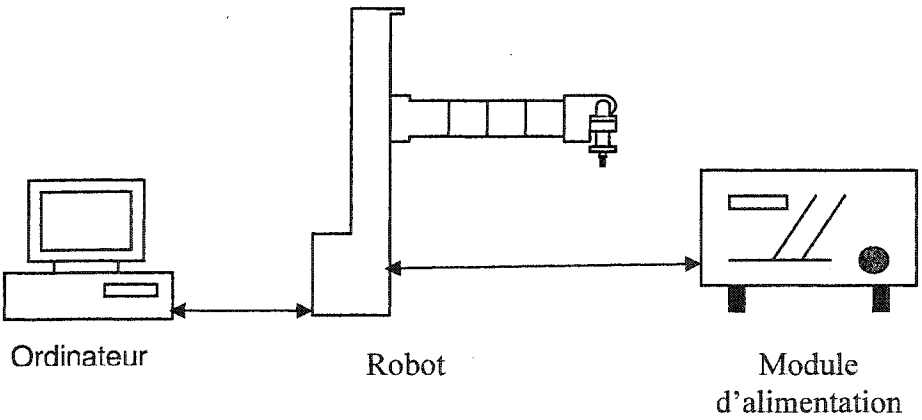


Figure 22 Design Mécanique

4.4 Le design électrique du système

Le cœur du système est l'ordinateur, composé d'un processeur AMD Athlon de 756MHz. C'est sur cet ordinateur que l'algorithme de commande du robot a été programmé. Cet algorithme, comme nous l'avons vu dans les chapitres précédents, calcule le couple à envoyer aux moteurs présents sur le robot, en fonction de la position des différents joints. Afin de bien comprendre le rôle des différents composants électriques, suivons, sur la figure 23, le chemin de l'information et ce de l'envoi du couple par l'ordinateur au retour de la position des joints à celui-ci.

L'ordinateur envoie le couple des différents moteurs du robot via le port USB. L'information arrive au microcontrôleur USB qui la traduit en SPI. La carte de contrôle de chaque axe reçoit alors le couple qu'elle traduit en PWM et envoie ensuite à la carte de puissance associée à cet axe. La carte de puissance renvoie alors la position, qui passe par la carte de contrôle pour être codée en SPI. Cette position retourne à l'ordinateur via le microcontrôleur USB. Nous représentons ce design sur la figure 23.

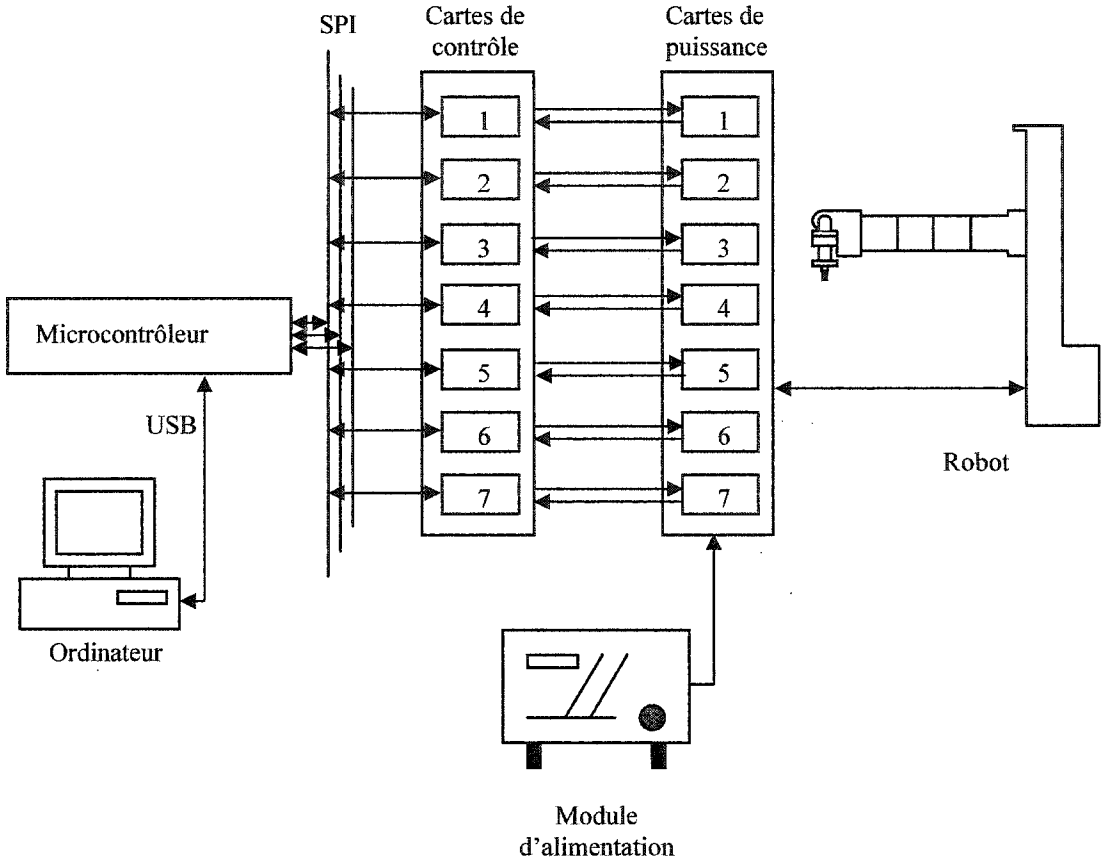


Figure 23 Design électrique

Détaillons maintenant le rôle des différents composants.

4.4.1 L'alimentation des différents modules

L'alimentation de l'ordinateur est classique, il fonctionne sous les 110 volts alternatifs du réseau. Le microcontrôleur USB est alimenté, via le port USB branché à l'ordinateur, par une alimentation de 5V. Les cartes de contrôle sont alimentées en 5V également. Le robot, et ainsi les cartes de puissance, sont alimentés via le « module d'alimentation ». Les moteurs du robot reçoivent une tension de 24V en continu.

4.4.2 L'ordinateur sous environnement QNX

C'est le moteur central du système, il exécute l'algorithme de commande adaptative avec évitement d'obstacles. il envoie également les couples des sept moteurs via le port USB et reçoit la position de ces mêmes moteurs via l'USB. Tout ce travail est évidement réalisé en un laps de temps connu, et ceci, grâce à la stabilité de l'horloge interne de QNX. Ceci nous permet de réaliser une commande en temps réel.

4.4.3 Le microcontrôleur USB AT43USB355

Ce microcontrôleur joue un rôle d'interface entre l'ordinateur et les cartes de contrôle. Comme nous l'avons expliqué précédemment, les cartes de contrôle reçoivent l'information en mode SPI et ce microcontrôleur possède une sortie SPI. Nous avons aussi choisi un microcontrôleur USB car la vitesse de transfert de l'USB nous permet de transmettre et de recevoir les données, sans dépasser les contraintes de temps.

Le microcontrôleur utilisé est le AT43USB355 d'ATMEL, présent sur la carte de développement AT43DK355. Cette carte est livrée avec un logiciel permettant de programmer le microcontrôleur. Le programme du micro est relativement simple, recevoir le couple à appliquer aux sept moteurs via le port USB, dépaqueter l'information et la placer dans le registre SPI. Et enfin, envoyer un couple à chaque carte de contrôle en activant le « chip select » adéquat.

Il reçoit également la position des différents moteurs par le port SPI qu'il empaquette suivant le protocole USB pour la retourner à l'ordinateur.

Par le port USB, nous prévoyons de transmettre au microcontrôleur un mot de 8 octets qui réunira les 7 couples à appliquer aux différents moteurs codés, chacun sur un octet, ainsi qu'un huitième octet indiquant le signe des sept couples précédent soit un bit par

couple. Il reste donc un bit pour cet octet qui ne nous importe pas. Nous effectuerons le même codage pour le retour des positions des moteurs, soit un octet pour chaque moteur et un octet indiquant les sept signes des positions des moteurs.

En ce qui concerne le transfert en mode SPI, nous enverrons successivement à chacune des cartes de contrôle :

- un octet pour donner la valeur du couple et recevoir une valeur non significative.
- un octet pour donner le signe du couple et recevoir la position du moteur.
- un octet non significatif afin de recevoir le signe de la position du moteur.

Nous aurons donc besoin de décoder l'information reçue par le port USB pour la transmettre par le port SPI et inversement, notamment pour les signes des couples et des positions.

4.4.4 Les encodeurs

Afin de pouvoir contrôler le robot, nous avons besoin de connaître la position des différents moteurs qui le compose. Les encodeurs angulaires nous permettent de faire correspondre des impulsions électriques à un déplacement en degré.

Les encodeurs génèrent des impulsions électriques grâce à des disques composés successivement de segments clairs puis foncés qui renvoient ou non un faisceau lumineux vers un capteur. Nous obtenons donc un signal carré de fréquence variable, en fonction de la vitesse du moteur, qui nous permet de quantifier le déplacement.

Si nous observons la figure 24, deux systèmes de comptage, déphasés de 90 degrés, nous permettent de déterminer le sens de rotation du moteur.

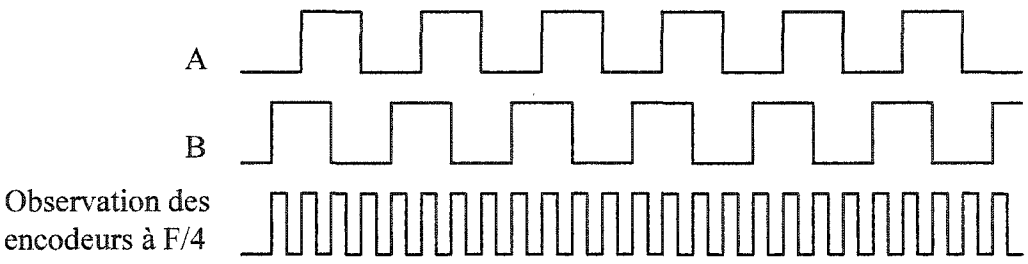


Figure 24 Observation des encodeurs

Une observation des résultats au quadruple nous permet d'augmenter la résolution de l'observation. En fait, nous pouvons ainsi obtenir des informations sur les fronts montants et descendants des signaux A et B. Les deux signaux A et B sont traités de manière à avoir deux signaux qui seront envoyés à des compteurs présents sur la carte de contrôle. Le premier signal enverra des impulsions au compteur lorsque le moteur tournera dans un sens. Le second enverra des impulsions lorsque le moteur tournera dans l'autre sens.

Par la suite, nous allons réaliser un étalonnage géométrique des encodeurs pour faire correspondre à une impulsion, un déplacement en degrés ou en mètres suivant l'articulation.

4.4.5 Les cartes de contrôle

Les cartes de contrôle, représentées figure 25, permettent de transformer le couple reçu en PWM et transformer les impulsions reçues par les encodeurs en positions. De plus elles réalisent un asservissement de courant pour les moteurs.

Le principal composant de ces cartes est un microcontrôleur ATMEGA16 de ATMEL. Ce microcontrôleur reçoit du microcontrôleur USB, le couple à appliquer au moteur et lui renvoie la position de celui-ci en nombre d'impulsions.

Afin de réaliser un asservissement du moteur, il reçoit de celui-ci la valeur du courant qui parcourt le moteur. Grâce à ce courant et au couple reçu, il transmet au moteur la valeur du PWM ainsi que la direction que doit prendre le moteur.

Il reçoit enfin les signaux traités des encodeurs pour renvoyer la position via son port SPI au microcontrôleur USB.

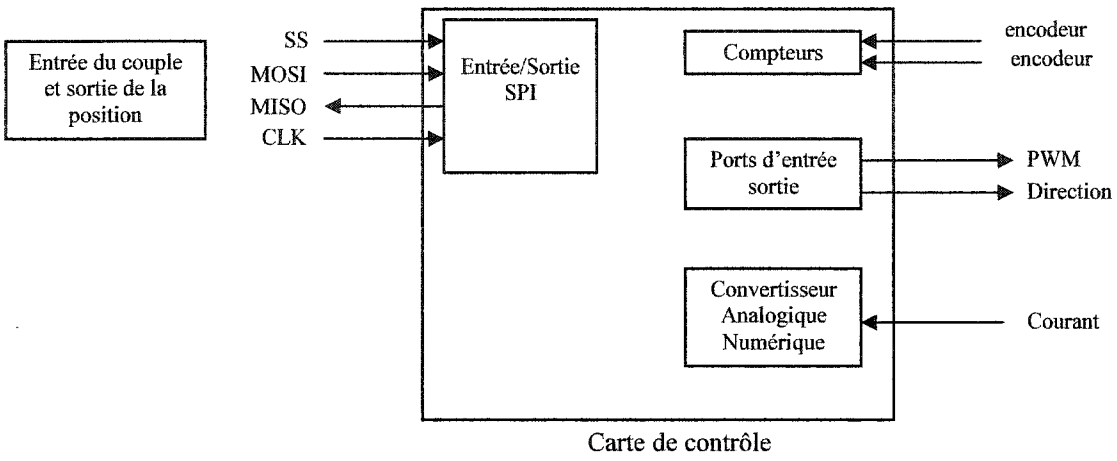


Figure 25 Schéma bloc des cartes de contrôle

4.5 Interface graphique sous QNX à l'aide du logiciel PhAB

Cette interface a pour but de rendre les simulations et les expériences plus simple à exécuter. Elle nous permet aussi de visualiser les résultats ou encore d'enregistrer des expériences. Nous la détaillons dans cette section.

4.5.1 Acquisition des paramètres

La première fenêtre de l'interface, figure 26, permet de choisir entre l'acquisition des paramètres, le lancement de la simulation, le lancement de l'expérience et la visualisation des résultats sous forme de courbes. Commençons par l'acquisition des

paramètres. Nous créons ou ouvrons d'abord le fichier dans lequel nous enregistrons ou lisons les paramètres nécessaires à la simulation ou l'expérience.

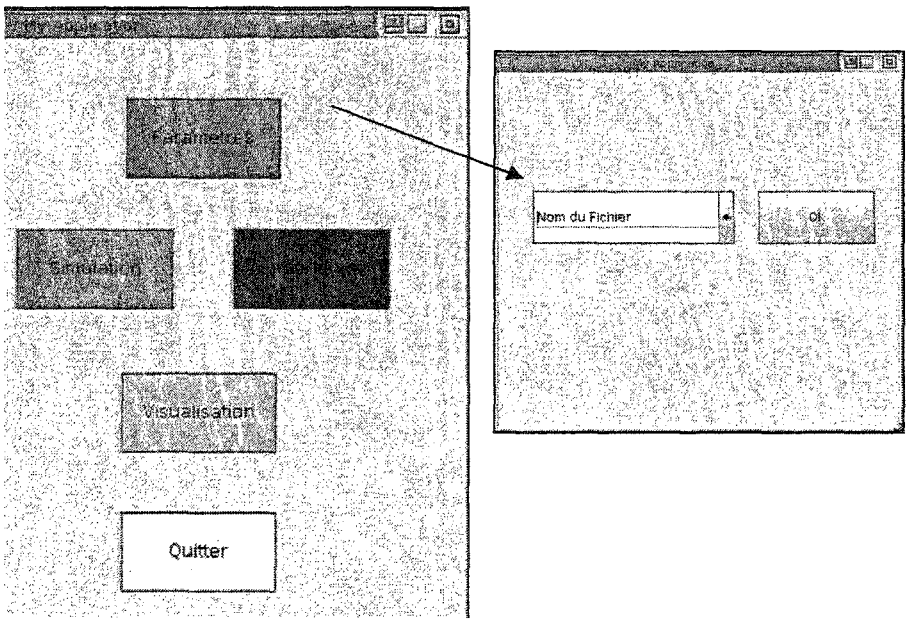


Figure 26 Fenêtre du menu principal

Ce fichier choisi ou créé, la fenêtre suivante s'ouvre, figure 27. Elle nous permet de rentrer tous les paramètres initiaux nécessaires pour l'expérience.

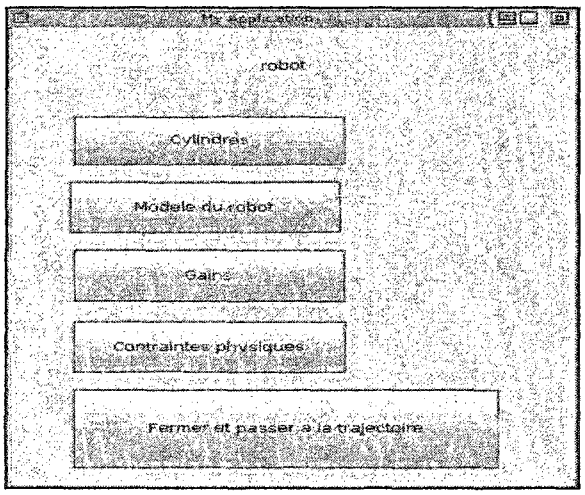


Figure 27 Fenêtre d'initialisation

Cliquons sur le bouton cylindre afin d'initialiser les paramètres associés à ce bouton. Les valeurs présentes dans les fenêtres (figure28) sont alors, associées au fichier d'initialisation choisi ou, si nous créons un nouveau fichier, initialisées à zéro. Il est évidemment possible de changer les valeurs affichées dans l'interface. Celles-ci convenant, nous les sauvegardons en appuyant sur le bouton prévu à cet effet. Voici, à titre d'exemple, la fenêtre associée au cylindre 2. Nous rappelons qu'il est possible d'avoir jusqu'à trois cylindre pour une simulation.

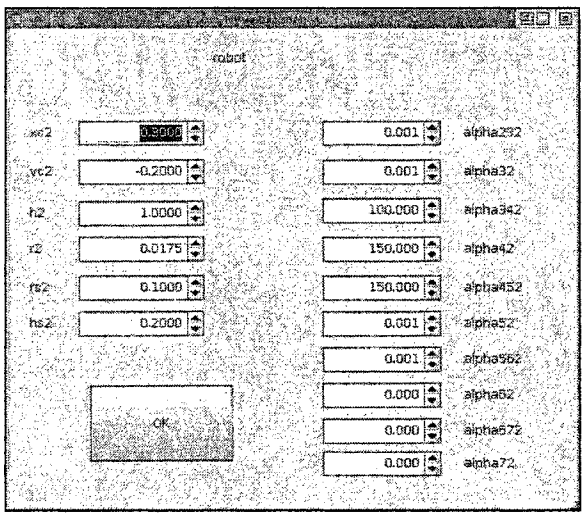


Figure 28 Fenêtre d'initialisation du cylindre 2

Dans cette fenêtre nous pouvons initialiser la position, la hauteur et le rayon du cylindre mais aussi les paramètres associés à celui-ci pour l'évitement d'obstacles.
Sur le même principe, nous pouvons modifier les paramètre dynamiques du robot, modifier les gains de la commande adaptative ou encore modifier les gains afin que le robot respecte ses limites physiques. Ces paramètres initialisés, nous pouvons initialiser la trajectoire.

Dans la fenêtre point, figure 29, nous rentrons les coordonnées des coins du polyèdre qui vont représenter la trajectoire de l'effecteur du robot. Comme ce nombre de points est

variable selon les expériences à effectuer, nous enregistrons successivement leurs coordonnées.

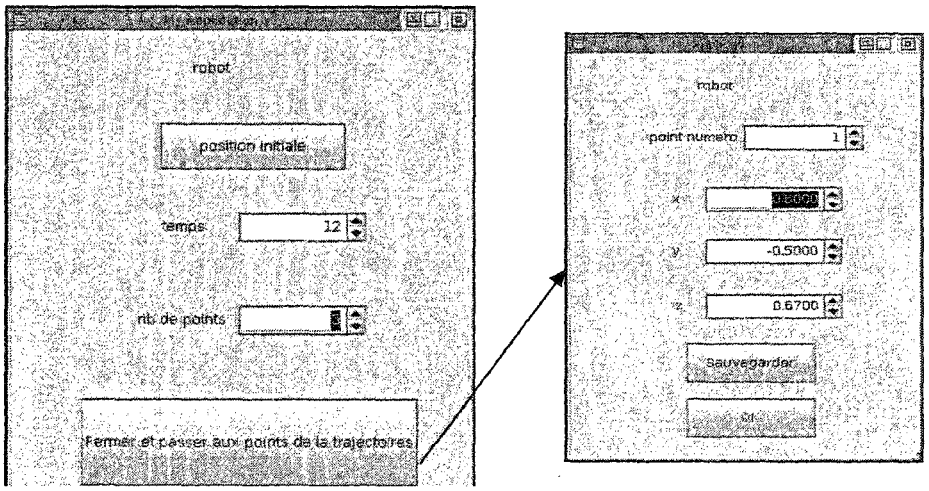


Figure 29 Fenêtre d'initialisation de la trajectoire.

Nous pouvons également entrer la position initiale du robot et ce, dans l'espace des joints.

4.5.2 Lancement de la simulation

Les paramètres étant enregistrés, nous nous en servons pour lancer la simulation figure 30. L'utilisateur va d'abord choisir le nom du fichier dans lequel on a enregistré les paramètres initiaux.

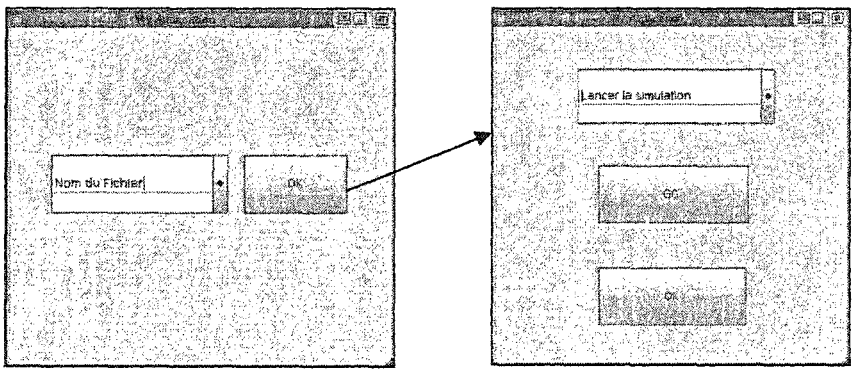


Figure 30 Fenêtre de lancement de simulation

Le programme de simulation a dû être compilé séparément avec le terminal en créant une librairie statique. L'interface appelle ensuite l'exécutable. Et la simulation est réalisée en fonction de l'initialisation des paramètres.

4.5.3 Visualisation des résultats de simulation

Une fois que les résultats ont été enregistrés, il est possible de les visualiser (figure 31). Les résultats sont obligatoirement enregistrés dans un fichier nommé « résultats ». Cependant il suffit de renommer ce fichier pour le garder. Pour afficher les courbes, il a fallu créer les fenêtres sans PhAB en écrivant des programmes appelés par des boutons de l'interface.

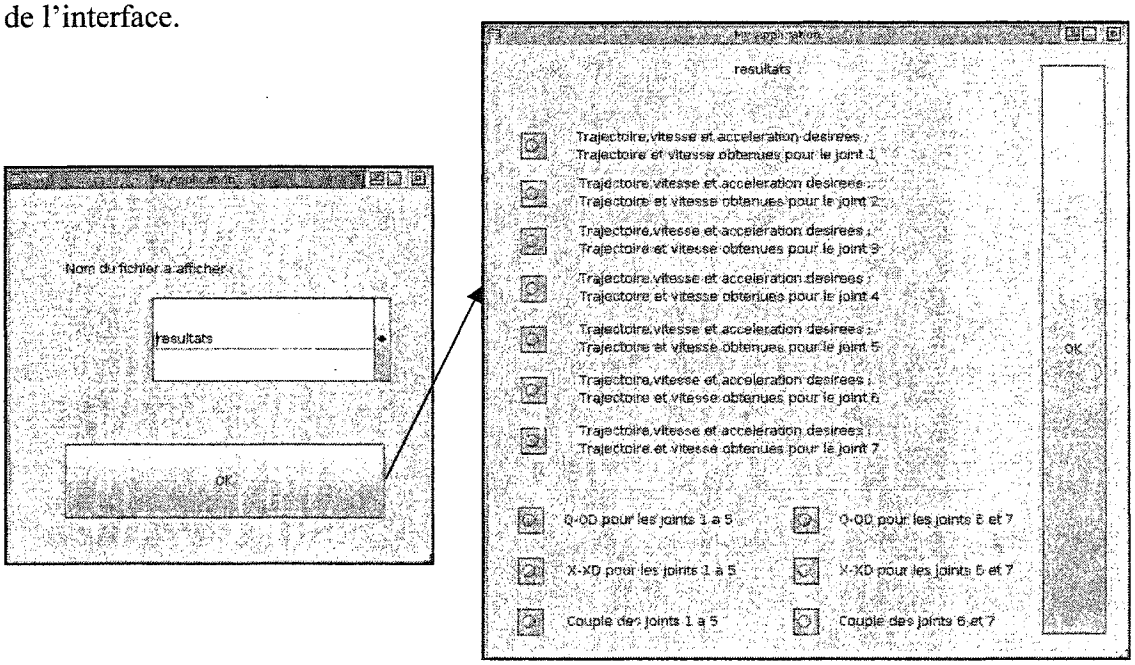


Figure 31 Fenêtre de choix du fichier de résultats à visualiser

Cette fenêtre nous permet de choisir les différentes courbes que nous pouvons visualiser. Nous avons le choix entre la trajectoire, la vitesse et l'accélération désirées et celles obtenues pour chaque joint, les différentes erreurs de suivi et enfin le couple appliqué aux différents joints.

4.5.4 Simulations et résultats graphiques

Nous montrons ici deux exemples de courbes obtenues, figures 32 et 33. Nous les avons comparées avec les courbes correspondantes obtenues grâce à Simulink, et elles concordent. Le premier exemple montre la position, la vitesse et l'accélération désirées ainsi que la position et la vitesse obtenues. Le deuxième exemple montre les couples des joints un à cinq.

Les résultats concordent bien avec ceux obtenus avec Simulink, une petite différence peut être obtenue, dépendamment du choix de la période d'échantillonnage.

Pour le contrôle du robot en temps réel, au lieu de choisir l'option simulation, nous pouvons choisir l'option expérience. Le couple est alors envoyé au robot.

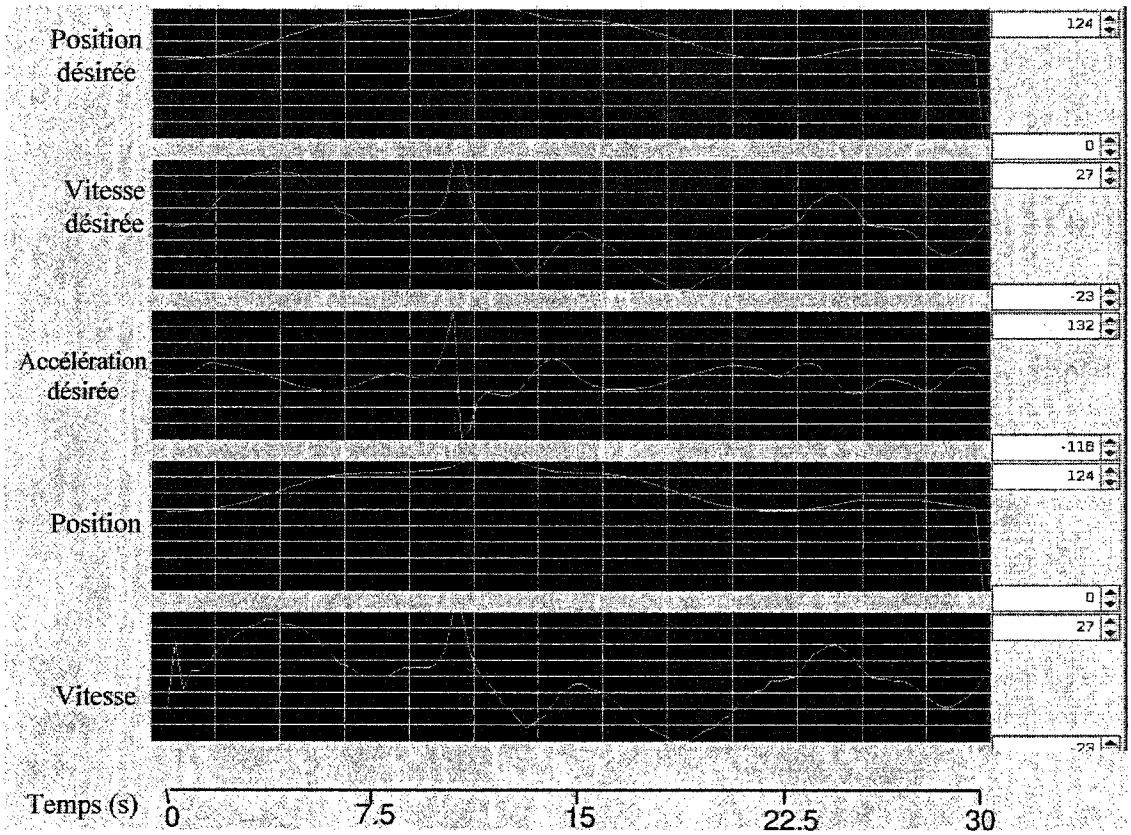


Figure 32 Position, vitesse et accélération désirées et position et vitesse obtenues du joint 1

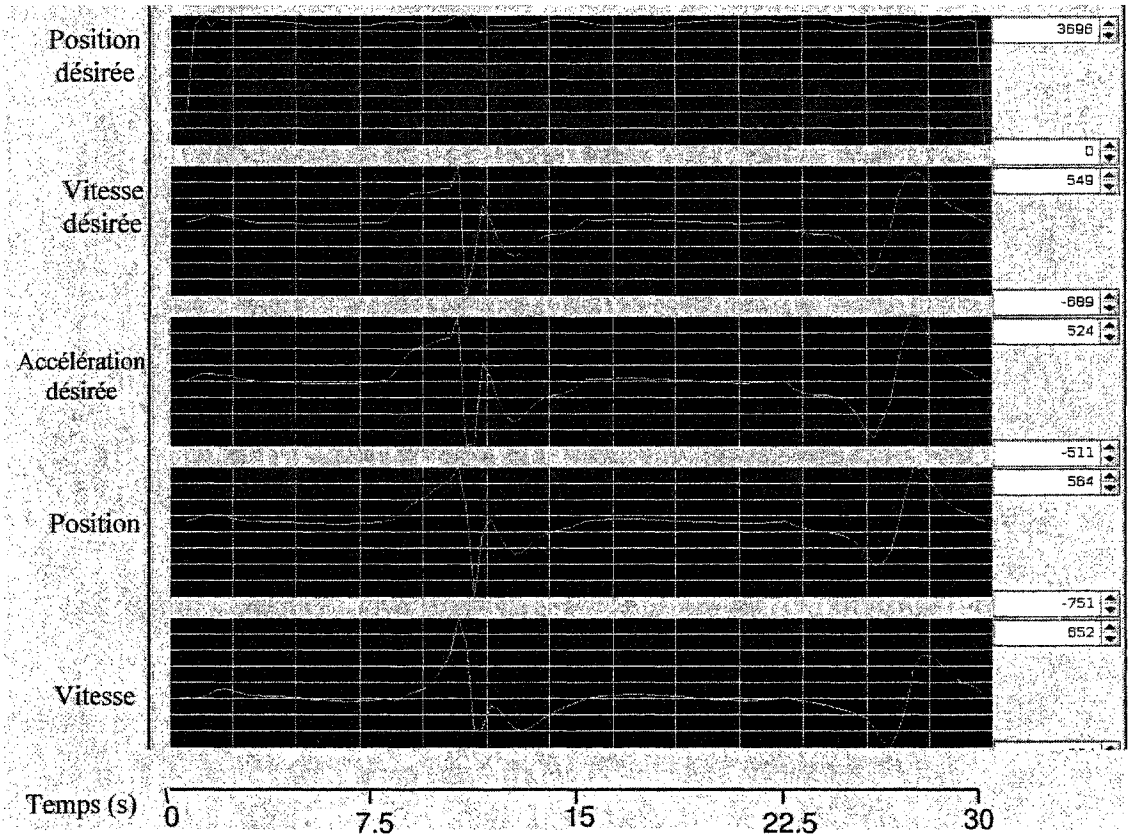


Figure 33 Couple appliqué aux joints 1 à 5

4.6 Contrôle en temps réel

Nous connaissons maintenant les différents outils qui nous permettent de contrôler le robot. Les différents composants sont : l'ordinateur sous environnement QNX qui gère l'algorithme de commande adaptative, le microcontrôleur USB qui interface l'ordinateur et les cartes de contrôle et enfin les cartes de contrôle qui récupèrent les positions et envoient les PWM désirés aux moteurs.

Nous allons voir dans cette partie le côté plus pratique du projet, la conversion des impulsions des encodeurs en degrés ou mètres pour déterminer la position du robot, la conversion du couple à appliquer aux moteurs en PWM, et enfin, le transfert de données via le microcontrôleur USB.

4.6.1 Conversion du couple en PWM

Les moteurs utilisés sont à courant continu. Les cartes de puissance, les alimentant, reçoivent un signal PWM (modulation de largeur d'impulsion), pour faire varier leur vitesse. Il nous faut donc convertir le couple en signal PWM.
Commençons par étudier les équations qui régissent le moteur à courant continu en régime statique.

L'équation aux mailles nous permet d'écrire :

$$V = E + R * I \tag{6.1}$$

Où:

- V représente la tension appliquée aux bornes du moteur.
- E représente la force électromotrice du moteur.
- R la résistance des armatures du moteur.
- I le courant absorbé par l'induit.

La force électromotrice est régie par l'équation suivante :

$$E = k\Phi\Omega \tag{6.2}$$

Où :

- k représente une constante du moteur.
- Φ représente le flux dans l'induit.
- Ω représente la vitesse de rotation du moteur

Enfin l'équation régissant la puissance du moteur :

$$P = E\,I = \tau\,\Omega \quad \Rightarrow \quad \tau = k\,\Phi\,I \tag{6.3}$$

Où τ représente le couple appliqué au moteur.

En combinant ces équations, nous obtenons la tension appliquée au moteur en fonction de la vitesse et du couple :

$$V = k\Phi\Omega + \frac{R\tau}{k\Phi} \tag{6.4}$$

Regardons maintenant, sur la figure 34, un signal PWM qui est un signal carré de fréquence constante mais de rapport cyclique r_c variable et d'amplitude V_a égale à la tension nominale du moteur. Le moteur reçoit une tension proportionnelle au rapport cyclique :

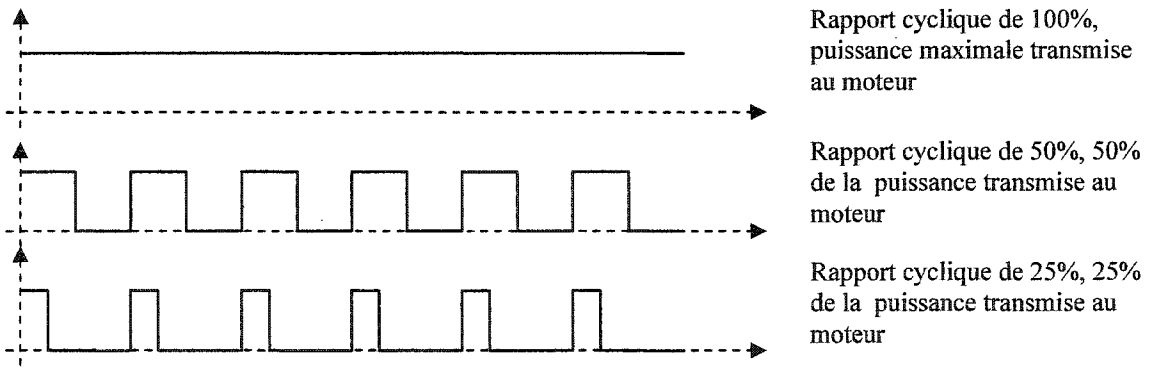


Figure 34 Signal PWM de rapport cyclique variable

La valeur de la tension appliquée au moteur est donc

$$V = V_a \times r_c \tag{6.5}$$

Si nous combinons les équations 6.4 et 6.5 nous obtenons la valeur du rapport cyclique du signal PWM en fonction du couple et de la vitesse du moteur :

$$r_c = \frac{k\Phi}{V_a}\Omega + \frac{R}{k\Phi V_a}\tau \tag{6.6}$$

4.6.2 Étalonnage des encodeurs

Afin de faire correspondre aux pulsations générées par les encodeurs, une position en degrés ou en mètres, nous devons effectuer quelques tests sur les différents axes du robot.

Nous allons donc déplacer chaque axe du robot de l'une à l'autre de ses limites physiques et observer le nombre d'impulsions envoyées aux différents compteurs.

Tableau IV

Correspondance entre les impulsions et les mesures physiques

Axe	Limites physiques en degrés ou mètres	Nombre d'impulsions par degré ou mètre
1	De L_0 = 0.57m à 1.27m	-
2	De -90° à +90°	49.3
3	De -90° à +90°	49.3
4	De -90° à +90°	49.3
5	De -90° à +90°	49.3
6	De 0° à 180°	49.3
7	De 0° à 360°	49.3

Dans le programme du microcontrôleur présent sur la carte de contrôle, nous ajoutons le nombre d'impulsions dans le sens positif et nous soustrayons le nombre d'impulsions dans le sens négatif. Nous obtenons ainsi une position en « impulsion » que l'on envoie à l'ordinateur. L'ordinateur réalise ensuite la conversion en unités physiques.

4.6.3 Transmission des couples et des positions via le microcontrôleur USB et les cartes de contrôle

Notre algorithme de commande adaptative utilise l'intégration numérique. Nous devons donc avoir un temps d'échantillonnage de moins de 5 ms pour avoir des données cohérentes. Le temps de calcul pour obtenir les couples à envoyer aux moteurs est de 0.7 ms il nous reste donc environ 4.3 ms pour envoyer ces couples et recevoir les positions des différents moteurs.

Nous allons donc voir dans un premier temps, le programme d'entrée sortie que nous avons réalisé sous QNX pour communiquer avec le robot via le port USB. Puis nous allons regarder le programme du microcontrôleur USB. Enfin, nous reviendrons sur les temps de transferts de ces informations.

4.6.3.1 Programme d'entrée sortie via le port USB

Comme nous l'avons dit précédemment, l'environnement QNX possède une bibliothèque de fonctions permettant de communiquer via un port USB. Nous avons dû, dans un premier temps, étudier le protocole USB pour comprendre comment utiliser les différentes fonctions de la librairie. Nous ne rentrerons pas dans les détails de ce protocole assez complexe mais nous vous invitons à visiter le site du protocole USB [19]. Nous allons détailler le corps du programme d'entrée sortie pour comprendre le protocole :

- Il nous faut d'abord nous connecter à la pile du microcontrôleur USB; nous utilisons la fonction *usbd_connect()* ; cette fonction possède deux fonctions de

rappel, une pour l'insertion de l'appareil USB et une pour le retrait de cette appareil.

- Pour la fonction de rappel d'insertion :
  - Nous devons, dans un premier temps, nous attacher au contrôleur USB grâce à la fonction *usbd_attach()*. Cette fonction permet d'obtenir certains paramètres de connexion.
  - Nous appelons ensuite la fonction *usbd_descriptor()* qui permet d'obtenir les descripteurs USB et notamment les descripteurs d'interface et de configuration.
  - Enfin, nous ouvrons un canal de communication qui permet de transférer les informations au contrôleur USB avec la fonction *usbd_open_pipe()*. En fait, nous ouvrons deux canaux, un d'entrée et un de sortie pour les deux terminaisons (points d'arrivée ou de sortie de l'information) que nous utilisons sur le contrôleur USB.

- Pour la fonction de retrait de l'appareil :
  - Nous utilisons la fonction usbd_detach() afin de détacher l'appareil USB.

- Nous avons maintenant tous les paramètres nécessaires de l'appareil USB pour pouvoir commencer la communication. Il reste maintenant à savoir ce que nous désirons envoyer ou recevoir (taille et type du mot). Pour cela, nous utilisons les fonctions *usbd_setup_bulk(), usbd_setup_interrupt()* suivant le type de transfert que nous désirons réaliser.

- Enfin nous commençons le transfert grâce à la fonction *usbd_io()*.

Comme nous l'avons dit, nous transférons les couples par un mot de huit octets, sept octets pour les valeurs des couples des sept moteurs et un bit pour les sept signes de ces

couples. Et nous recevons un mot de seize octets, mais les seuls quatorze premiers octets nous importent, ce sont les positions des sept moteurs codées sur deux octets chaque. Ces données sont acheminées par transfert dit « bulk ».

4.6.3.2 Programme du microcontrôleur USB

Le microcontrôleur reçoit les huit octets de QNX. Il les range dans des registres en associant le signe à chacun de ces couples. Le microcontrôleur envoie ensuite l'octet représentant le couple du premier moteur. Il reçoit alors 128 pour vérifier la synchronisation, il envoie ensuite un octet pour indiquer le signe et reçoit l'octet le moins significatif de la position du moteur. Enfin, il envoie 128 pour vérifier la synchronisation et reçoit alors l'octet le plus significatif de la position du moteur. Si le microcontrôleur USB ne reçoit pas 128 lors du premier envoi, l'envoi du couple, il renvoie 128 puis le couple jusqu'à obtenir 128 lors de l'envoi d'un couple. Les envois seront alors de nouveau synchronisés.

Le microcontrôleur répète cette opération pour les sept moteurs et renvoie ensuite les 14 bits représentant les positions des sept moteurs à l'ordinateur.

4.6.3.3 Programme du microcontrôleur des cartes de contrôle

Le programme reste le même pour les sept axes. Tout d'abord, le microcontrôleur attend une donnée via le port SPI. Il reçoit le couple et renvoie 128 pour la synchronisation. Il calcule alors la position du moteur après avoir décodé les signaux des encodeurs et place l'octet le moins significatif de cette position dans le registre SPI. Il reçoit ensuite le signe du couple et renvoie l'octet le moins significatif de la position et place l'octet le plus significatif dans le registre SPI. Il reçoit alors 128 et envoie l'octet le plus significatif de la position. Il place ensuite 128 dans le registre SPI. Il regarde alors s'il a bien reçu 128 lors du dernier envoi pour vérifier la synchronisation. Si tel est le cas, il

réalise un contrôle proportionnel intégrale (PI) du moteur en regardant le courant circulant dans le moteur et en le comparant au couple envoyé, il en déduit le PWM à appliquer au moteur. Il attend alors l'envoi d'un nouveau couple.

4.6.3.4 Les problèmes de temps de transfert

Après avoir écrit le programme d'entrée sortie, nous avons commencé à réaliser certains tests pour envoyer et recevoir les données que nous avons détaillées dans la partie précédente. Le transfert fonctionne très bien jusqu'à une certaine vitesse. En fait, nous nous sommes rendus compte que nous ne pouvions pas envoyer et recevoir un mot de huit bits en moins de 3 ms, ce qui est largement au dessus des temps minimums que nous nous sommes accordés. En fait, le taux de transfert est amplement suffisant, mais seul le temps minimal entre un envoi et une réception importe dans notre cas. Ce temps minimal est fixé à 1 ms pour un transfert par interruption et est sensiblement le même pour un transfert en mode bulk. De plus ce temps minimal, même si ce n'est pas spécifié dans les données constructeurs et sur les spécifications de l'USB 2.0, croît lorsque la taille du mot envoyé croît. Ainsi il est encore plus long d'envoyer un mot de huit octets et de recevoir un mot de seize octets.

Dans ces conditions, l'interface limite le nombre de joints à commander, nous verrons dans les recommandations les différentes voies que nous pouvons prendre pour résoudre le problème.

Nous avons toutefois décidé de commander trois axes du robot, trois axes de la partie redondante. Nous avons donc une période d'échantillonnage de 9 ms. Même si cette expérience n'est pas optimale, elle va nous permettre de tester les transmissions en temps réel et toute la partie « hardware » du projet.

4.7 Expérience en temps réel sur trois axes du robot

4.7.1 Plan d'expérience

Pour cette expérience, nous commençons par appliquer une commande proportionnelle dérivée au robot. Cette loi de commande prend la forme suivante :

$$\tau = K_p(q_d - q) + K_d(\dot{q}_d - \dot{q}) \tag{6.7}$$

Où les matrices diagonales K_p et K_d sont respectivement les gains proportionnel et dérivé du correcteur.

Nous faisons suivre au robot une trajectoire planaire. La trajectoire sera suivie par le quatrième axe rotatif du robot. Nous plaçons deux cylindres d'un mètre de hauteur et de 35 millimètres de diamètre aux coordonnées [0.1,-0.15] et [0.3,0]. Nous demandons ensuite au robot de répéter une trajectoire qui suit dans l'ordre les quatre points suivants : [0.25,-0.33], [0.05,-0.25], [0.25,-0.33], [0.45,-01]. L'intérêt de cet environnement de travail est que l'on voit le robot travailler aux limites physiques de son espace de travail. En effet l'articulation numéro deux va pratiquement toujours travailler à +90°. De plus, nous imposons au robot d'utiliser au maximum son mouvement interne afin d'éviter les obstacles.

Les coefficients de la fonction objective sont tous à 0.01 sauf celui de l'axe 2 pour l'évitement de l'obstacle de gauche qui est à 10 et celui de l'axe 2 pour l'évitement de l'obstacle de droite qui est à 20 En effet, compte tenu de la tâche à réaliser, ils ont beaucoup plus de risque de heurter les cylindres. Il faut donc augmenter ces coefficients afin que l'évitement d'obstacle soit réalisé correctement. La trajectoire désirée est suivie en trente secondes. Observons l'environnement d'expérience sur la figure 35.

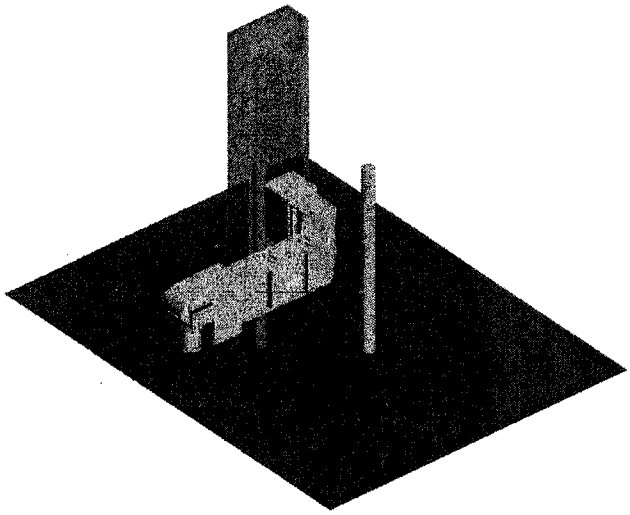


Figure 35 Environnement d'expérience

Observons maintenant les résultats de l'expérience grâce à l'interface réalisée avec le logiciel PhAB.

4.7.2 Analyse des résultats d'expérience.

Comparons les positions et vitesses désirées des différentes articulations aux positions et vitesses suivies par le robot.

Pour le premier joint, figure 36, nous observons un bon suivi de trajectoire, la position du robot est pratiquement identique à la position désirée. Pour ce qui est de la vitesse nous observons une forme d'onde identique mais un signal très bruité. Ceci s'explique à cause de la période d'échantillonnage assez importante de 9 ms qui augmente l'erreur lors de la dérivation. De plus, quelques erreurs de communication comme à t= 12 secondes crées des pics. Une solution serait de filtrer ce signal afin d'obtenir un signal plus propre.

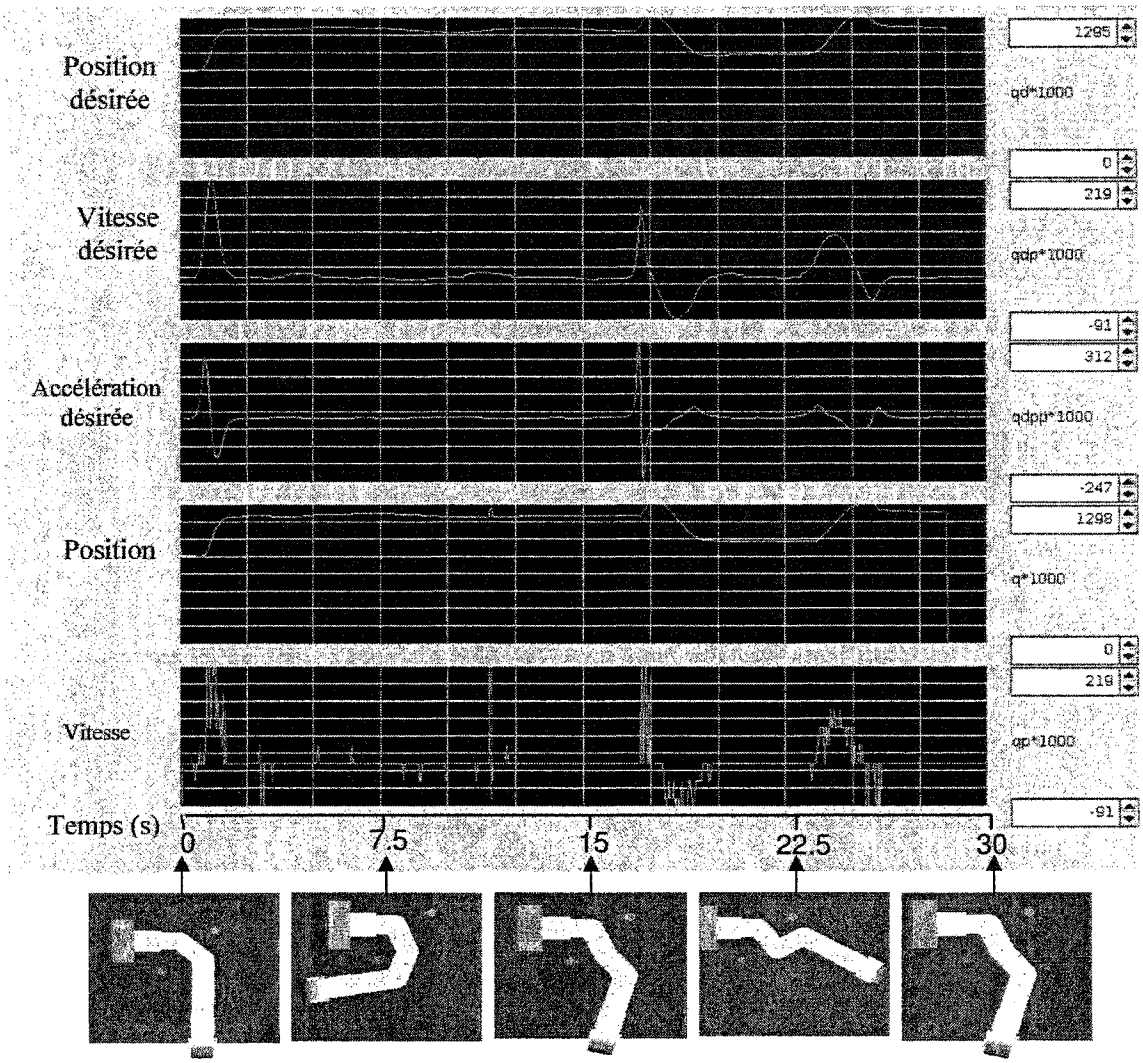


Figure 36 Résultats d'expérience pour le premier joint

Regardons maintenant les courbes pour le joint numéro deux.

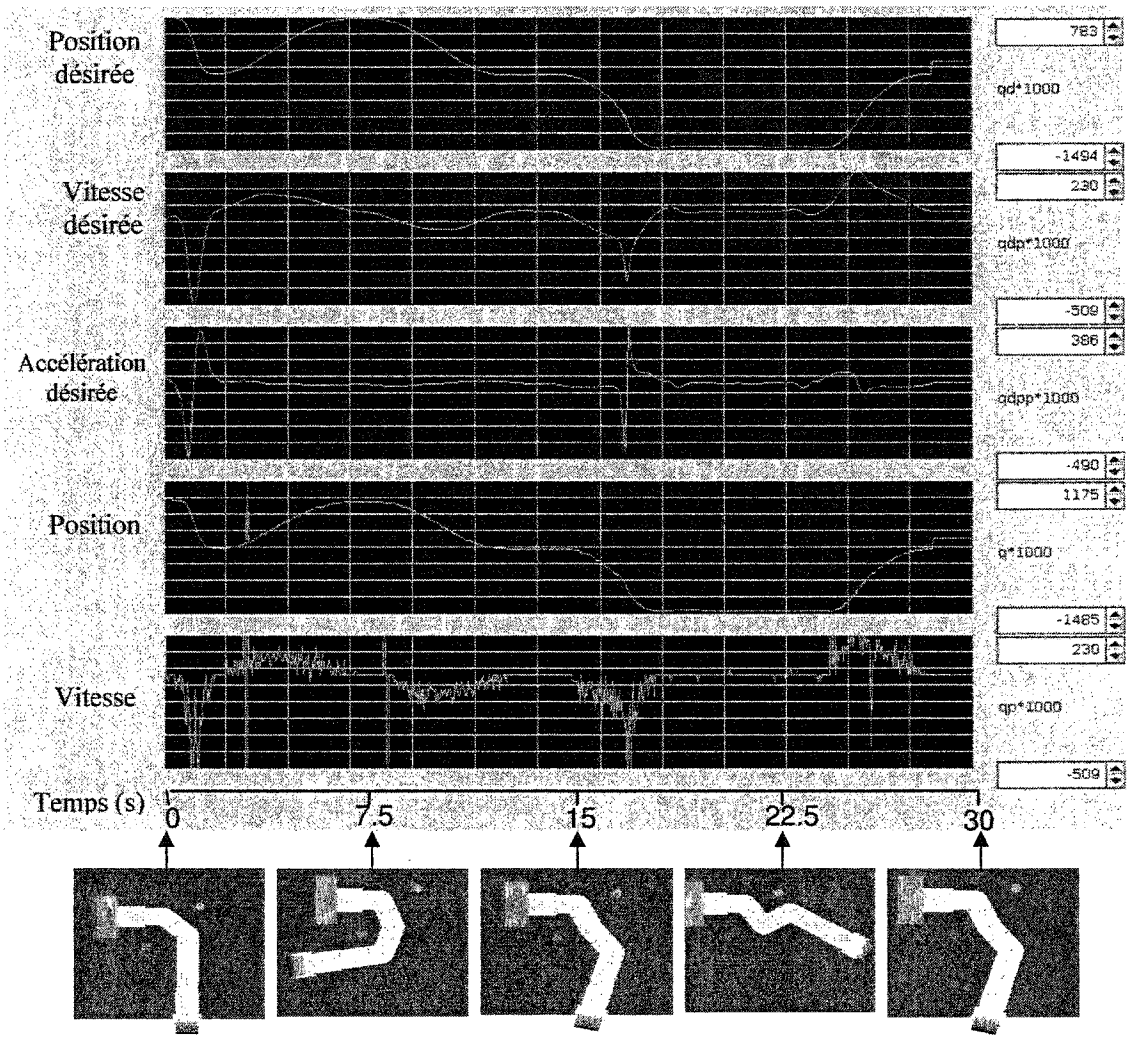


Figure 37 Résultats d'expérience pour le joint deux

Pour le joint numéro deux, figure 37, nous observons les mêmes résultats que pour le premier joint, un bon suivi de position mais une vitesse bruitée.

Enfin regardons les résultats pour le troisième axe sur la figure 38.

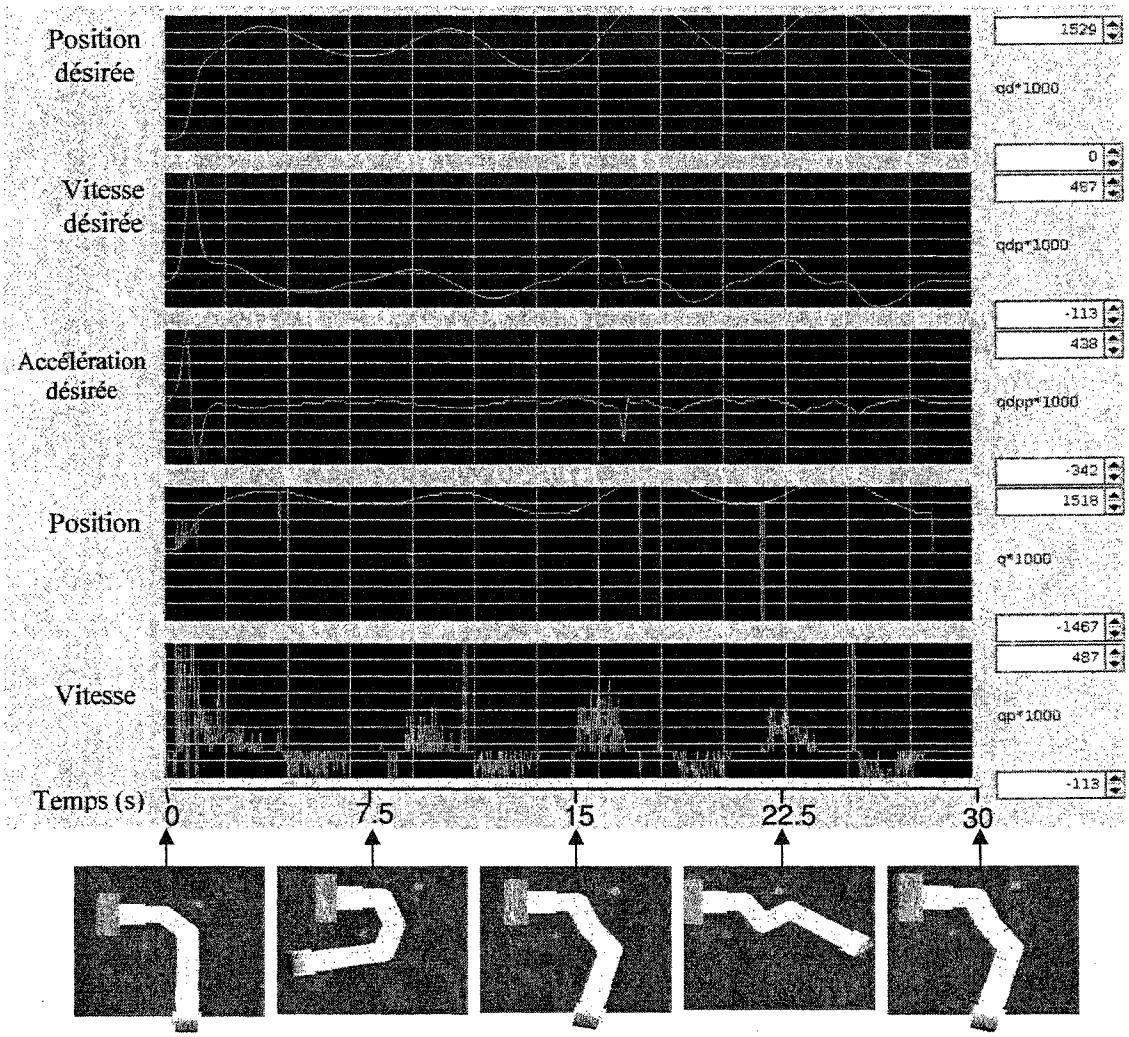


Figure 38 Résultats d'expérience pour le joint trois

La encore nous observons les mêmes résultats mais avec une vitesse plus bruitée due à sa variation plus rapide que pour les autres axes. Nous remarquons également des pics, sur la courbe représentant la position du robot, qui entraînent une variation importante sur le signal de vitesse.

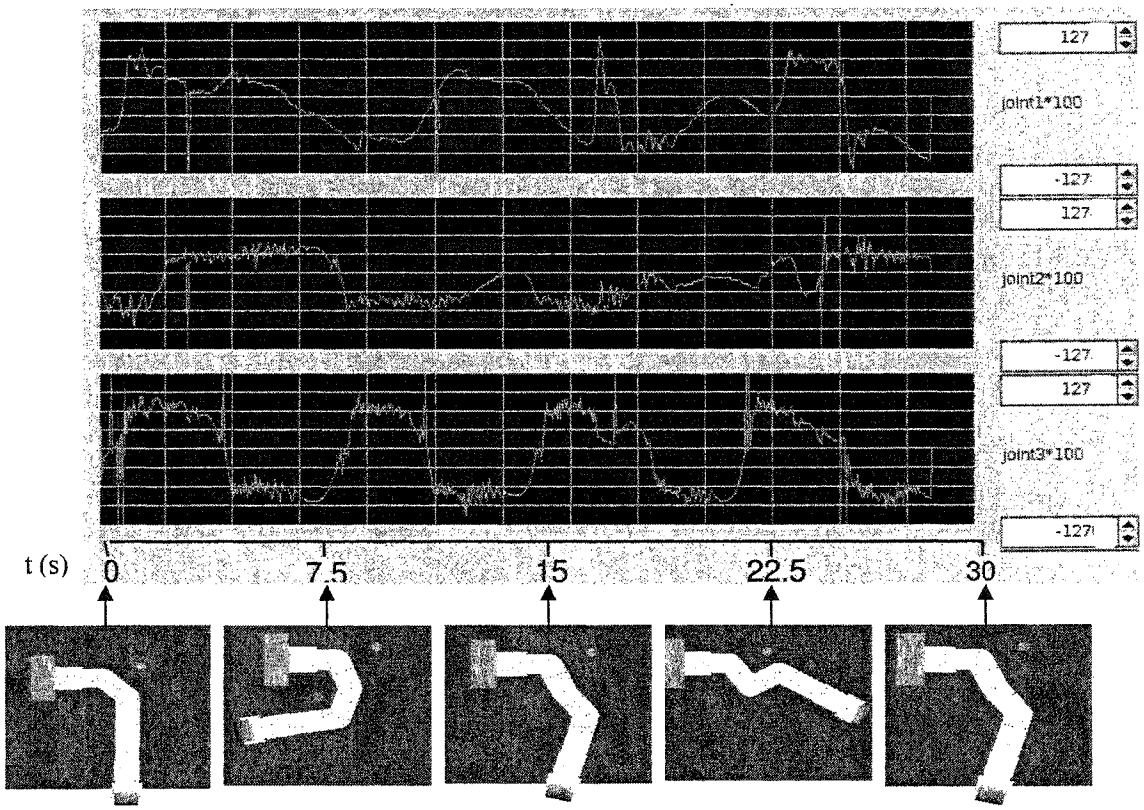


Figure 39 Couple appliqué aux trois joints du robot

Pour ce qui est du couple appliqué aux trois joints, figure 39, nous observons également une oscillation du signal, ceci est dû à la vitesse puisqu'elle fait partie de la commande proportionnelle dérivée.

Ces expériences permettent de valider la partie communication entre le robot et l'ordinateur. Toutefois la période d'échantillonnage étant trop importante, nous observons du bruit sur le signal de vitesse des différents joints du robot. Nous ne pouvons donc pas appliquer la commande adaptative.

DISCUSSION ET INTERPRETATION DES RÉSULTATS

Le projet que nous venons de terminer était de réaliser une plateforme de commande modulable pour le robot ANAT. Cette plateforme composée de trois éléments principaux comprend : un ordinateur opérant sous QNX, une interface avec le robot et le robot en lui-même. En outre, nous avons développé un algorithme d'évitement d'obstacles afin d'exploiter la redondance du robot.

Tout au long du développement, nous avons gardé à l'esprit que le nombre de joints du robot, composé de sept, peut en fait, varier suivant les applications que nous devons réaliser. Cette variation ne devait pas entraîner de modifications importantes, que ce soit au niveau du code informatique ou des cartes de contrôle du robot. Ainsi nous devions être capable de faire varier le nombre de joints sans avoir, d'une part à reprogrammer tout le code et l'interface graphique et d'autre part refaire tous les branchements électriques entre l'ordinateur et le robot.

Pour ce qui est de la partie programmation, nous avons développé un algorithme de commande adaptative avec une cinématique différentielle à évitement d'obstacle, dont nous commenterons les performances plus tard. Cet algorithme peut très bien être utilisé pour commander un robot avec un nombre d'axes variable. Nous l'avons d'ailleurs appliqué pour sept axes, six axes (tous sauf l'axe prismatique), ou encore quatre et trois axes (les axes parallèles formant la partie modulaire du robot). Nous avons donc rempli nos objectifs pour la modularité du code informatique.

Pour ce qui est maintenant de la partie hardware, le choix d'attribuer une carte de contrôle par axe et de communiquer avec elles par un protocole série, le protocole SPI, nous permet d'avoir cette modularité au niveau « physique ». En effet, si nous ajoutons un axe par exemple, il nous suffit d'ajouter une carte de contrôle et de connecter les quatre fils du protocole SPI et nous pouvons commander le robot avec un axe de plus.

Les cartes étant identiques, cela reste très pratique. Mais ce choix de protocole nous a posé beaucoup de problèmes étant donné que nous n'avons pas réussi à trouver de carte d'entrée sortie ayant ce protocole en sortie et ayant des drivers pour QNX. Nous nous sommes orientés vers une interface USB/SPI en utilisant un microcontrôleur USB. Mais les temps d'attente entre un envoi et une réception nous ont surpris et ont rendu cette interface moins performante que prévu. La commande du robot telle que nous l'avons prévu était donc difficile. Toutefois, nous avons pu tester l'interface que nous avons choisie en appliquant une commande proportionnelle dérivée sur trois axes et les résultats que nous avons obtenus sont satisfaisant.

Passons maintenant à l'algorithme de commande. Commençons par la cinématique différentielle inverse avec évitement d'obstacles. Cet algorithme fonctionne correctement et ce, pour les différentes simulations que nous avons réalisées. Bien que la méthode de l'inverse généralisée soit assez lourde à implémenter, l'utilisation de maple pour les expressions des matrices permet de faciliter grandement le travail. Le point négligeable est sans doute le nombre important de paramètres à initialiser pour la fonction objective. Mais ce nombre de paramètres permet aussi d'être plus performant et de pouvoir traiter une plus grande variété d'évitements d'obstacles. Cet algorithme est indépendant du robot. Donc même si nous n'avons pas fait de tests sur le robot, les simulations et la visualisation des résultats, grâce au logiciel créé en Open GL, nous prouvent toute son efficacité.

La commande adaptative fonctionne aussi correctement lors des simulations. La réalisation pratique devrait fonctionner mais il est probable que nous devrons ajuster certains gains. Cette commande adaptative ne nous oblige pas à connaître les paramètres dynamiques du robot et ceci constitue un grand avantage par rapport aux commandes plus classiques tel qu'une commande proportionnelle intégrale dérivée.
Nous avons apporté un effort particulier à l'interface graphique sous QNX pour pouvoir réaliser des simulations ou des expériences dans les meilleures conditions. Enfin, un

programme en Open GL permet aussi de visualiser le robot après une simulation. Ce programme est indispensable car, en regardant des courbes de position des différents joints, il nous est presque impossible de savoir si le robot réalise correctement l'évitement d'obstacles.

CONCLUSION

Ce mémoire est le fruit d'un travail de plus d'un an et demi au sein du laboratoire de robotique du GREPCI de l'École de Technologie Supérieure. Il est composé tout d'abord d'une partie théorique comprenant l'évitement d'obstacles et la commande adaptative. Les informations recueillies dans la littérature, et celles que nous avons apportées, ont permis de développer un algorithme de commande adaptative avec évitement d'obstacle pour un robot se déplaçant dans l'espace. Les résultats de simulation sont très satisfaisants puisque l'erreur de suivi de trajectoire est de l'ordre du millimètre. Nous trouvons actuellement peu d'articles traitant de robots redondants évoluant dans l'espace et, dans ce sens, notre travail apporte sa principale contribution. La manière de modéliser les obstacles et l'introduction des pseudo distances entre le robot et les obstacles, nous permettent de travailler dans l'espace sans trop alourdir les calculs dans la cinématique différentielle inverse avec évitement d'obstacle. Enfin la commande adaptative basée sur la théorie de Lyapunov, nous permet de commander le robot sans connaître ses paramètres dynamiques. Là aussi, les résultats de simulations prouvent son bon fonctionnement.

La deuxième partie est l'implantation de cet algorithme en temps réel sous environnement QNX, afin de confirmer les résultats de simulation par des expériences sur le robot. Pour ce faire, nous avons dû d'abord traduire le code de Simulink en C pour qu'il puisse fonctionner sous QNX. Nous avons aussi réalisé une interface graphique afin de pouvoir lancer des simulations, des expériences, modifier les paramètres d'initialisation et visualiser les résultats. Nous avons réalisé les cartes de contrôle permettant de commander le robot, effectué des tests pour, notamment, associer une position physique aux impulsions envoyées par les encodeurs. Mais, nous avons une partie qui n'est pas optimale, l'interface entre le robot et l'ordinateur. Si ceci paraît étonnant, c'est tout simplement que le protocole SPI est relativement récent et est un protocole de communication entre microcontrôleurs et non un protocole de sortie

d'ordinateur. Nous verrons les solutions alternatives dans les recommandations pour pallier à ce problème.

Enfin, le travail réalisé, au niveau de la commande adaptative, de l'évitement d'obstacle, du hardware, des outils de visualisation graphique, nous a permis de construire une plateforme de commande de robot en temps réel quasi opérationnelle. Elle permettra de commander un robot d'une technologie performante dans une optique de modularité et de robustesse.

RECOMMANDATIONS

Nous allons bien sûr commencer nos recommandations par la manière de contourner le problème que nous connaissons avec l'interface. Ne trouvant pas sur le marché de carte d'entrée sortie PCI/SPI, nous pouvons nous orienter vers une carte d'entrée sortie digitale, possédant une vitesse de transfert compatible avec la vitesse de transfert du protocole SPI que l'on utilise. Il nous faudrait alors programmer le protocole SPI sur cette carte afin de communiquer selon ce protocole.

Pour ce qui est de la fonction objective permettant d'éviter les obstacles, le réglage des poids associés aux différents joints et aux obstacles est assez long, nous pourrions imaginer un moyen de régler tous ces poids automatiquement et en fonction de la trajectoire et de la position des obstacles.
Si ces poids peuvent varier automatiquement, nous pouvons alors imaginer un processus de détection d'obstacles afin de réaliser de l'évitement d'obstacles mobiles.

Enfin, nous travaillons actuellement avec des accéléromètres utilisant la technologie des MEMS. Ceux-ci vont nous permettre de mesurer l'accélération de chaque articulation du robot en temps réel. Pour réaliser la commande adaptative, nous avons besoin de cette accélération et nous l'obtenons actuellement en dérivant la vitesse, ce qui n'est pas optimum. Cette méthode de mesure devrait nous permettre de gagner de la précision pour la commande adaptative.

ANNEXE 1

Matrices de transformation homogènes

Voici les matrices de transformation homogènes pour les différents joints :

$$T01 := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & q1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T12 := \begin{bmatrix} \cos(q2) & -\sin(q2) & 0 & L1 \\ \sin(q2) & \cos(q2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

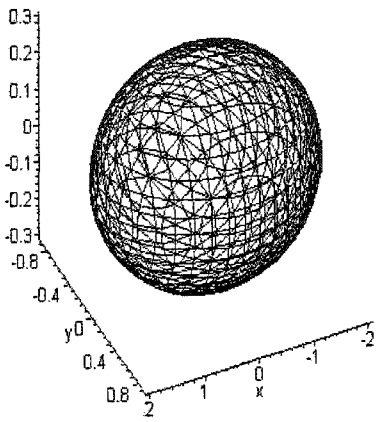
$$T23 := \begin{bmatrix} \cos(q3) & -\sin(q3) & 0 & L \\ \sin(q3) & \cos(q3) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T34 := \begin{bmatrix} \cos(q4) & -\sin(q4) & 0 & L \\ \sin(q4) & \cos(q4) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T45 := \begin{bmatrix} \cos(q5) & -\sin(q5) & 0 & L \\ \sin(q5) & \cos(q5) & 0 & 0 \\ 0 & 0 & 1 & L2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T56 := \begin{bmatrix} \cos(q6) & -\sin(q6) & 0 & L3 \\ 0 & 0 & -1 & 0 \\ \sin(q6) & \cos(q6) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

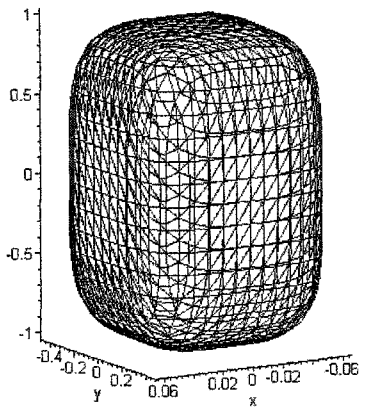
$$T67 := \begin{bmatrix} \cos(q7) & -\sin(q7) & 0 & 0 \\ 0 & 0 & 1 & -L4 \\ -\sin(q7) & -\cos(q7) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ANNEXE 2

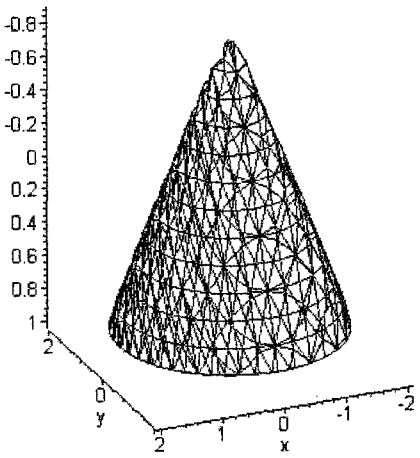
Fonctions de surface super quadratiques



$$S(x,y,z)=\left(\frac{x}{a}\right)^2+\left(\frac{y}{b}\right)^2+\left(\frac{z}{c}\right)^2-1.$$

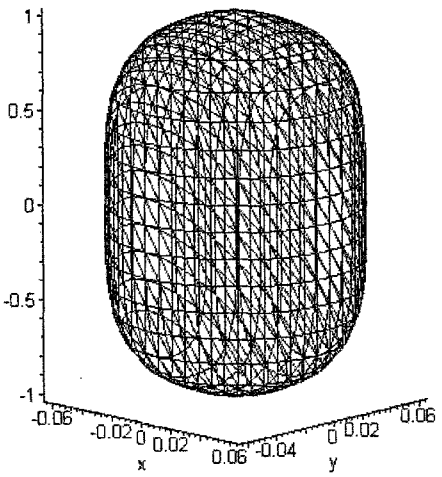


$$S(x,y,z)=\left(\frac{x}{a}\right)^{2n}+\left(\frac{y}{b}\right)^{2n}+\left(\frac{z}{c}\right)^{2n}-1.$$



$S(x,y,z)\mapsto$

$$\left(\frac{x}{mz+d}\right)^2+\left(\frac{y}{mz+d}\right)^2+\left(\frac{z}{c}\right)^{2n}-1=0.$$



$$S(x,y,z)=\left(\frac{x}{r}\right)^2+\left(\frac{y}{r}\right)^2+\left(\frac{z}{h}\right)^{2n}-1.$$

ANNEXE 3

Trajectoires désirées avec évitement d'obstacle, dans l'espace des articulations, appliquées à l'entrée du controleur adaptatif

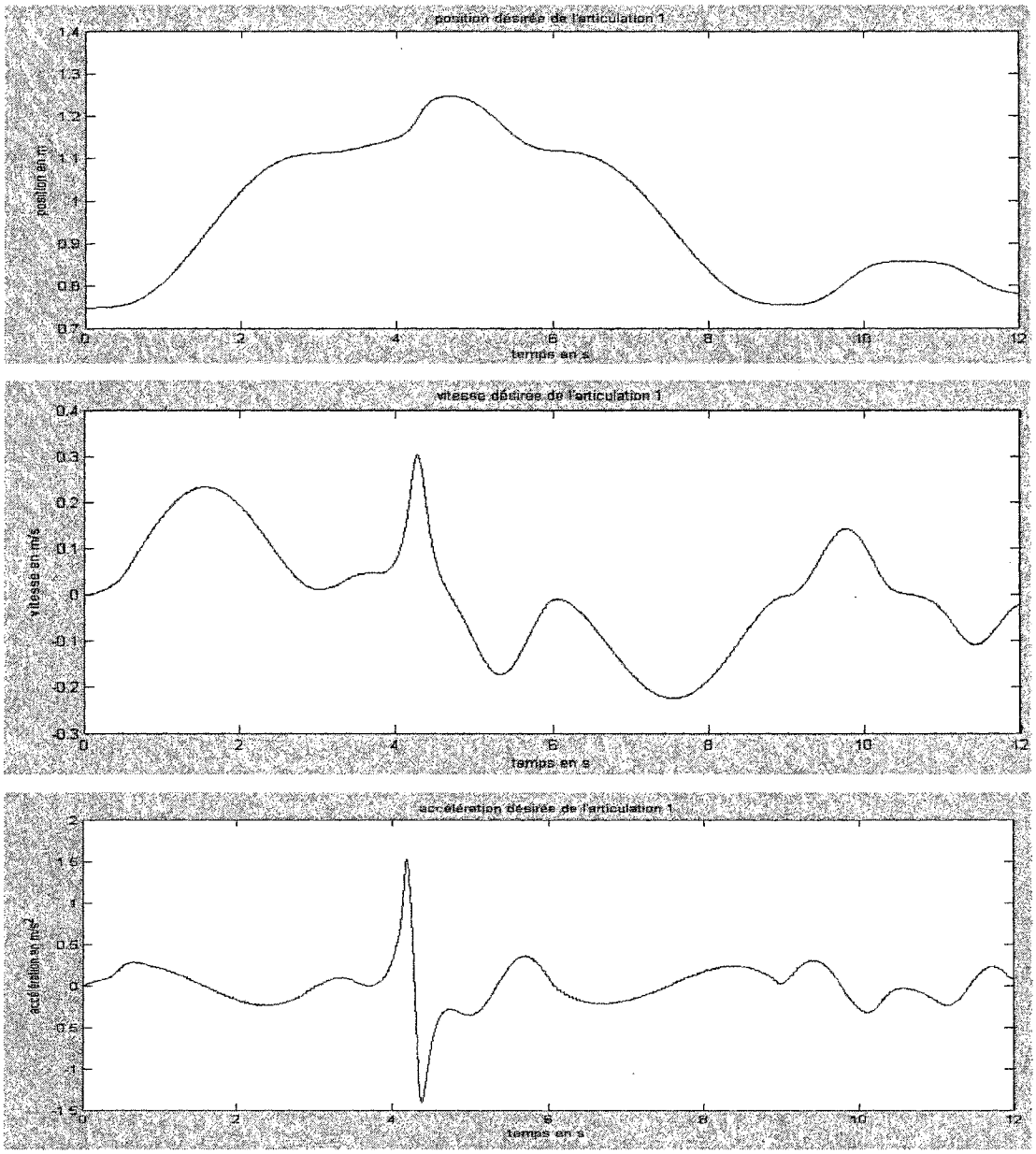


Figure 40 Trajectoire désirée de l'articulation 1

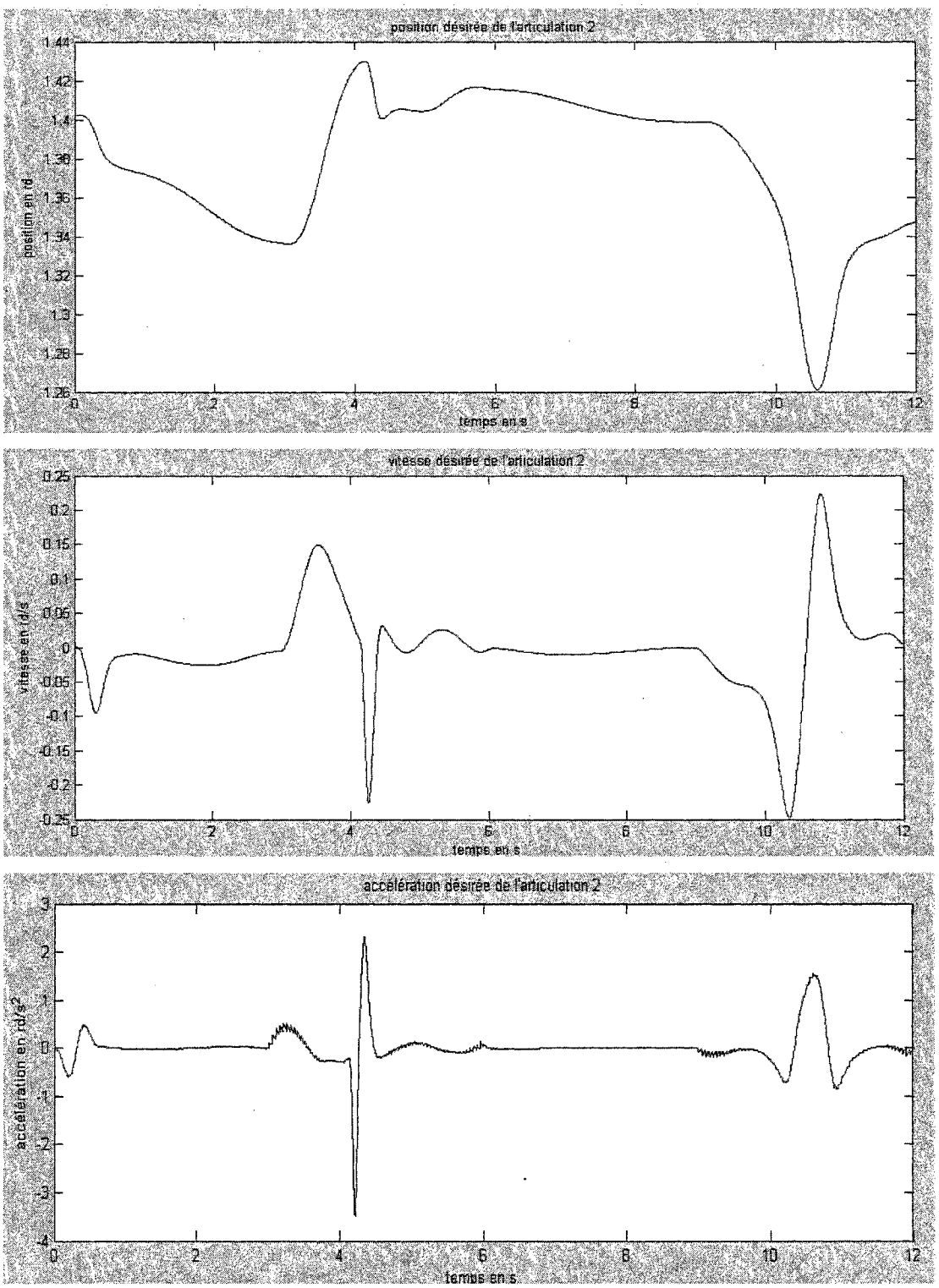


Figure 41 Trajectoire désirée de l'articulation 2

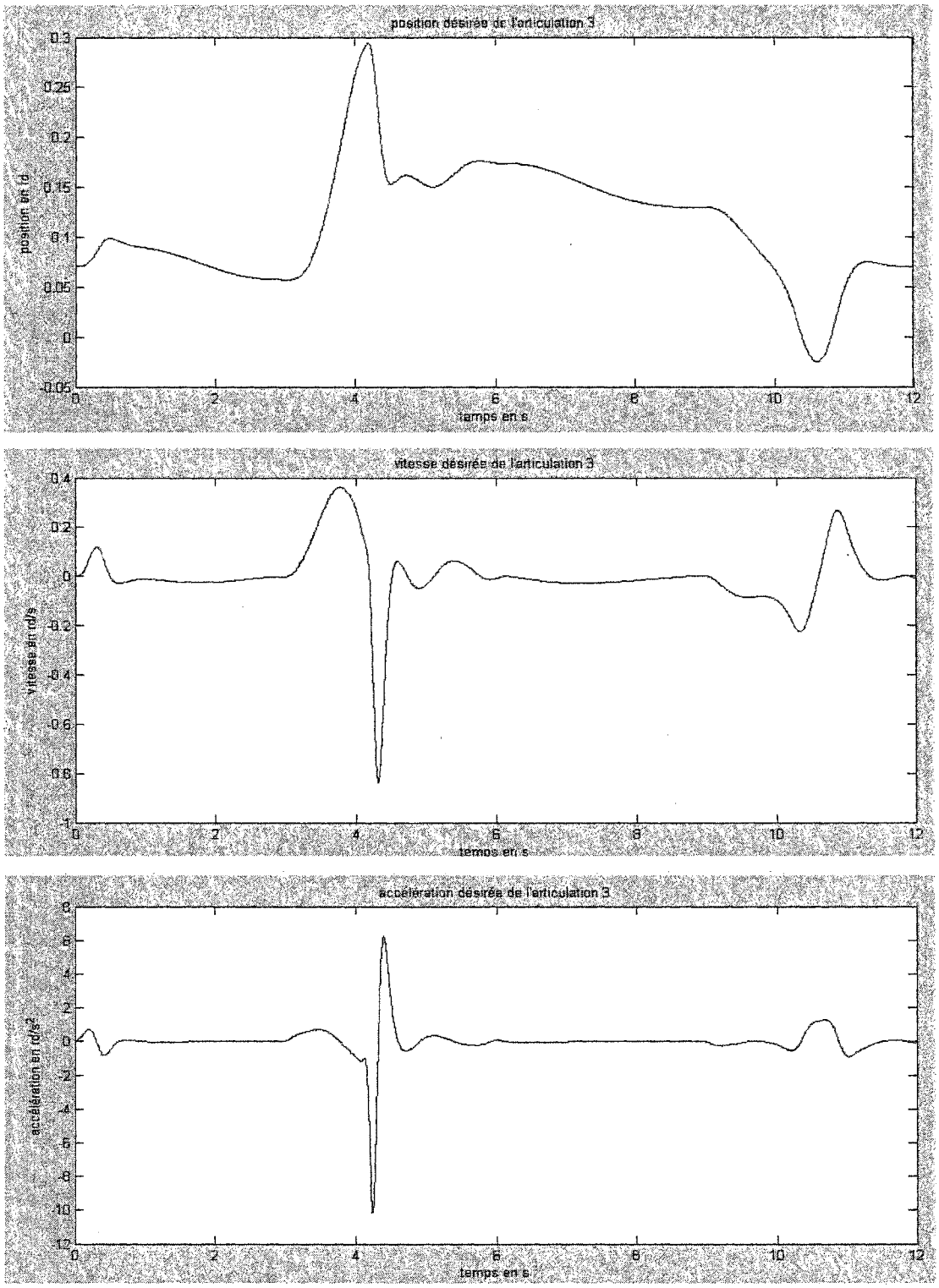


Figure 42 Trajectoire désirée de l'articulation 3

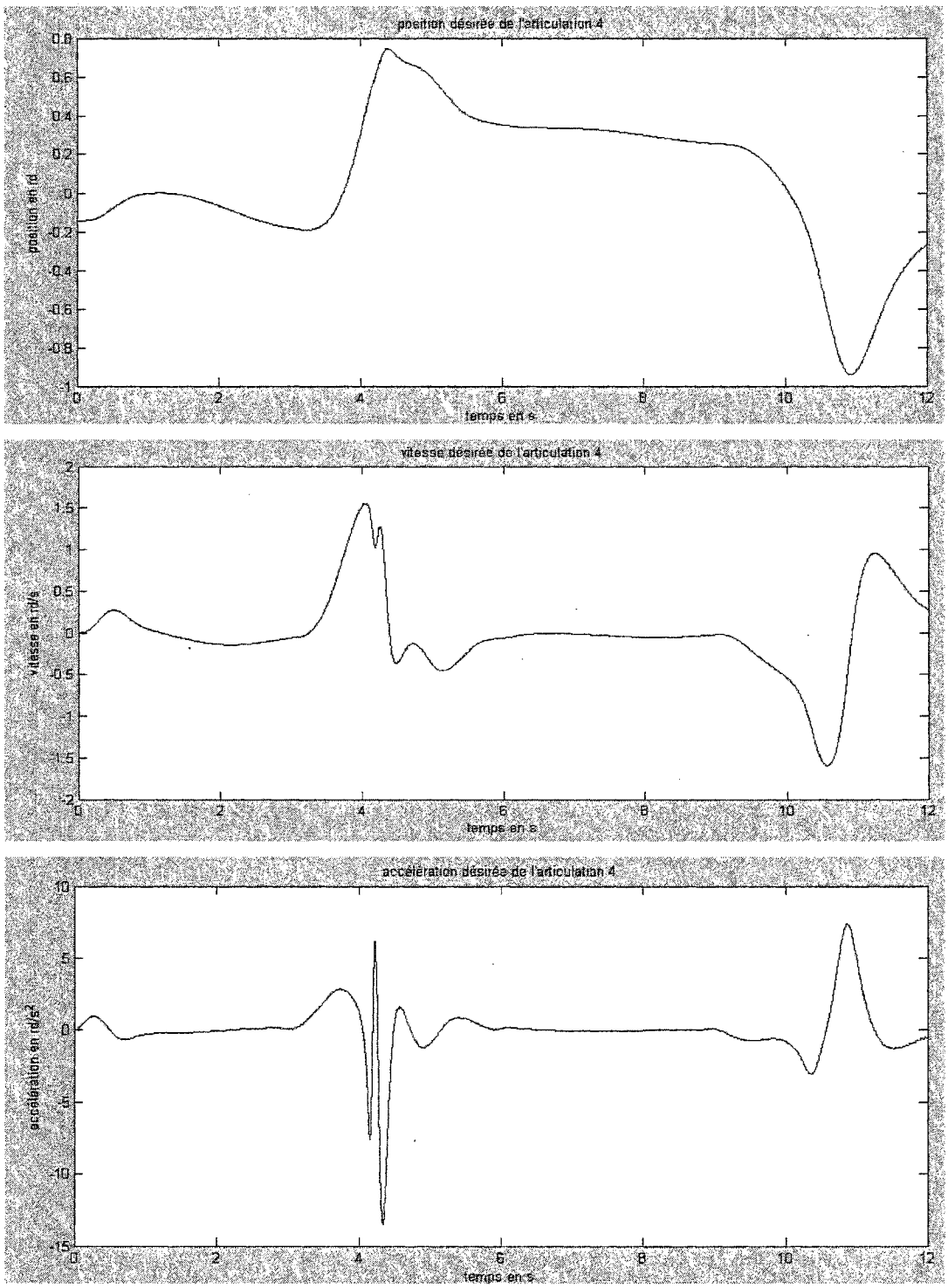


Figure 43 Trajectoire désirée de l'articulation 4

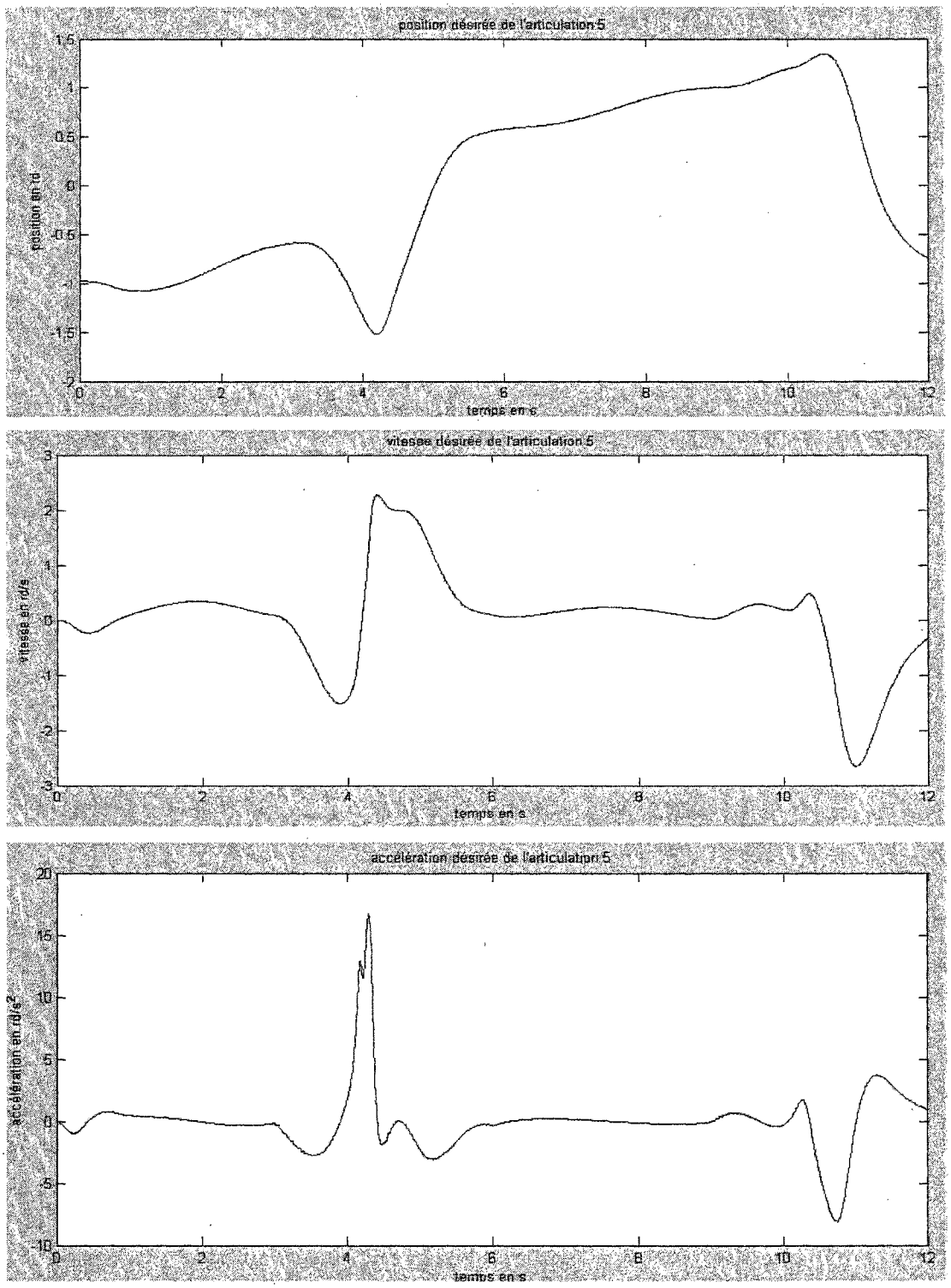


Figure 44 Trajectoire désirée de l'articulation 5

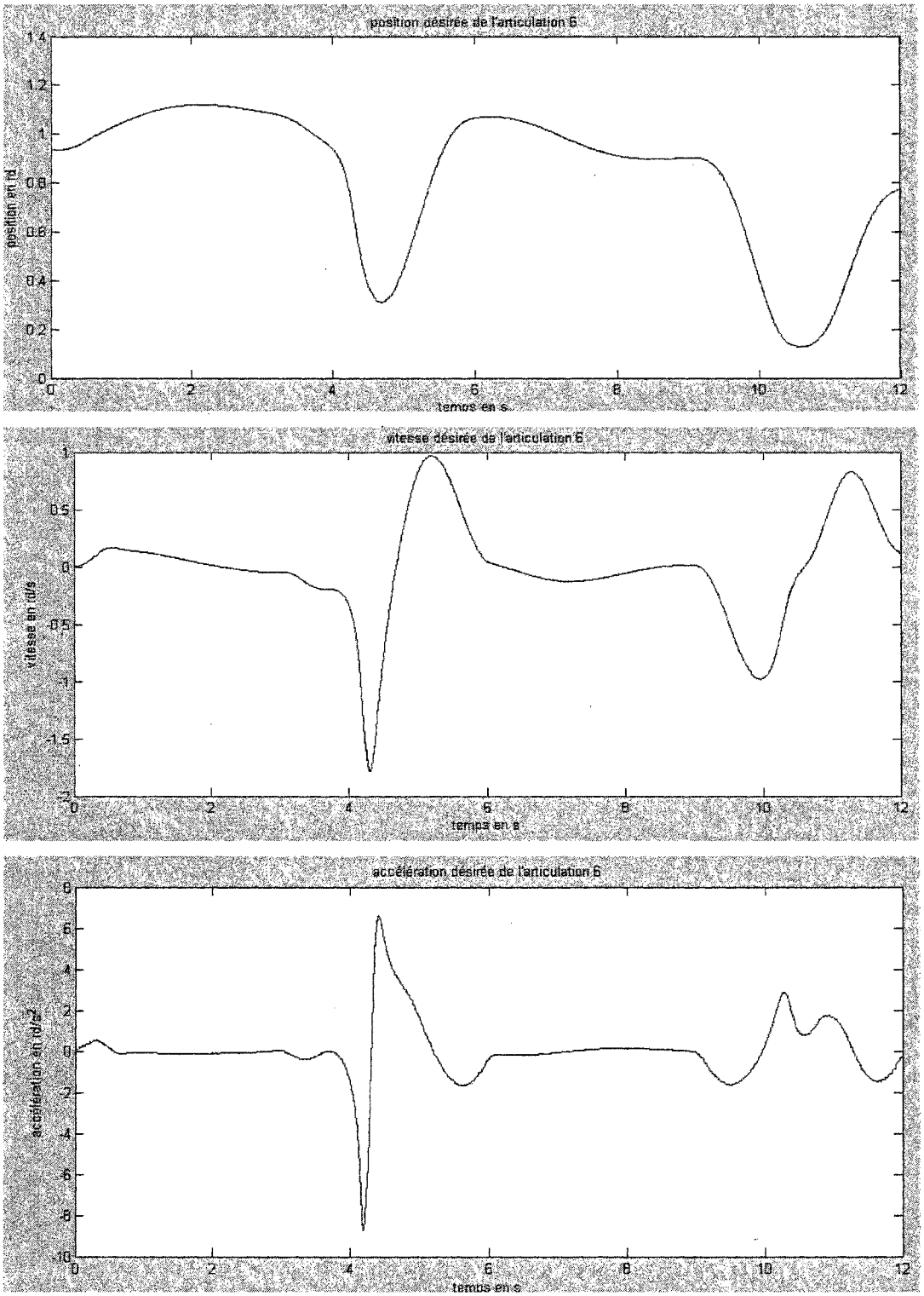


Figure 45 Trajectoire désirée de l'articulation 6

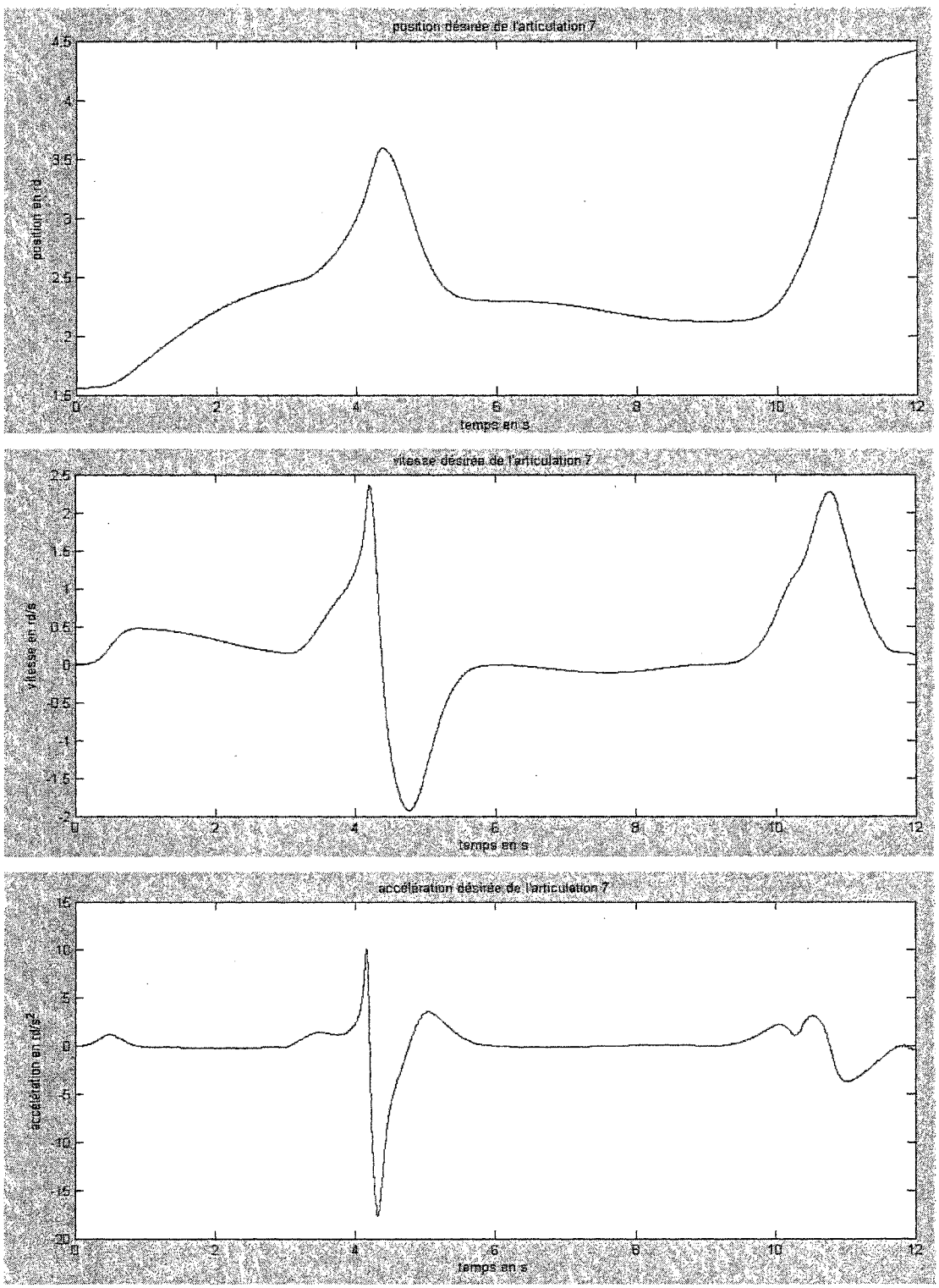


Figure 46 Trajectoire désirée de l'articulation 7

ANNEXE 4

Couples appliqués au robot, résultant de la commande adaptative

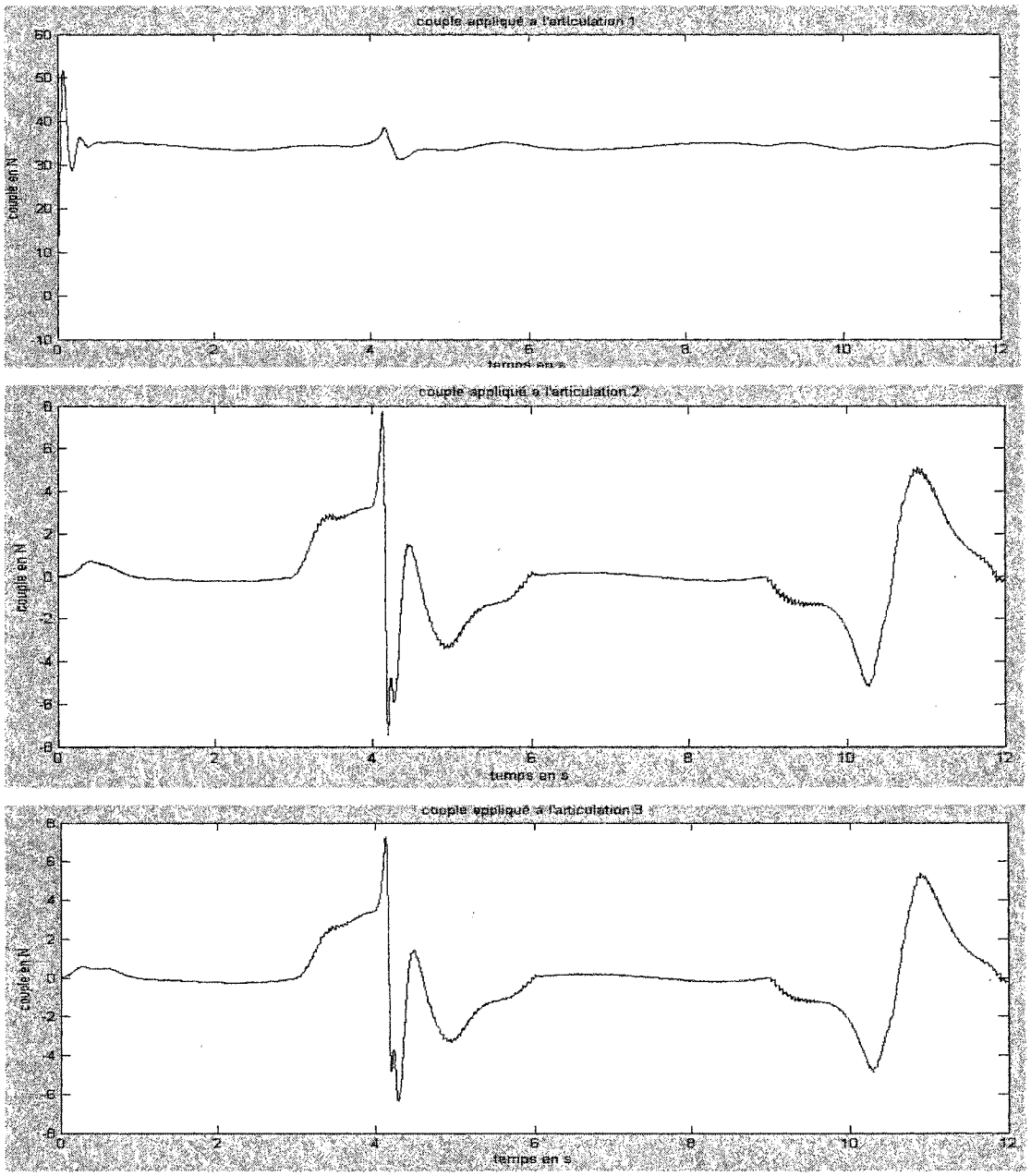


Figure 47 Couple appliqué au joints 1, 2 et 3

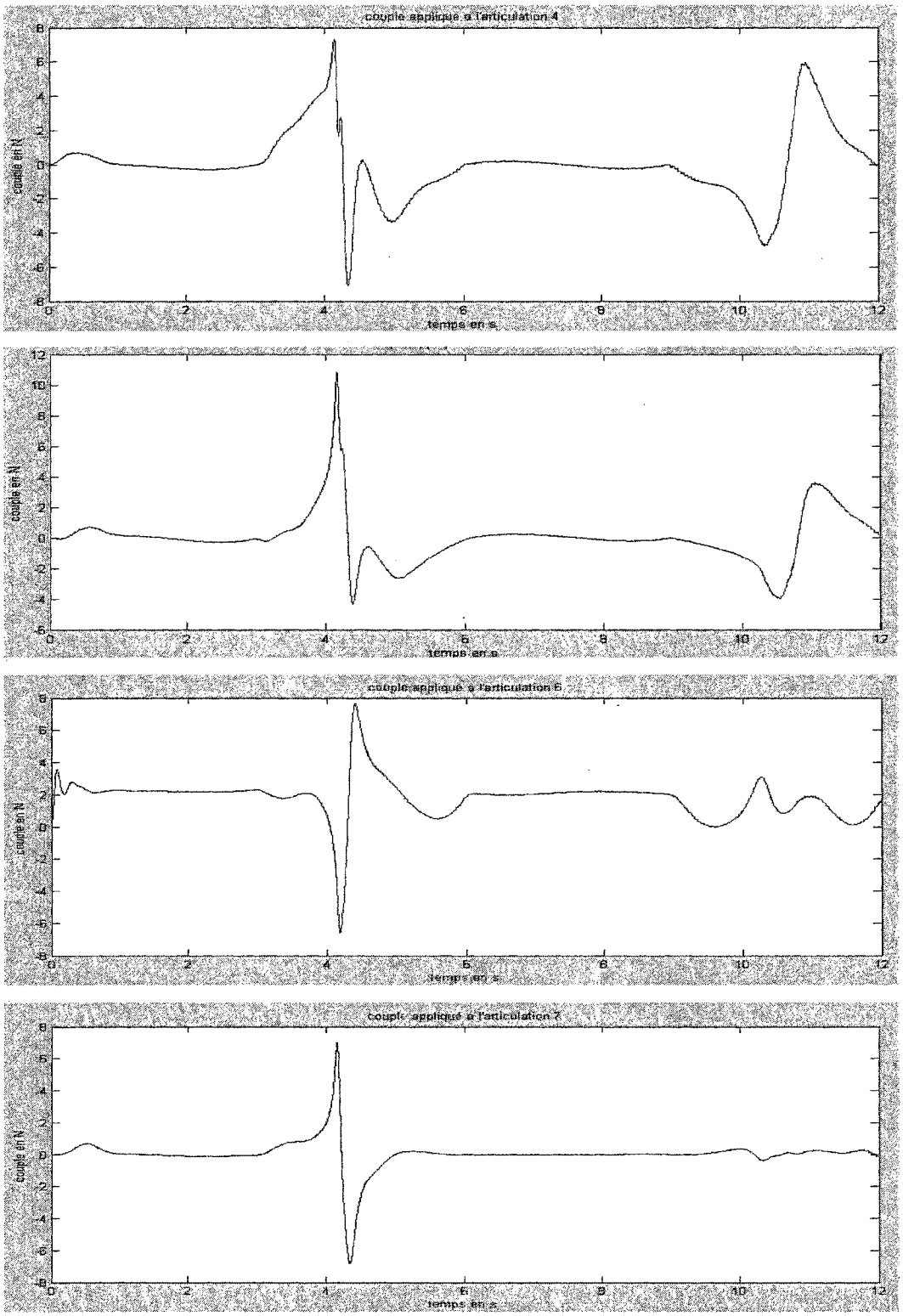


Figure 48 Couple appliqué au joints 4, 5, 6 et 7

BIBLIOGRAPHIE

[1] Zlajpah, L., Nemec, B., (2002). Kinematics control algorithms for on-line avoidance for redundant manipulators. Intelligent Robots and System IEEE/RSJ International Conference on 30 Sept.-5 Oct. 2002, 1898 -1903 vol.2.

[2] Wang, C.C., Kumar V., Chiu, G.M, (1998). A motion control and obstacle avoidance algorithm for hyper-redundant manipulators. Underwater Technology, proceedings of the 1998 International Symposium on, 15-17 April 1998, 466 - 471.

[3] Baillieul, J., (1985). Kinématic programming alternatives for redundant manipulators. Robotics and Automation. 1985 Proceedings. IEEE International Conference on, March 1985, 722 -728.

[4] Baillieul, J., (1986). Avoiding obstacles and resolving kinematic redundancy. Robotics and Automation. Proceedings. 1986 IEEE International Conference on, 3 April 1986, 1698 -1704.

[5] Ramdane-Cherif, A., Perdereau, V., Drouin, M., (1996). Penalty approach for a constrained optimization to solve on-line the kinematic problem of redundant manipulator. Robotics and Automation, 1996. Proceedings. IEEE International Conference on, Volume: 1, 22-28 April 1996. Page(s): 133 -138 vol.1.

[6] Perdereau, V., Passi, C., Drouin, M., (2002). Real-time control of redundant robotic manipulators for mobile obstacle avoidance. ELSEVIER, robotics and autonomous systems 41 (2002). Page(s): 41-59.

[7] Khoogar, A.R., Parker, J.K., (1991). Obstacle avoidance of redundant manipulators using genetic algorithms. Southeastcon '91, IEEE Proceedings of, 7-10 April 1991. Page(s): 317 -320 vol.

[8] Ramdane-Cherif, A., Meddah, D.Y., Perdereau, V.; Drouin, M., (1997) Inverse kinematic solution based on Lyapunov function for redundant and non-redundant robots. Computational Intelligence in Robotics and Automation. CIRA'97, Proceedings, 1997 IEEE International Symposium.

[9] Ahson, S.I., Sharkey, N.E., Nicolas, B., (1996). Avoiding joint limits and obstacles for kinematically redundant manipulators: a fuzzy-logic based approach. Fuzzy Systems, 1996, Proceedings of the Fifth IEEE International Conference on, 8-11 Sept. 1996, Page(s): 1777 -1781 vol.3

[10] Zlajpah, L., (1997). Control of redundant robots in presence of external forces Intelligent Engineering Systems, 1997. INES '97. Proceedings, 1997 IEEE International Conference on, 15-17 Sept. 1997. Page(s): 95 -100

[11] Khosla, P. and Volpe, R., (1988). Super quadric artificial potentials for obstacle avoidance and approach. Robotics and Automation, 1988. Proceedings, 1988 IEEE International Conference on, 24-29 April 1988. Page(s): 1778 -1784 vol.3.

[12] Bigras, P. (2003). *SYS-866 Modélisation et commande des robots en contact*, notes de cours, École de Technologie Supérieure, Département de génie de la production automatisée, hiver 2003.

[13] Naumov, B. N. (1990). *Philosophy of nonlinear control systems* . Collection: Advances in science and technology in the USSR. Boca Raton, Flor. : CRC Press.

[14] MOHLER, R. R. (1991). *Nonlinear systems*. Englewood Cliffs, N.J. Prentice-Hall.

[15] Lamnabhi-Lagarrigue, F., Rouchon, P. (2003). *Commandes non linéaires* . Paris : Hermès Science Publications.

[16] Lozano, R., Taoutaou, D. (2001). *Commande adaptative et applications*. Paris: Hermès Science Publications.

[17] Borne, P., (1990). *Commande et optimisation des processus*. Paris : Technip.

[18] Nganga-Kouya, D., Saad, M., Lamarche, L., (2002). Backstepping adaptive hybrid force/position control for robotic manipulators . Proceedings of the American control conference. Anchorage AK May 2002. Pages(s): 4595 – 4600.

Sites web:

[19] http://www.usb.org (page consultée le 22 avril 2004)

[20] http://www.roboticsdesign.qc.ca/Main.htm (page consultée le 4 mars 2003)