ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE EN GÉNIE DE LA PRODUCTION AUTOMATISÉE
M. Ing.

PAR
HAMELIN, Philippe

SIMULATION AVEC MATÉRIEL DANS LA BOUCLE D'UN ROBOT EN INTERACTION AVEC UN ENVIRONNEMENT RÉEL

MONTRÉAL, LE 25 AOÛT 2008

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

M. Pascal Bigras, directeur de mémoire
Département du génie de la production automatisée à l'École de technologie supérieure

M. Guy Gauthier, président du jury
Département du génie de la production automatisée à l'École de technologie supérieure

M. Pierre-Luc Richard, membre du jury
Institut de recherche d'Hydro-Québec

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

LE 25 JUILLET 2008

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Ce mémoire est l'aboutissement de deux années de recherche. Celles-ci m'ont permis d'acquérir des connaissances inestimables dans le domaine de la science qui me passionne le plus, soit la robotique. La réalisation d'un tel projet n'aurait pu se faire si efficacement sans la contribution des organismes et individus qui ont été impliqués de proche ou de loin dans ce projet.

Je tiens tout d'abord à remercier mon directeur de recherche, le professeur Pascal Bigras de l'École de technologie supérieure. Il a mis à ma disposition son bagage de connaissances et il m'a offert une disponibilité exemplaire tout au long de mes travaux.

Ensuite, je remercie toute ma famille et plus particulièrement ma conjointe Manon, pour avoir cru en moi et m'avoir encouragé à poursuivre mes études aux cycles supérieurs.

Je remercie aussi Michel Blain et Julien Beaudry de l'Institut de recherche d'Hydro-Québec pour m'avoir offert l'opportunité d'effectuer mes travaux de recherche à l'unité Automatisation et systèmes de mesure. En mettant à ma disposition leurs équipements et des ressources humaines, ils m'ont permis de réaliser un projet de recherche appliqué en solutionnant une problématique industrielle.

Finalement, je tiens à remercier sincèrement l'École de technologie supérieure et le Conseil de recherches en sciences naturelles et en génie du Canada pour avoir supporté financièrement ce projet, sans quoi je n'aurais pu le réaliser.

SIMULATION AVEC MATÉRIEL DANS LA BOUCLE D'UN ROBOT EN INTERACTION AVEC UN ENVIRONNEMENT RÉEL

HAMELIN, Philippe

RÉSUMÉ

Le développement de systèmes robotisés nécessite généralement plusieurs itérations, qui peuvent être basées sur une approche par simulation, par réalisation d'un prototype physique ou une combinaison des deux. D'une part, le développement à l'aide de la simulation doit tenir compte non seulement de la modélisation du robot, mais également de celle de son environnement. Malheureusement, le modèle de l'environnement n'est pas toujours bien connu et peut être imprécis. D'autre part, le développement basé sur la fabrication d'un prototype physique, permet de palier aux problèmes de mauvaise connaissance et d'imprécision du modèle de l'environnement, mais au détriment d'un coût et d'un temps de prototypage élevés. La solution hybride qu'est la simulation avec matériel dans la boucle permet de remplacer, dans la simulation, seulement le sous-système dont le modèle n'est pas bien connu par un prototype physique. Cette solution est un compromis et permet de ne pas sacrifier la précision tout en réduisant les coûts et le temps de développement.

Ce mémoire propose d'utiliser l'approche de la simulation avec matériel dans la boucle dans le but de faciliter le développement futur d'un robot de meulage sous-marin à l'Institut de Recherche d'Hydro-Québec (IREQ). L'objectif est donc de comparer et de valider deux schémas de simulation d'un robot virtuel en interaction avec un environnement réel. Pour ce faire, un robot réel est utilisé pour reproduire la dynamique du robot simulé. Ceci permet au robot simulé d'interagir virtuellement avec l'environnement réel. Une contrainte rhéonomique est établie afin d'en déduire une loi de commande qui assure la synchronisation des mouvements de l'outil du robot réel avec ceux de l'outil du robot virtuel. Puis, les deux schémas sont étudiés pour la simulation du robot virtuel : le premier avec et le deuxième sans application de la contrainte rhéonomique. L'étude expérimentale présentée dans ce mémoire permet notamment de comparer ces deux schémas pour la simulation d'une loi de commande d'impédance. Pour faciliter la preuve de ce concept, un environnement simple et bien connu est utilisé pour comparer les résultats de simulation avec matériel dans la boucle avec ceux d'une simulation pure.

Ce mémoire présente également le développement d'une procédure de prototypage rapide de la simulation avec matériel dans la boucle. Cette méthode permet de générer automatiquement toutes les équations et les programmes requis et ce, uniquement à partir des paramètres cinématiques et dynamiques des robots. Ainsi, en connaissant les paramètres d'un robot virtuel, il est possible en quelques minutes de connaître son comportement face à un environnement réel.

Les résultats expérimentaux démontrent que la simulation avec matériel dans la boucle donne des résultats fiables. Toutefois, la friction de coulomb présente sur les actionneurs linéaires perturbe visiblement la simulation. Elle se traduit respectivement par un effet d'oscillation de

type cycle-limite et une erreur en régime permanent pour le schéma avec et sans application de la contrainte. Une action intégrale est ajoutée à la loi de commande afin de compenser cette perturbation dans le schéma sans application de la contrainte. Il est démontré expérimentalement que cet ajout améliore considérablement les performances de ce schéma de simulation, ce qui en fait à priori la solution préférée, en considérant sa simplicité d'implantation à comparer le schéma avec application de la contrainte.

HARDWARE-IN-THE-LOOP SIMULATION OF ROBOTS INTERACTING WITH A REAL ENVIRONMENT

HAMELIN, Philippe

ABSTRACT

The development of automated systems generally requires several iterations. These may be based on a simulation approach, the manufacture of a physical prototype, or a combination of the two. Development using simulation must take into account not only the modeling of the robot, but also that of its environment. Unfortunately, modeling the environment can be a challenge in terms of the information known about it, and so the result may be imprecise. Development based on the manufacture of a physical prototype can counter the problem of insufficient information, but this is an expensive and time-consuming option. The hybrid solution, which is known as hardware-in-the-loop simulation, involves replacing in the simulation only the subsystem whose model is not well known in the form of a physical prototype. Although it is a compromise, accuracy is not sacrificed with this solution, and both costs and development time are reduced.

This Master's degree thesis proposes using the hardware-in-the-loop approach to facilitate the future development of an underwater grinding robot for Hydro Quebec's Research Institute (IREQ). The objective is to compare and validate two simulation schemes involving a virtual robot interacting with a real environment. To achieve this, a real robot is used to reproduce the dynamics of the simulated robot, allowing the simulated robot to interact virtually with the real environment. A rheonomic constraint is applied to derive a control law which provides synchronization of the movements of the real robot tool with those of the virtual robot tool. Then, both schemes are studied to build the simulation of the virtual robot: the first with the application of the rheonomic constraint, and the second without. With the experimental study presented here, it is possible to compare these two schemes for the simulation of an impedance control law. To ease the proof of concept, a simple and well-known environment is used to compare the hardware-in-the-loop simulation results with those of the pure simulation.

Also proposed in this thesis is a procedure for the rapid prototyping of a hardware-in-the-loop simulation. This method makes it possible to automatically generate all the equations and programs required for our simulation based only on the kinematic and dynamic parameters of the robots. Thus, by knowing the parameters of a virtual robot, it is possible to learn, in just a few minutes, how it behaves as it interacts with a real environment.

The experimental results show that the hardware-in-the-loop simulation gives reliable results. However, the Coulomb friction of the real robot apparently degrades the simulation. This is reflected in a cycle-limit oscillation and a steady-state error for the scheme with and without application of the constraint. An integral term is added to the real robot control law to compensate for this disturbance in the scheme without application of the constraint. It is demonstrated experimentally that this addition greatly improves the performance of the

simulation method. This makes it the preferred option, considering its ease of implementation compared to the scheme involving application of the constraint.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

IREQ	Institut de Recherche d'Hydro-Québec
MICROB	Modules intégrés pour le contrôle de robots
ÉTS	École de Technologie Supérieure
HIL	« Hardware-In-the-Loop »
ODE	« Ordinary Differential Equations »
DAE	« Differential Algebraic Equations »
CLIK	« Closed-Loop Inverse Kinematics »

LISTE DES SYMBOLES ET UNITÉS DE MESURE

m	Nombre de degrés de liberté du robot virtuel
n	Nombre de degrés de liberté du robot réel
Δe_r	Distance entre la base du robot réel et l'environnement
m_{1r}	Masse de la membrure 1 du robot réel
m_{2r}	Masse de la membrure 2 du robot réel
q_{1r}	Coordonnée généralisée de l'articulation 1 du robot réel
q_{2r}	Coordonnée généralisée de l'articulation 2 du robot réel
$\mathbf{q}_r$	Vecteur des coordonnées généralisées du robot réel
k_E	Constante de rigidité de l'environnement
b_E	Constante d'amortissement de l'environnement
y_r	Position cartésienne du robot réel selon l'axe Y du repère de l'environnement
z_r	Position cartésienne du robot réel selon l'axe Z du repère de l'environnement
$\mathbf{p}_r$	Pose cartésienne du robot réel dans le repère de l'environnement
$\mathbf{J}_r$	Matrice jacobienne du robot réel
$\mathbf{M}_r$	Matrice de masse du robot réel
$\mathbf{F}_r$	Vecteur des forces de Coriolis et centrifuges du robot réel
f_y	Force de l'environnement selon l'axe Y du repère de l'environnement
f_z	Force de l'environnement selon l'axe Z du repère de l'environnement
$\mathbf{f}_c$	Vecteur des forces de l'environnement
f_c	Force de contact avec l'environnement
$\boldsymbol{\tau}_r$	Vecteur des forces généralisées du robot réel
Δe_s	Distance entre la base du robot virtuel et l'environnement
l_1	Longueur de la membrure 1 du robot virtuel
l_2	Longueur de la membrure 2 du robot virtuel
m_{1s}	Masse de la membrure 1 du robot virtuel
m_{2s}	Masse de la membrure 2 du robot virtuel
q_{1s}	Coordonnée généralisée de l'articulation 1 du robot virtuel
q_{2s}	Coordonnée généralisée de l'articulation 2 du robot virtuel
$\mathbf{q}_s$	Vecteur des coordonnées généralisées du robot virtuel
s_{1s}	Défini comme $s_{1s} = \sin(q_{1s})$
c_{1s}	Défini comme $c_{1s} = \cos(q_{1s})$
s_{1s2s}	Défini comme $s_{1s2s} = \sin(q_{1s} + q_{2s})$
c_{1s2s}	Défini comme $c_{1s2s} = \cos(q_{1s} + q_{2s})$

y_s	Position cartésienne du robot virtuel selon l'axe Y du repère de l'environnement
z_s	Position cartésienne du robot virtuel selon l'axe Z du repère de l'environnement
$\mathbf{p}_s$	Pose cartésienne du robot virtuel dans le repère de l'environnement
$\mathbf{J}_s$	Matrice jacobienne du robot virtuel
$\mathbf{M}_s$	Matrice de masse du robot virtuel
$\mathbf{F}_s$	Vecteur des forces de Coriolis et centrifuges du robot virtuel
T	Énergie cinétique totale du robot virtuel
V	Énergie potentielle totale du robot virtuel
$\mathbf{\Phi}$	Contrainte rhéonomique de pose entre les deux robots
$\mathbf{\Phi}_{\mathbf{q}_s}$	Matrice jacobienne de la contrainte rhéonomique
$\boldsymbol{\lambda}$	Multiplicateur de Lagrange
$\mathbf{p}_d, \dot{\mathbf{p}}_d, \ddot{\mathbf{p}}_d$	Position, vitesse et accélération désirées (trajectoire)
$\mathbf{p}_c, \dot{\mathbf{p}}_c, \ddot{\mathbf{p}}_c$	Position, vitesse et accélération corrigées avec la loi d'impédance
$\mathbf{K}_{p_s}$	Gain proportionnel du contrôleur du robot virtuel
$\mathbf{K}_{d_s}$	Gain dérivé du contrôleur du robot virtuel
$\mathbf{u}_p$	Commande provenant de la partie linéaire du contrôleur du robot virtuel
$\mathbf{u}$	Commande provenant de la linéarisation dans l'espace de la tâche du contrôleur du robot virtuel
$\mathbf{\Delta}_{dc}$	Différence entre la pose désirée et la pose corrigée de la loi d'impédance du robot virtuel
$\mathbf{M}_c$	Matrice des masses de la loi d'impédance du robot virtuel
$\mathbf{B}_c$	Matrice des constantes d'amortissements de la loi d'impédance du robot virtuel
$\mathbf{K}_c$	Matrice des constantes de rigidités de la loi d'impédance du robot virtuel
$\mathbf{e}_s$	Erreur de position du contrôleur du robot virtuel
$\tilde{\ddot{\mathbf{p}}}_s$	Estimation de l'accélération de la pose du robot virtuel calculée à partir du modèle non contraint
$\mathbf{K}_{p_r}$	Gain proportionnel du contrôleur du robot réel
$\mathbf{K}_{d_r}$	Gain dérivé du contrôleur du robot réel
$\mathbf{K}_{i_r}$	Gain intégral du contrôleur du robot réel
$\mathbf{e}_r$	Erreur de position du contrôleur du robot virtuel
$\boldsymbol{\xi}$	Vecteur des états indépendants du robot virtuel
$\mathbf{P}$	Matrice de projection dont les colonnes annulent la contrainte

INTRODUCTION

Hydro-Québec opère plusieurs centaines de digues et barrages à travers la province de Québec et plusieurs d'entre eux ont été mis en service au milieu du siècle dernier. Afin d'assurer la pérennité et la sécurité de ces installations, des travaux de réfection des structures métalliques doivent être effectués périodiquement. Actuellement, ces travaux sont réalisés par des ouvriers spécialisés, ce qui nécessite que la section du barrage ciblée soit sécurisée. Afin de réduire le coût des travaux, le temps d'exécution et pour améliorer la sécurité des travailleurs, l'Institut de recherche d'Hydro-Québec (IREQ) a initié un projet de développement d'un système robotisé qui permettra d'effectuer la réfection de ces structures métalliques, notamment par un procédé de meulage.

La réalisation d'un tel système nécessite habituellement le développement de plusieurs prototypes avant d'en arriver à une solution optimale. Dans le processus de conception de ces prototypes, deux méthodes se distinguent typiquement. La première méthode consiste à simuler le système entier. Cette simulation est dite idéale et nécessite le développement des modèles de tous les sous-systèmes, incluant le robot et l'environnement avec lequel il interagit. Dans des tâches de réfection comme le meulage, le modèle d'interaction du robot avec l'environnement est très complexe et peut mener à des résultats de simulation imprécis. La seconde méthode, à l'opposé, consiste à fabriquer un prototype physique afin d'étudier son comportement réel. Des résultats exacts sont ainsi obtenus, mais au détriment d'un coût et d'une durée de développement élevés.

L'objectif de ce mémoire est donc d'établir une architecture de prototypage qui permettra de simuler un système robotisé sans avoir à modéliser les forces d'interaction avec l'environnement. Pour ce faire, les deux méthodes, décrites dans le paragraphe précédent, sont combinées pour devenir une approche plus flexible nommée, simulation avec matériel dans la boucle (« hardware-in-the-loop », HIL). Cette solution hybride se caractérise par l'utilisation de parties réelles (matériel) à l'intérieur de la boucle de simulation.

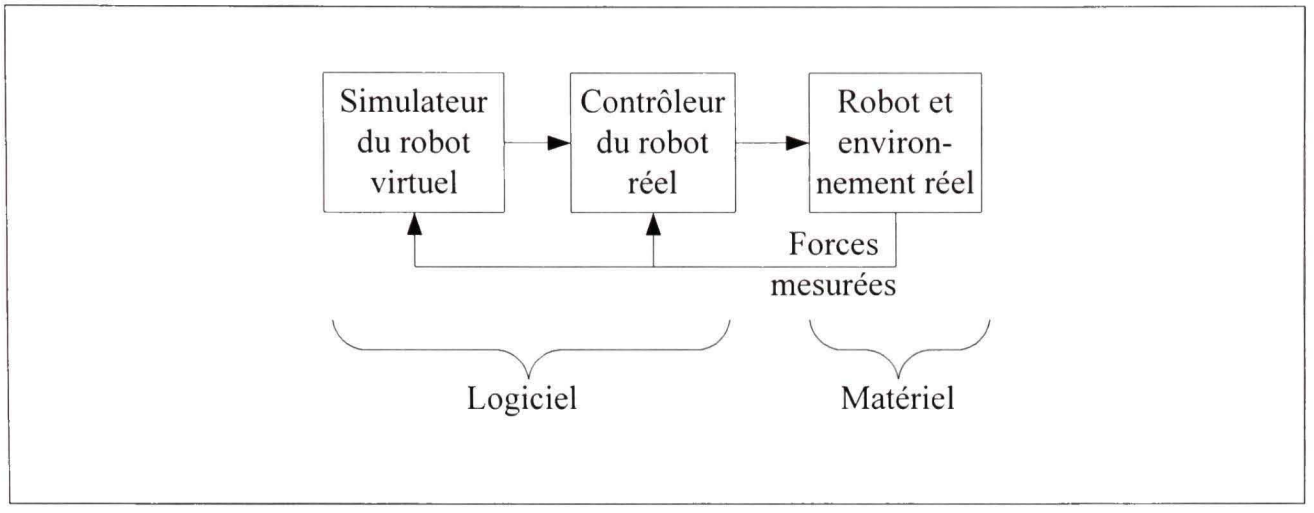

Schéma de principe de la simulation avec matériel dans la boucle.

Pour répondre au cas particulier de la problématique de ce projet, l'environnement virtuel doit être remplacé par un environnement réel (voir figure ci-haut). Toutefois, cet environnement réel à lui seul n'est pas suffisant pour effectuer la simulation HIL. En effet, un robot réel doit être ajouté afin d'interagir avec ce dernier. La dynamique et la configuration du système robotisé (robot virtuel) étant différentes de celles du robot réel, les forces mesurées de l'interaction du robot réel avec l'environnement peuvent être utilisées par le robot virtuel uniquement sous certaines conditions :

- Le robot réel doit reproduire la dynamique du robot virtuel;
- Les deux robots doivent être soumis aux mêmes forces d'interaction avec l'environnement.

Ces deux conditions forment ainsi la base pour la conception de la loi de commande du robot réel, qui est l'un des objectifs de ce mémoire. Pour ce faire, une loi de commande basée sur la linéarisation exacte dans l'espace de la tâche est utilisée. La conception doit débuter par l'imposition d'une contrainte entre le robot réel et le robot virtuel au niveau des accélérations dans l'espace de la tâche. Ensuite, une commande linéaire proportionnel-dérivée est ajoutée à la loi de commande ainsi obtenue. Dans ce mémoire, l'ajout d'une action intégrale est

également proposée afin de limiter les erreurs de suivi de trajectoire. En appliquant cette loi de commande au robot réel, les forces mesurées de l'interaction du robot réel avec l'environnement réel pourront être utilisées par le robot virtuel. La simulation HIL permettra donc de simuler le robot virtuel sans avoir à modéliser ces forces, qui peuvent être très complexes, notamment dans le cas du procédé de meulage.

Pour cette étude, deux schémas de simulation sont évalués. La première méthode consiste à appliquer la contrainte entre le robot virtuel et le robot réel. C'est-à-dire qu'à chaque itération la pose du robot virtuel est synchronisée avec celle du robot réel. Cette méthode fait appel au modèle dynamique contraint du robot virtuel et nécessite la simulation d'un système d'équations différentielles-algébriques (DAE). À l'opposé, la seconde méthode n'applique pas explicitement en temps réel la contrainte entre les deux robots. Elle propose plutôt d'intégrer le modèle dynamique non-contraint du robot virtuel, ce qui revient essentiellement à simuler un système d'équations différentielles ordinaires (ODE).

Afin de vérifier expérimentalement la validité de la méthode HIL proposée pour simuler un système robotisé en interaction avec un environnement réel, la simulation idéale est utilisée comme base de comparaison. Or, comme il a été mentionné précédemment, ceci requiert la modélisation de tous les sous-systèmes, notamment la dynamique du robot et de l'environnement. Donc, puisque la problématique initiale est justement de ne pas avoir à modéliser l'environnement, un cas simple où le modèle peut être facilement identifié sera utilisé comme preuve de concept. L'environnement choisi est composé d'une plaque d'acier coulée dans un bloc de béton. Ainsi, la loi de commande du robot réel et ses variantes pourront être validées et ce, pour les deux schémas de simulation proposés.

Pour cette preuve de concept, un robot virtuel à deux joints rotatifs, commandé par une loi d'impédance, sera simulé. Le robot réel utilisé pour les expérimentations possèdent également deux joints, mais ceux-ci sont linéaires à entraînement direct. Le robot réel devra donc reproduire la dynamique du robot virtuel même s'ils sont topologiquement différents.

La contribution de ce mémoire consiste en une étude expérimentale de la simulation HIL dans un contexte, à notre connaissance, inexploré jusqu'à maintenant. Ce contexte est caractérisé par l'utilisation d'une commande d'impédance pour le robot virtuel et d'un robot réel possédant des moteurs à entraînement direct. De plus, une nouvelle variante du contrôleur du robot réel est proposée afin d'améliorer les performances de la simulation HIL sans application de la contrainte. Finalement, une contribution indirecte, mais significative de ce mémoire, est le développement d'une procédure de prototypage rapide. À partir des paramètres cinématiques et dynamiques des robots, elle permet de générer automatiquement les équations symboliques nécessaires à la simulation HIL, le code en langage C ainsi que sa compilation. Ainsi le contrôleur du robot réel peut s'exécuter sur le système d'exploitation temps-réel QNX. Les résultats expérimentaux présentés dans ce mémoire ont fait l'objet de deux articles de conférence (Hamelin et al., 2008a; Hamelin et al., 2008b) qui seront présentés prochainement et un article de journal qui sera bientôt soumis à un journal avec comité de lecture.

Le présent mémoire est composé de 5 chapitres qui s'organisent comme suit :

- Le chapitre 1 présente la revue de la littérature portant sur la problématique du prototypage virtuel, ainsi que sur les différentes lois de commande HIL.
- Le chapitre 2 décrit en détails la modélisation du robot virtuel, du robot réel et de l'environnement. Plus particulièrement, pour chacun des robots, la cinématique directe, la cinématique inverse, la cinématique différentielle directe, la cinématique différentielle inverse et le modèle dynamique sont présentés.
- Le chapitre 3 présente la simulation idéale, soit la base de comparaison qui servira à vérifier la validité de la simulation HIL. Le contrôleur d'impédance du robot virtuel est décrit et l'étude de stabilité du système en boucle fermé est présentée. Aussi, la méthode de sélection des paramètres de l'impédance désirée est expliquée.
- Le chapitre 4 présente la simulation avec matériel dans la boucle, qui est le cœur de ce mémoire. Le contrôleur du robot réel et ses variantes y sont développés. Ensuite,

les deux schémas de simulation y sont expliqués pour le cas à l'étude, mais aussi pour le cas général.

- Le chapitre 5 décrit le banc d'essai utilisé pour les expérimentations et présente la procédure de prototypage rapide. L'identification de l'environnement et du robot réel est ensuite effectuée. Puis, la sélection des paramètres de l'impédance désirée et le calcul des gains des contrôleurs sont présentés. Finalement, les résultats expérimentaux sont montrés et discutés pour chacun des cas à l'étude.

CHAPITRE 1

REVUE DE LA LITTÉRATURE

Dans ce chapitre, la revue de la littérature sur la simulation avec matériel dans la boucle est présentée. Plus particulièrement, la problématique du prototypage de robots sera étudiée pour ensuite introduire la simulation avec matériel dans la boucle appliquée à la simulation de robots virtuels en contact avec un environnement réel.

1.1 Prototypage de robots

Dans (Martin, Scott et Emami, 2006), on distingue principalement deux méthodes de prototypage de robots. Tout d'abord, le prototypage analytique consiste à simuler l'ensemble du système, ce qui nécessite inévitablement la modélisation de tous les sous-systèmes, incluant le robot et son environnement. Cependant, les équations dynamiques qui régissent les forces de contact du robot avec son environnement, tel que revues dans (Gilardi et Sharf, 2002), ne sont pas toujours connues et peuvent mener à des résultats imprécis. Ceci est particulièrement applicable lorsque le robot réalise un procédé comme le meulage. En effet, les différents modèles de simulation du meulage sont répertoriés dans (Brinksmeier et al., 2006) et on constate qu'ils sont très complexes et nécessitent généralement beaucoup de données expérimentales pour les calibrer. La seconde méthode, appelée prototypage physique, consiste à fabriquer le robot afin d'en étudier son comportement réel. Des résultats beaucoup plus réalistes peuvent être obtenus, mais au détriment de coûts monétaires et temporels élevés.

1.2 Simulation avec matériel dans la boucle ou « hardware-in-the-loop »

La simulation avec matériel dans la boucle, ou souvent appelée « hardware-in-the-loop », consiste à remplacer certaines parties de la boucle de simulation par des composantes matérielles. Ainsi, la qualité de la simulation est améliorée en remplaçant les sous-systèmes dont les modèles sont imprécis ou inconnus par leur équivalent physique. Cette méthode, qui

a pour objectif de palier aux défauts de chacune des méthodes traditionnelles proposées à la section précédente, a émergé dans différentes sphères de la science.

Par exemple, dans le domaine de l'automobile, (Short et Pont, 2005) propose une façon d'utiliser la simulation HIL pour valider la fiabilité et la sécurité des systèmes électroniques embarqués à l'intérieur de véhicules automobiles. Dans ce contexte, la partie matérielle est l'ordinateur embarqué tandis que la partie simulée est la dynamique du véhicule.

En aérospatial, (Leitner, 1996) utilise la technologie hardware-in-the-loop dans un laboratoire afin d'intégrer des composantes à une navette spatiale et de valider leurs capacités. Étant donné qu'il n'est pas réaliste d'effectuer des essais expérimentaux complets, le modèle mathématique de la navette est la partie simulée, tandis que les composantes à intégrer représentent la partie matérielle. Cette méthode permet entre autre de vendre les concepts aux administrateurs, de diagnostiquer à l'avance des problèmes d'intégration et d'explorer des scénarios d'anomalies matérielles et logicielles.

Dans le cadre de ce projet de recherche, la simulation avec matériel dans la boucle est appliquée à la problématique du prototypage de robots. Plus particulièrement, les forces d'interaction du robot avec l'environnement sont supposées inconnues. Un robot réel est donc utilisé pour permettre à un robot simulé d'interagir virtuellement avec un environnement réel. Ainsi, la partie simulée est représentée par le robot virtuel tandis que la partie matérielle est constituée d'un robot et d'un environnement réels. La prochaine section présente l'état de l'art de cet aspect de la simulation HIL sur lequel très peu de chercheurs se sont penchés jusqu'à maintenant.

1.3 La simulation HIL pour l'émulation de robots en contact

L'article (Krenn et Schaefer, 1999) propose une approche intuitive qui consiste à simuler un robot spatial et à utiliser la position de son extrémité comme consigne de position pour un robot industriel terrestre. Cependant, les limitations au niveau de la fréquence

d'échantillonnage et la latence générée par le robot industriel réduisent la plage d'opération dynamique où le système est stable.

Dans (de Carufel, Martin et Piedboeuf, 2000), il est démontré que cette approche, basée sur une consigne de position, comporte des limitations au niveau de la stabilité et des performances, plus particulièrement lorsque le robot entre en contact avec l'environnement. On y propose plutôt l'utilisation d'une commande basée sur l'accélération, qui présente de bons résultats malgré la présence de phénomènes dynamiques non-modélisés, comme la friction. Une commande linéaire proportionnel-dérivée est greffée à la commande d'accélération afin de réduire les erreurs de vitesse et de position. Deux systèmes linéaires simples à un degré de liberté ont été utilisés pour valider le concept : un système à une masse pour le joint rigide du robot réel et un système à deux masses pour le joint flexible du robot spatial virtuel. Ainsi, il a été démontré à l'aide d'un banc d'essai à un degré de liberté, qu'il est possible de contrôler un robot réel rigide de façon à ce qu'il reproduise les mouvements dynamiques de l'outil d'un robot spatial flexible virtuel. L'article propose également une généralisation de l'approche, mais sans présenter de résultats.

Ensuite, (Aghili et Piedboeuf, 2000) propose également l'utilisation d'une commande d'accélération, mais en ajoutant la linéarisation du modèle dynamique du robot réel. Pour obtenir cette commande linéarisante, on introduit le concept de contrainte entre les deux robots et on fait appel à la méthode des multiplicateurs de Lagrange (Goldstein, 1980) pour obtenir le modèle dynamique contraint du robot virtuel. De plus, la stabilisation de Baumgarte[1] (Baumgarte, 1972) est utilisée pour assurer la stabilité exponentielle de l'erreur sur la contrainte. En appliquant cette commande de couple ainsi obtenue sur le robot réel, la

---

[1] La stabilisation de Baumgarte est une reformulation mathématique pour la stabilisation de contraintes de systèmes dynamiques. Cet approche purement cinématique propose d'ajouter les termes de vitesse et de position dans la dérivée seconde de la contrainte afin de simuler un système algébro-différentiel (DAE).

contrainte est en théorie respectée. Cependant, en présence d'erreurs de modélisation, une erreur de suivi va nécessairement apparaître au niveau de la vitesse et de la position. Ainsi, on propose de réimposer la contrainte sur le robot virtuel, c'est-à-dire que la pose du robot virtuel est synchronisée en temps-réel avec celle du robot réel. Cette nouvelle méthode de simulation est basée sur la projection des coordonnées généralisées dans deux sous-espaces orthogonaux représentant les coordonnées dépendantes et indépendantes du robot virtuel. Les coordonnées dépendantes sont obtenues via les coordonnées généralisées du modèle linéarisé du robot réel tandis que les coordonnées indépendantes sont calculées à partir du modèle non contraint du robot virtuel. Des résultats expérimentaux obtenus avec un banc d'essai à un joint démontrent la fiabilité de la méthode dans l'espace libre et en contact.

Dans (Aghili et Piedboeuf, 2002), les travaux sur la commande d'accélération sont poursuivis. La différence avec la méthode proposée précédemment réside principalement dans la stabilisation de l'erreur sur la contrainte. Au lieu d'utiliser la stabilisation de Baugmarte, on ajoute une commande linéaire proportionnel-dérivée à la commande d'accélération afin d'obtenir une stabilité asymptotique. Des essais expérimentaux effectués avec un banc d'essai composé d'un joint rotatif ont démontrés qu'il est possible de reproduire fidèlement la dynamique d'un joint flexible avec la méthode proposée.

Dans (Aghili et Piedboeuf, 2006), la commande d'accélération est formalisée comme l'application de la méthode bien connue du couple pré-calculé dans l'espace de la tâche. Aussi, une méthode alternative pour calculer les états du robot virtuel est proposée. Elle consiste à utiliser l'algorithme CLIK[2] (Wolovich et Elliott, 1984) pour calculer les coordonnées généralisées dépendantes du robot virtuel à partir des coordonnées généralisées

---

[2] L'algorithme CLIK (« Closed-Loop-Inverse-Kinematics ») est une méthode en boucle fermée qui permet de calculer la cinématique inverse. Elle peut être basée notamment sur la transposée ou la pseudo-inverse de la matrice jacobienne.

du robot réel. Le robot virtuel est simulé en appliquant la contrainte et on y présente également la méthode sans application de la contrainte. Des résultats expérimentaux sont obtenus en simulant un robot flexible avec un robot réel rigide à trois degrés de liberté. En présence d'erreurs de modélisation, il est démontré que le schéma de simulation avec application de la contrainte résulte en une erreur de position tandis que le schéma de simulation sans application de la contrainte produit plutôt une erreur sur la force de contact.

Des résultats prometteurs sont obtenus avec les deux schémas de simulation HIL proposés par l'ASC (Aghili et Piedboeuf, 2006). Cependant, cette approche est appliquée dans un contexte de robotique spatiale et fait intervenir plusieurs limitations au niveau de la validation :

i) La commande utilisée pour valider l'approche est pré-calculée est n'est donc pas influencée par la rétroaction de force et de position du robot virtuel;
ii) Le robot réel utilisé comporte des réducteurs harmoniques qui introduisent une flexibilité non négligeable et qui compromet la rigidité du manipulateur;
iii) Le robot virtuel est flexible, ce qui compromet également la rigidité du système et peut constituer une limitation dans un autre contexte.

Ces limitations amènent à reconsidérer la validation de cette approche dans le contexte défini dans ce mémoire : la simulation avec matériel dans la boucle du robot de meulage de l'IREQ. En effet, les robots de meulage peuvent être caractérisés par une loi de commande de force ou d'impédance (Chiaverini, Siciliano et Villani, 1999; Lawrence, 1988; Siciliano et Villani, 1999). Aussi, le meulage exige une grande rigidité qui est plus difficile à obtenir lorsque les joints du robot sont munis de réducteurs de vitesse harmoniques (Kircanski et Goldenberg, 1997). Le contexte choisi pour la validation fait donc intervenir un robot réel à entrainement direct (sans réducteur de vitesse) et un robot virtuel rigide accompagné d'une loi de commande d'impédance. La loi de commande d'impédance a été choisie parce qu'elle peut être interprétée comme une généralisation de la commande de force.

On constate à la lumière de cette revue de la littérature que le domaine de la simulation avec matériel dans la boucle est en pleine effervescence. L'évolution du développement des algorithmes de commande pour la simulation de robots en contact avec un environnement réel a été présentée. L'application de la simulation HIL à cet aspect particulier de la robotique est en pleine croissance. Il existe encore peu de publications sur le sujet. Cependant, les deux principales approches proposées par l'ASC sont prometteuses et seront validées dans le contexte préliminaire du développement d'un robot de meulage à l'IREQ.

CHAPITRE 2

MODÉLISATION DES ROBOTS ET DE L'ENVIRONNEMENT

L'objectif de la simulation avec matériel dans la boucle est d'utiliser un robot réel afin de reproduire la dynamique d'un robot virtuel dans le but d'intégrer l'interaction avec l'environnement réel dans la simulation. Ainsi, le robot simulé va interagir virtuellement avec un environnement réel. Afin de valider cette méthode, la simulation idéale est comparée à la simulation avec matériel dans la boucle. Par conséquent, les modèles du robot virtuel et de l'environnement doivent être développés. De plus, le modèle du robot réel est requis, puisque la loi de commande proposée au chapitre 4 est basée sur la méthode du couple pré-calculé (J. Craig, 2004). Ce chapitre présente donc la modélisation des robots virtuel et réel ainsi que celle de l'environnement.

Avant tout, il importe de rappeler certaines conventions et hypothèses utilisées dans ce chapitre. Le robot virtuel, dénoté par l'indice s, et le robot réel, dénoté par l'indice r, possèdent respectivement m et n degrés de liberté. L'espace de la tâche (environnement) possède n degrés de liberté et l'hypothèse que $m \geq n$ est posé.

2.1 Modélisation du robot réel

Le robot réel utilisé pour les expérimentations est schématisé à la Figure 2.1. Il possède deux joints prismatiques actionnés par des moteurs à entraînement direct, donc il n'y a aucun engrenage. Le premier joint est parallèle et le second est perpendiculaire à la surface de l'environnement. Plus d'informations sur le montage physique et l'architecture de contrôle sont présentées à la section 5.1.

Cette section présente en détails la modélisation cinématique et dynamique du robot réel. Cependant, une procédure de prototypage rapide, qui sera présentée à la section 5.2, est utilisée afin de générer automatiquement la majorité de ces équations. Ainsi, certains de ces détails sont présentés uniquement à titre indicatif.

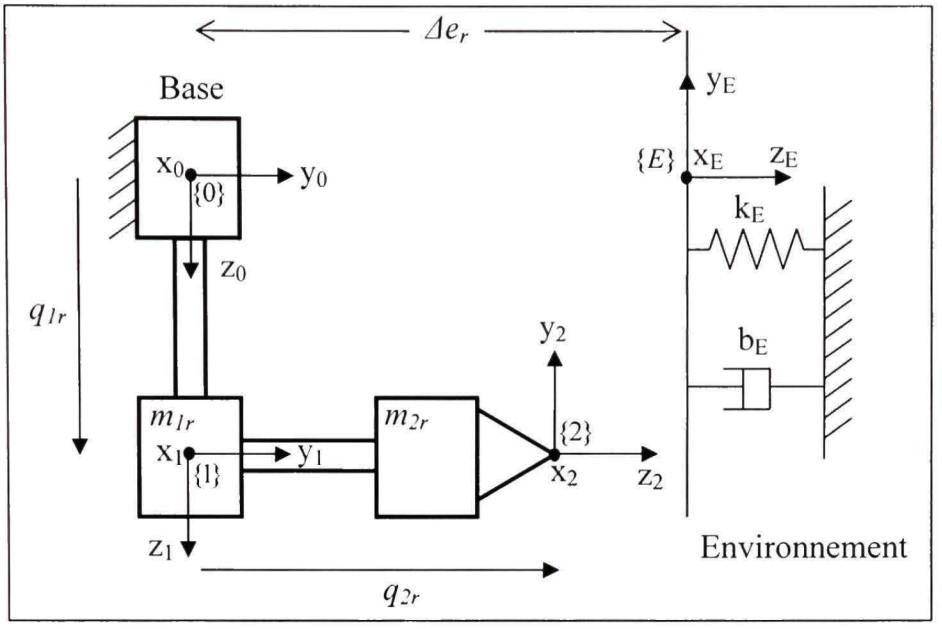

Figure 2.1 ***Schéma du robot réel.***

2.1.1 Cinématique directe

La cinématique directe du robot réel consiste à exprimer la position du dernier repère du robot par rapport au repère de l'environnement en fonction de $\mathbf{q}_r \in \mathbb{R}^n = \begin{bmatrix} q_{1r} & q_{2r} \end{bmatrix}^T$ où q_{1r} et q_{2r} représentent respectivement les coordonnées généralisées des joints prismatiques 1 et 2. La Figure 2.1 présente le schéma du robot réel ainsi que la position de chacun des repères, qui ont été déterminées par la méthode de Craig (J. Craig, 2004). Le repère de base $\{0\}$ du robot est positionné à une distance Δe_r du repère de l'environnement $\{E\}$. Le Tableau 2.1 présente les paramètres de Danevit-Hartenberg (DH) modifiés du robot réel.

Tableau 2.1

Paramètres DH du robot réel

Membre *i*	α_i	a_{i-1}	d_i	q_i
1	0	0	q_{1r}	0
2	-π /2	0	q_{2r}	0

Selon la convention de Craig (J. Craig, 2004), la matrice de transformation homogène qui relie le repère $\{i\}$ au repère $\{i-1\}$ est donnée par :

$$ {}^{i-1}_{i}\mathbf{T} = \begin{bmatrix} cq_i & -sq_i & 0 & \alpha_{i-1} \\ sq_i c\alpha_{i-1} & cq_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ sq_i s\alpha_{i-1} & cq_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.1} $$

Selon la Figure 2.1, la transformation qui relie la base du robot au repère de l'environnement est décrite par une rotation de $-\pi/2$ autour de x_0 suivi d'une translation de Δe_r le long de y_0:

$$ {}^{0}_{E}\mathbf{T}_r = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & \Delta e_r \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2} $$

La chaîne cinématique complète est donc :

$$ {}^{E}_{2}\mathbf{T}_r = {}^{0}_{E}\mathbf{T}_r^{-1} \times {}^{0}_{1}\mathbf{T}_r \times {}^{1}_{2}\mathbf{T}_r = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -q_{1r} \\ 0 & 0 & 1 & q_{2r} - \Delta e_r \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3} $$

Étant donné que l'extrémité du robot se déplace dans un plan et qu'il ne subit aucun déplacement angulaire, la pose du robot peut être exprimée sous la forme simplifiée suivante :

$$ \mathbf{p}_r(\mathbf{q}_r) = \begin{bmatrix} y_r \\ z_r \end{bmatrix} = \begin{bmatrix} -q_{1r} \\ q_{2r} - \Delta e_r \end{bmatrix} \tag{2.4} $$

La cinématique directe du robot réel est relativement simple et aurait pu être déterminée par inspection à partir de la Figure 2.1. Cependant, l'approche logicielle utilisée, qui sera décrite à la section 5.2, permet d'automatiser le calcul de la cinématique à partir du tableau des paramètres HD. Ainsi, la simulation du système peut facilement être adaptée à des changements de configurations des robots virtuels et/ou réels.

2.1.2 Cinématique inverse

La cinématique inverse du robot réel consiste à déterminer les coordonnées généralisées pour une pose donnée. À partir de l'équation (2.4), on détermine par inspection que :

$$\mathbf{q}_r = \begin{bmatrix} q_{1r} \\ q_{2r} \end{bmatrix} = \begin{bmatrix} -y_r \\ z_r + \Delta e_r \end{bmatrix} \tag{2.5}$$

2.1.3 Cinématique différentielle directe

La cinématique différentielle directe permet de déterminer la vitesse du robot dans l'espace de la tâche, c'est-à-dire par rapport à l'environnement. Pour ce faire, la matrice jacobienne de la pose donnée par l'équation (2.4) est utilisée:

$$\dot{\mathbf{p}}_r = \begin{bmatrix} \dot{y}_r \\ \dot{z}_r \end{bmatrix} = \mathbf{J}_r(\mathbf{q}_r)\dot{\mathbf{q}}_r = \begin{bmatrix} \dfrac{\partial y_r}{\partial q_{1r}} & \dfrac{\partial y_r}{\partial q_{2r}} \\ \dfrac{\partial z_r}{\partial q_{1r}} & \dfrac{\partial z_r}{\partial q_{2r}} \end{bmatrix} \begin{bmatrix} \dot{q}_{1r} \\ \dot{q}_{2r} \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{q}_{1s} \\ \dot{q}_{2s} \end{bmatrix} \tag{2.6}$$

où $\mathbf{J}_r \in \mathbb{R}^{n \times n}$ représente la matrice jacobienne exprimée dans le repère de l'environnement. La transposée de cette matrice sera également utile pour transformer les forces statiques de l'espace de travail à l'espace des articulations.

2.1.4 Cinématique différentielle inverse

La cinématique différentielle inverse du robot réel consiste à trouver la relation inverse de (2.6), c'est-à-dire de déterminer les vitesses des articulations en fonction de la vitesse du robot par rapport à l'environnement. La solution suivante est obtenue en inversant la matrice jacobienne de (2.6) :

$$\dot{\mathbf{q}}_r = \begin{bmatrix} \dot{q}_{1r} \\ \dot{q}_{2r} \end{bmatrix} = \mathbf{J}_r^{-1}(\mathbf{q}_r)\dot{\mathbf{p}}_r = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{y}_r \\ \dot{z}_r \end{bmatrix} \tag{2.7}$$

2.1.5 Modèle dynamique

De façon générale, le modèle dynamique d'un robot peut être obtenu par différentes méthodes. Les deux approches les plus populaires sont celles de Newton-Euler (J. Craig, 2004) et de Lagrange (Goldstein, 1980). Avec l'une ou l'autre des méthodes, le modèle dynamique du robot peut s'exprimé sous la forme générale suivante :

$$\mathbf{M}_r(\mathbf{q}_r)\ddot{\mathbf{q}}_r + \mathbf{F}_r(\mathbf{q}_r, \dot{\mathbf{q}}_r) - \mathbf{J}_r^T \mathbf{f}_e = \boldsymbol{\tau}_r \tag{2.8}$$

où $\mathbf{M}_r \in \mathbb{R}^{n \times n}$ est la matrice de masse, $\mathbf{F}_r \in \mathbb{R}^n$ est le vecteur des forces de Coriolis et centrifuges, $\mathbf{J}_r \in \mathbb{R}^{n \times n}$ est la matrice jacobienne, $\mathbf{f}_e \in \mathbb{R}^n$ est le vecteur des forces externes et $\boldsymbol{\tau}_r \in \mathbb{R}^n$ est le vecteur des forces généralisées. Puisque le robot réel n'admet aucune rotation, son modèle peut facilement être déterminé à partir de la seconde loi de Newton *f=ma*. De plus, puisqu'il n'y a pas de mouvement rotatif et qu'on suppose que les forces de frottement sont négligeables $\mathbf{F}_r = 0$. Le modèle dynamique s'exprime alors de la façon suivante :

$$\begin{bmatrix} m_{1r} + m_{2r} & 0 \\ 0 & m_{2r} \end{bmatrix} \begin{bmatrix} \ddot{q}_{1r} \\ \ddot{q}_{2r} \end{bmatrix} - \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} f_y \\ f_z \end{bmatrix} = \begin{bmatrix} \tau_{1r} \\ \tau_{2r} \end{bmatrix} \tag{2.9}$$

Pour le robot réel, les forces de l'environnement $\mathbf{f}_e$ sont celles mesurées par le capteur de force.

2.2 Modélisation de l'environnement

L'environnement, tel que présenté à la Figure 2.2, est modélisé par un ressort et un amortisseur. Les forces de l'environnement sont représentées dans l'espace de la tâche à n=2 dimensions. Étant donné que la friction n'est pas considérée, la force de contact dans le repère de l'environnement ne possède qu'une composante en Z et est donc fonction de la position selon l'axe Z :

$$\mathbf{f}_e(p_z) = \begin{bmatrix} f_y & f_z \end{bmatrix}^T = \begin{bmatrix} 0 & -f_c(p_z) \end{bmatrix}^T \tag{2.10}$$

La force de contact $f_c(z_r)$ est définie comme suit:

$$f_c(z_r) = \begin{cases} k_e p_z + b_e \dot{p}_z & \text{si } p_z > 0 \\ 0 & \text{sinon} \end{cases} \tag{2.11}$$

où p_z est la position du robot selon l'axe Z du repère de l'environnement, k_e est la constante de rigidité de l'environnement et b_e est la constante d'amortissement de l'environnement. Dans le domaine de Laplace, le modèle de l'environnement peut être formulé de la façon suivante :

$$\mathbf{F}_e(s) = -(\mathbf{B}_e s + \mathbf{K}_e)\mathbf{p}(s) = -\left(\begin{bmatrix} 0 & 0 \\ 0 & b_e \end{bmatrix} s + \begin{bmatrix} 0 & 0 \\ 0 & k_e \end{bmatrix}\right)\begin{bmatrix} p_y(s) \\ p_z(s) \end{bmatrix} \tag{2.12}$$

lorsqu'il y a contact avec l'environnement, c'est-à-dire lorsque $p_z > 0$. Cette représentation dans le domaine de Laplace sera utilisée lors de l'analyse du système de commande d'impédance présenté à la section 3.2.2.

2.3 Modélisation du robot virtuel

Le robot virtuel à l'étude, présenté à la Figure 2.2, possède deux articulations rotatives selon une configuration planaire.

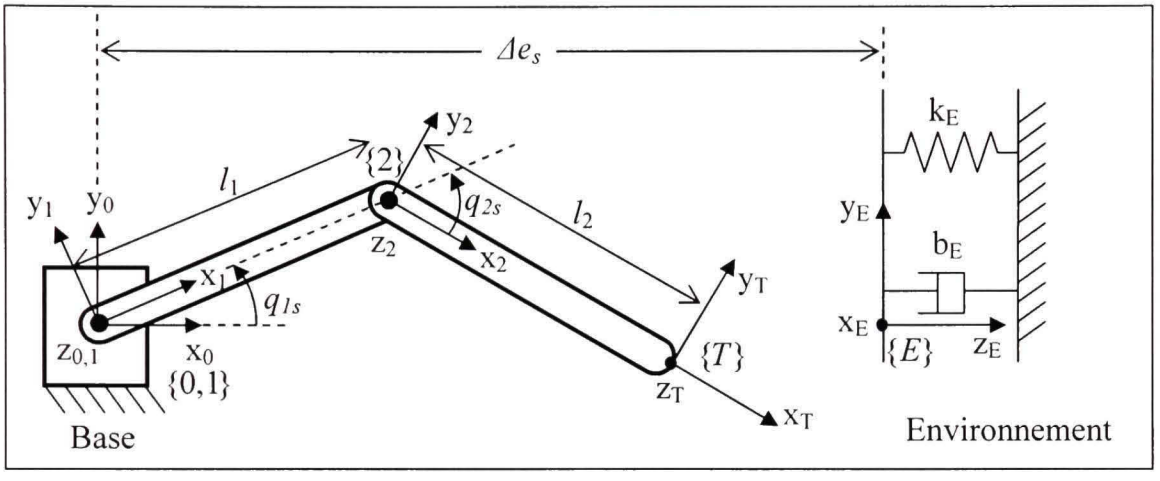

Figure 2.2 ***Schéma du robot virtuel.***

Étant donné que le robot virtuel sera synchronisé avec le robot réel, le développement de son modèle dynamique est nécessaire pour l'élaboration des lois de commande et des simulations qui seront présentées aux chapitres 3 et 4. Les modèles de la cinématique directe et inverse, sont préalablement développés puisqu'ils seront également nécessaires. Comme dans le cas de la modélisation du robot réel, les modèles cinématiques et dynamiques sont présentés en détails. Cependant, la procédure de prototypage rapide, qui sera présentée à la section 5.2, permet de générer automatiquement la majorité de ces modèles. Ainsi, certains des détails de calcul sont présentés uniquement à titre indicatif.

2.3.1 Cinématique directe

La cinématique directe consiste à exprimer la position du repère de l'outil $\{T\}$ par rapport au repère de l'environnement $\{E\}$ en fonction des coordonnées généralisées $\mathbf{q}_s \in \mathbb{R}^m = \begin{bmatrix} q_{1s} & q_{2s} \end{bmatrix}^T$ (m=2). Les repères de chacune des articulations ont été positionnés

selon la méthode de Craig (J. Craig, 2004) tel qu'illustré sur la Figure 2.2. Le Tableau 2.2 présente les paramètres de Danevit-Hartenberg (DH) modifiés du robot virtuel.

Tableau 2.2

Paramètres DH modifiés du robot virtuel

Membre *i*	α_i	a_{i-1}	d_i	q_i
1	0	0	0	q_{1s}
2	0	l_1	0	q_{2s}

La matrice de transformation homogène qui relie le repère $\{i\}$ au repère $\{i-1\}$ est donnée par (2.1). La transformation entre le repère de l'outil $\{T\}$ et celui de l'articulation 2 est définie par une translation le long de l'axe X_2 :

$$ {}^{2}_{T}\mathbf{T}_s = \begin{bmatrix} 1 & 0 & 0 & l_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.13} $$

De façon similaire, la transformation qui relie le repère de base du robot au repère de l'environnement est définie par :

$$ {}^{0}_{E}\mathbf{T}_s = \begin{bmatrix} 0 & 0 & 1 & \Delta e_s \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.14} $$

La cinématique directe du robot qui relie le repère de l'outil $\{T\}$ au repère de l'environnement $\{E\}$ peut être déterminée à partir des équations (2.1), (2.13), (2.14) et du Tableau 2.2 :

$$ {}^{E}_{T}\mathbf{T}_s = {}^{0}_{E}\mathbf{T}_s^{-1} \times {}^{0}_{1}\mathbf{T}_s \times {}^{1}_{2}\mathbf{T}_s \times {}^{2}_{T}\mathbf{T}_s = \begin{bmatrix} 0 & 0 & -1 & 0 \\ s_{1s2s} & c_{1s2s} & 0 & l_1 s_{1s} + l_2 s_{1s2s} \\ c_{1s2s} & -s_{1s2s} & 0 & l_1 c_{1s} + l_{2s} c_{1s2s} - \Delta e_s \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.15} $$

où $s_{1s} = \sin(q_{1s})$, $c_{1s} = \cos(q_{1s})$, $s_{1s2s} = \sin(q_{1s} + q_{2s})$ et $c_{1s2s} = \cos(q_{1s} + q_{2s})$. Étant donné que l'extrémité du robot se déplace dans un plan, la pose du robot virtuel peut être exprimée par une représentation minimale de la position:

$$ \mathbf{p}_s(\mathbf{q}_s) = \begin{bmatrix} y_s \\ z_s \end{bmatrix} = \begin{bmatrix} l_1 s_{1s} + l_2 s_{1s2s} \\ l_1 c_{1s} + l_{2s} c_{1s2s} - \Delta e_s \end{bmatrix} \tag{2.16} $$

2.3.2 Cinématique inverse

La cinématique inverse (J. Craig, 2004) du robot virtuel consiste à déterminer les coordonnées généralisées pour une pose donnée. Donc, l'inverse de l'équation (2.16) doit être déterminé. Tout d'abord, la somme du carré des positions est calculée :

$$ y_s^2 + (z_s + \Delta e_s)^2 = l_1^2 s_{1s}^2 + l_2^2 s_{1s2s}^2 + 2 l_1 l_2 s_{1s} s_{1s2s} + l_1^2 c_{1s}^2 + l_2^2 c_{1s2s}^2 + 2 l_1 l_2 c_{1s} c_{1s2s} \tag{2.17} $$

Après avoir simplifié trigonométriquement et isolé c_{2s} dans l'équation (2.17), on obtient :

$$ c_{2s} = \frac{y_s^2 + (z_s + \Delta e_s)^2 - l_1^2 - l_2^2}{2 l_1 l_2} \tag{2.18} $$

La solution de q_{2s} peut être déterminé à partir d'identités trigonométriques couramment utilisées en robotique (J. Craig, 2004) :

$$q_{2s} = \text{atan2}\left(c_{2s}, \pm\sqrt{1-c_2^2}\right) \tag{2.19}$$

La fonction *atan2* permet de calculer l'arc-tangente de coordonnées (x, y) en tenant compte des quatre quadrants, c'est-à-dire pour la plage $]-\pi,\pi]$. En appliquant de nouveau une identité trigonométrique, A.5 dans (J. Craig, 2004), l'équation (2.16) est réécrite sous la forme suivante :

$$\begin{bmatrix} y_s \\ z_s + \Delta e_s \end{bmatrix} = \begin{bmatrix} (l_1 + l_2 c_{2s}) s_{1s} + l_2 s_{2s} c_{1s} \\ (l_1 + l_2 c_{2s}) c_{1s} - l_2 s_{2s} s_{1s} \end{bmatrix} \tag{2.20}$$

En appliquant à l'équation (2.20) les identités trigonométriques appropriées, C.11 et C.12 dans (J. Craig, 2004), on trouve que :

$$q_{1s} = \text{atan2}\left((l_1 + l_2 c_{2s}) y_s - l_2 s_{2s}(z_s + \Delta e_s), (l_1 + l_2 c_{2s})(z_s + \Delta e_s) + l_2 s_{2s} y_s\right) \tag{2.21}$$

La cinématique inverse du robot virtuel est utilisée dans le cadre de la simulation avec matériel dans la boucle présentée au chapitre 4, et plus particulièrement dans le cas de la simulation avec application de la contrainte présentée à la section 4.2.1.

2.3.3 Cinématique différentielle directe

La cinématique différentielle directe permet de déterminer la vitesse du repère de l'outil $\{T\}$ par rapport au repère de l'environnement $\{E\}$. Cette cinématique des vitesses peut être obtenue en dérivant en chaîne la relation (2.16) :

$$\dot{\mathbf{p}}_s = \mathbf{J}_s(\mathbf{q}_s)\dot{\mathbf{q}}_s = \begin{bmatrix} \dfrac{\partial y_s}{\partial q_{1s}} & \dfrac{\partial y_s}{\partial q_{2s}} \\ \dfrac{\partial z_s}{\partial q_{1s}} & \dfrac{\partial z_s}{\partial q_{2s}} \end{bmatrix} \begin{bmatrix} \dot{q}_{1s} \\ \dot{q}_{2s} \end{bmatrix} = \begin{bmatrix} l_1 c_{1s} + l_2 c_{1s2s} & l_2 c_{1s2s} \\ -l_1 s_{1s} - l_2 s_{1s2s} & -l_2 s_{1s2s} \end{bmatrix} \begin{bmatrix} \dot{q}_{1s} \\ \dot{q}_{2s} \end{bmatrix} \tag{2.22}$$

où $\mathbf{J}_s \in \mathbb{R}^{n\times m}$ est la matrice jacobienne exprimée dans le repère de l'environnement. La transposée de cette matrice sera également utile pour transformer les forces statiques de l'espace de travail à l'espace des articulations.

2.3.4 Cinématique différentielle inverse

La solution inverse de la cinématique différentielle permet de calculer les vitesses des articulations du robot pour une vitesse cartésienne donnée, soit la vitesse linéaire de l'outil par rapport au repère de l'environnement. Dans ce cas-ci, la solution est obtenue directemnt en inversant la matrice jacobienne $\mathbf{J}_s$ tel qu'exprimé par (2.22) :

$$\dot{\mathbf{q}}_s = \mathbf{J}_s^{-1}\dot{\mathbf{p}}_s = \frac{1}{l_1 s_{2s}}\begin{bmatrix} s_{1s2s} & c_{1s2s} \\ \dfrac{-l_1 s_{1s} - l_2 s_{1s2s}}{l_2} & \dfrac{-l_1 c_{1s} - l_2 c_{1s2s}}{l_2} \end{bmatrix}\begin{bmatrix} \dot{y}_s \\ \dot{z}_s \end{bmatrix} \tag{2.23}$$

L'inverse de la matrice jacobienne sera utilisé au chapitre 3 pour la partie linéarisation du contrôleur du robot virtuel.

2.3.5 Modèle dynamique sans contrainte

L'objectif du modèle dynamique est de déterminer les équations qui régissent les mouvements du robot virtuel. Comme il a déjà été mentionné, une des méthodes bien connues pour la modélisation des robots est celle de Lagrange (Goldstein, 1980). L'équation de Lagrange pour les systèmes non conservatifs et sans contrainte est définie comme :

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{\mathbf{q}}_s} - \frac{\partial T}{\partial \mathbf{q}_s} + \frac{\partial V}{\partial \mathbf{q}_s} = \mathbf{J}_s^T \mathbf{f}_e + \boldsymbol{\tau}_s \tag{2.24}$$

où T est l'énergie cinétique totale, V est l'énergie potentielle totale, $\mathbf{q}_s$ est le vecteur des coordonnées généralisées, $\boldsymbol{\tau}_s \in \mathbb{R}^m$ est le vecteur des forces généralisées, $\mathbf{J}_s \in \mathbb{R}^{n\times m}$ est la matrice jacobienne et $\mathbf{f}_e$ est le vecteur des forces externes qui, dans le contexte étudié dans ce mémoire, sont celles de l'interaction avec l'environnement. Une fois le modèle dynamique calculé à partir de (2.24), l'équation dynamique du robot virtuel peut se présenter sous la forme générale suivante (J. Craig, 2004) :

$$\mathbf{M}_s(\mathbf{q}_s)\ddot{\mathbf{q}}_s + \mathbf{F}_s(\mathbf{q}_s, \dot{\mathbf{q}}_s) - \mathbf{J}_s^T \mathbf{f}_e = \boldsymbol{\tau}_s \tag{2.25}$$

où $\mathbf{M}_s \in \mathbb{R}^{m\times m}$ est la matrice de masse, $\mathbf{F}_s \in \mathbb{R}^m$ est le vecteur des forces de Coriolis et centrifuges, $\mathbf{J}_s \in \mathbb{R}^{n\times m}$ est la matrice jacobienne, $\mathbf{f}_e \in \mathbb{R}^n$ est le vecteur des forces externes et $\boldsymbol{\tau}_s \in \mathbb{R}^m$ est le vecteur des couples aux articulations. À partir du principe même de la simulation avec matériel dans la boucle, le robot virtuel est soumis aux mêmes forces externes que le robot réel, soit $\mathbf{f}_e$. D'ailleurs, on remarque que ces forces sont définies dans l'espace de la tâche à n-dimensions et que la matrice jacobienne $\mathbf{J}_s$ permet de transformer ces efforts aux m articulations du robot virtuel. Les éléments du modèle dynamique sont obtenus automatiquement grâce à la procédure de prototypage rapide qui sera expliquée à la section 5.2. Cependant, par souci de compréhension, les détails du développement du modèle dynamique (2.25) sont présentés à l'annexe I. Le résultat suivant est obtenu :

$$\mathbf{M}_s = \begin{bmatrix} 2 + \frac{5}{4} l_1^2 m_{2s} + l_1^2 c_{2s} m_{2s} + \frac{1}{4} l_1^2 m_{1s} & 1 + \frac{1}{4} l_1^2 m_{2s} + \frac{1}{2} l_1^2 c_{2s} m_{2s} \\ 1 + \frac{1}{4} l_1^2 m_{2s} + \frac{1}{2} l_1^2 c_{2s} m_{2s} & 1 + \frac{1}{4} l_1^2 m_{2s} \end{bmatrix} \tag{2.26}$$

$$\mathbf{F}_s = \begin{bmatrix} -\frac{1}{2} l_1^2 m_{2s} s_{2s} \dot{q}_{2s} (2\dot{q}_{1s} + \dot{q}_{2s}) \\ \frac{1}{2} l_1^2 m_{2s} s_{2s} \dot{q}_1^2 \end{bmatrix} \tag{2.27}$$

$$\mathbf{J}_s = \begin{bmatrix} l_1 c_{1s} + l_2 c_{1s2s} & l_2 c_{1s2s} \\ -l_1 s_{1s} - l_2 s_{1s2s} & -l_2 s_{1s2s} \end{bmatrix} \tag{2.28}$$

Dans le cas de la simulation idéale, les forces d'interaction avec l'environnement sont calculées à partir du modèle de l'environnement présenté à la section 2.2. Pour des fins de comparaison et de validation, l'hypothèse suivante est posée: le modèle de l'environnement du robot virtuel est équivalent à celui de l'environnement du robot réel. Afin de garantir ceci, l'environnement réel est identifié au chapitre 5 et les paramètres trouvés sont utilisés pour la simulation idéale. Tel que présenté au chapitre 4, dans le cas de la simulation avec matériel dans la boucle, les forces d'interaction avec l'environnement sont celles mesurées par le capteur de force du robot réel.

2.3.6 Modèle dynamique avec contrainte

Au chapitre 4, la réalisation de la simulation avec matériel dans la boucle nécessitera l'ajout d'une contrainte au modèle du robot virtuel. En effet, la pose du robot virtuel devra être contrainte avec celle du robot réel. Cette contrainte holonomique est exprimée de la façon suivante :

$$\mathbf{\Phi}\left(\mathbf{q}_s, \mathbf{q}_r(t)\right) = \mathbf{p}_r\left(\mathbf{q}_r(t)\right) - \mathbf{p}_s\left(\mathbf{q}_s\right) = 0 \tag{2.29}$$

Puisque la contrainte dépend explicitement du temps, on dira aussi qu'elle est rhéonomique (Goldstein, 1980). Le modèle dynamique du robot sous la contrainte (2.29) peut être exprimé en utilisant la méthode des multiplicateurs de Lagrange (Goldstein, 1980) :

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{\mathbf{q}}_s} - \frac{\partial T}{\partial \mathbf{q}_s} + \frac{\partial V}{\partial \mathbf{q}_s} = \mathbf{\Phi}_{q_s}^T \boldsymbol{\lambda} + \boldsymbol{\tau}_s \tag{2.30}$$

où T est l'énergie cinétique totale, V est l'énergie potentielle totale, $\mathbf{q}_s \in \mathbb{R}^m$ est le vecteur des coordonnées généralisées, $\boldsymbol{\tau}_s \in \mathbb{R}^m$ est le vecteur des forces généralisées, $\boldsymbol{\Phi}_{q_s} = \partial\boldsymbol{\Phi} / \partial\mathbf{q}_s = -\mathbf{J}_s$ est la matrice jacobienne de la contrainte et $\boldsymbol{\lambda} \in \mathbb{R}^n$ est le vecteur des multiplicateurs de Lagrange qui représentent les forces de contrainte. Le modèle du robot virtuel contraint peut également s'exprimer sous une forme générale par le système d'équations suivant :

$$\mathbf{M}_s(\mathbf{q}_s)\ddot{\mathbf{q}}_s + \mathbf{F}_s(\mathbf{q}_s, \dot{\mathbf{q}}_s) + \boldsymbol{\Phi}_{q_s}^T \boldsymbol{\lambda} = \boldsymbol{\tau}_s \tag{2.31}$$

$$\boldsymbol{\Phi}(\mathbf{q}_s, \mathbf{q}_r(t)) = 0 \tag{2.32}$$

où $\mathbf{M}_s \in \mathbb{R}^{m \times m}$ est la matrice de masse, $\mathbf{F}_s \in \mathbb{R}^m$ est le vecteur des forces de Coriolis et centrifuges.

Dans de ce chapitre, la modélisation de l'environnement et des robots réel et virtuel a été présentée. Dans le cas des deux robots, les modèles cinématiques et dynamiques non-contraints sont développés. Par contre, dans le cas du robot virtuel, un modèle dynamique avec contrainte est aussi présenté puisqu'il est requis pour le développement de la simulation avec matériel dans la boucle. Dans les deux prochains chapitres, la simulation idéale et la simulation avec matériel dans la boucle sont présentées pour en venir finalement à une comparaison expérimentale.

CHAPITRE 3

SIMULATION IDÉALE

L'objectif de la simulation idéale est d'avoir une base de comparaison afin de valider la simulation avec matériel dans la boucle qui sera présentée au chapitre 4. La Figure 3.1 présente le schéma bloc typique de la simulation d'un système de commande robotique en interaction avec son environnement. Il comprend notamment le contrôleur, le modèle du robot virtuel et le modèle de l'environnement.

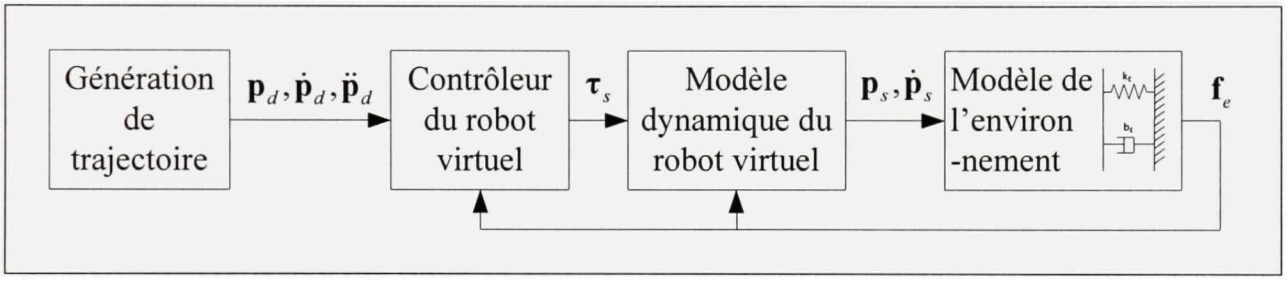

Figure 3.1 ***Schéma bloc de la simulation idéale.***

Dans le cas de la simulation idéale, le robot virtuel est simulé à partir du modèle dynamique non contraint défini par l'équation (2.25). En effet, les états du robot sont calculés par intégration numérique de l'accélération :

$$\ddot{\mathbf{q}}_s = \mathbf{M}_s^{-1}\left(\mathbf{J}_s^T\mathbf{f}_e + \boldsymbol{\tau}_s - \mathbf{F}_s\right) \tag{3.1}$$

Les forces externes $\mathbf{f}_e$, générées par l'environnement virtuel, sont calculées à partir du modèle développé à la section 2.2. Ensuite, elles sont injectées directement dans le modèle du robot (3.1) et dans le contrôleur décrit ci-après.

3.1 Contrôleur du robot virtuel

Le contrôleur du robot virtuel a été sélectionné de façon à ce qu'il y ait une rétroaction de force et de position. De cette façon, il sera possible de comparer la simulation idéale à la simulation avec matériel dans la boucle en se basant sur le comportement en boucle fermée de ces deux mesures. Ainsi, tel qu'illustré à la Figure 3.2, le robot virtuel est contrôlé par une loi de commande par impédance avec une boucle interne de position (Chiaverini, Siciliano et Villani, 1999). Le contrôleur se sous-divise en quatre parties :

- La **linéarisation dans l'espace articulaire** qui linéarise la dynamique du robot virtuel pour obtenir les couples à appliqués aux articulations.
- La **linéarisation dans l'espace de tâche** qui transforme les accélérations dans l'espace cartésien en efforts dans l'espace articulaire;
- Le **contrôleur linéaire proportionnel-dérivée** (PD) qui asservit la position à l'aide d'une commande d'accélération;
- La **loi d'impédance** qui module la trajectoire désirée afin de respecter l'impédance cible;

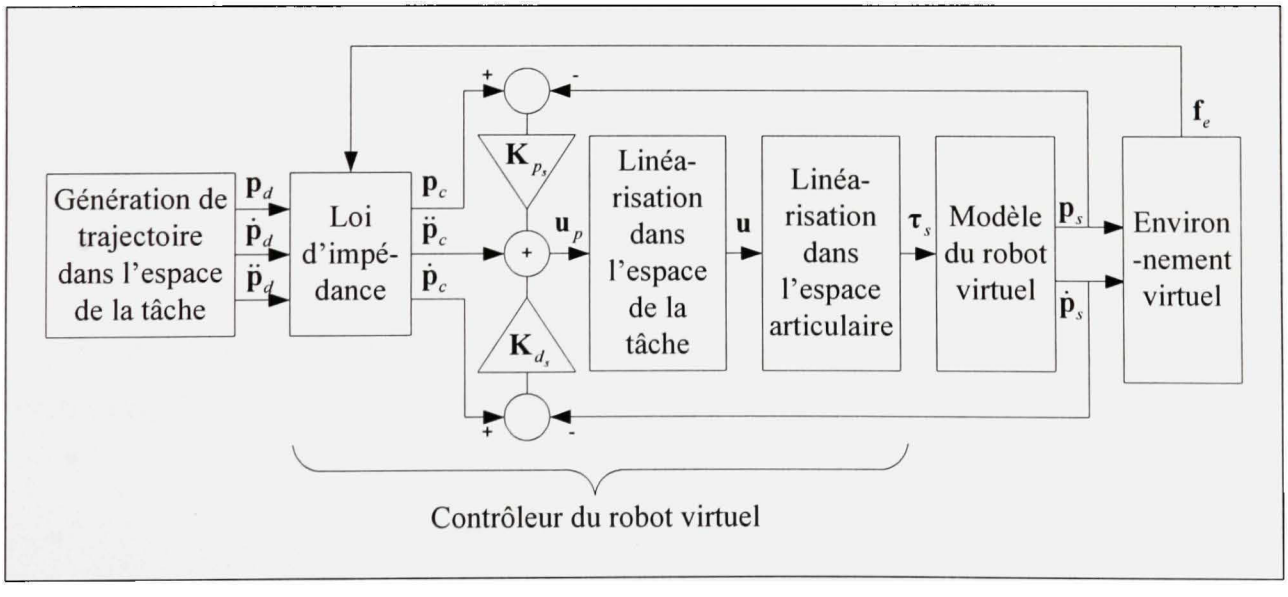

Figure 3.2 ***Schéma bloc de la simulation idéale avec le contrôleur d'impédance.***

3.1.1 Linéarisation dans l'espace articulaire

Cette première étape de linéarisation permet de découpler et linéariser la dynamique du robot dans l'espace articulaire. La méthode bien connue du couple pré-calculé est utilisée (Chiaverini, Siciliano et Villani, 1999) pour obtenir :

$$\boldsymbol{\tau}_s = \mathbf{M}_s\mathbf{u} + \mathbf{F}_s - \mathbf{J}_s^T\mathbf{f}_e \tag{3.2}$$

3.1.2 Linéarisation dans l'espace de la tâche

Cette seconde étape de linéarisation permet de transposer les efforts dans l'espace de la tâche en efforts dans l'espace des articulations. Tel que défini dans (Chiaverini, Siciliano et Villani, 1999), la seconde étape de linéarisation est effectuée par :

$$\mathbf{u} = \mathbf{J}_s^{-1}\left(\mathbf{u}_p - \dot{\mathbf{J}}_s\dot{\mathbf{q}}_s\right) \tag{3.3}$$

où $\mathbf{J}_s \in \mathbb{R}^{n\times m}$ est la matrice jacobienne du robot virtuel et $\dot{\mathbf{J}}_s \in \mathbb{R}^{n\times m}$ sa dérivée par rapport au temps.

3.1.3 Contrôleur linéaire proportionnel-dérivé

Le contrôleur linéaire proportionnel-dérivée (PD) fait partie de la boucle interne de position. En se rapportant à la Figure 3.2, on voit que la commande linéaire est effectuée dans l'espace de la tâche. On définit la commande PD par l'équation suivante :

$$\mathbf{u}_p = \ddot{\mathbf{p}}_c + \mathbf{K}_{d_s}\left(\dot{\mathbf{p}}_c - \dot{\mathbf{p}}_s\right) + \mathbf{K}_{p_s}\left(\mathbf{p}_c - \mathbf{p}_s\right) \tag{3.4}$$

où $\mathbf{K}_{p_s}$ et $\mathbf{K}_{d_s}$ sont les matrices diagonales des gains sur les erreurs de position et de vitesse.

3.1.4 Loi d'impédance

La loi d'impédance permet de moduler la trajectoire désirée en fonction des forces de l'environnement. Pour ce faire, on insère virtuellement un système masse-ressort-amortisseur entre la trajectoire désirée $\mathbf{p}_d$ et la trajectoire corrigée $\mathbf{p}_c$. La Figure 3.3 présente l'interprétation physique de la loi d'impédance.

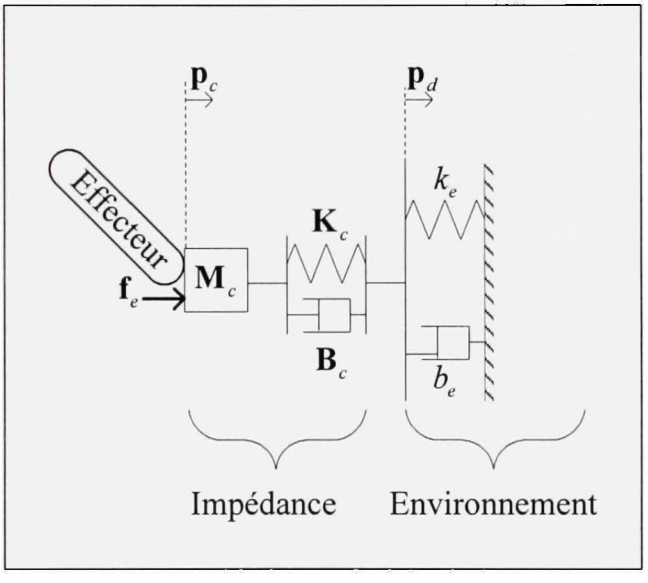

Figure 3.3 ***Interprétation physique de la loi d'impédance.***

On représente l'impédance désirée par l'équation dynamique suivante (Chiaverini, Siciliano et Villani, 1999):

$$\mathbf{M}_c\ddot{\mathbf{\Delta}}_{dc} + \mathbf{B}_c\dot{\mathbf{\Delta}}_{dc} + \mathbf{K}_c\mathbf{\Delta}_{dc} = -\mathbf{f}_e \tag{3.5}$$

où $\mathbf{\Delta}_{dc} = \mathbf{p}_d - \mathbf{p}_c$ est la différence entre la pose désirée et la pose corrigée, $\mathbf{M}_c \in \mathbb{R}^{m\times n}$ est la matrice des masses, $\mathbf{B}_c \in \mathbb{R}^{m\times n}$ est la matrice des constantes d'amortissements et $\mathbf{K}_c \in \mathbb{R}^{m\times n}$ est la matrice des constantes de rigidités. Tel que décrit à la section 2.2, le vecteur des forces de l'environnement $\mathbf{f}_e$ est exprimé dans le repère de l'environnement $\{E\}$. Afin d'être

cohérent avec cette convention, le modèle d'impédance de l'équation (3.5) utilise la valeur négative de $\mathbf{f}_e$ afin que le contact avec l'environnement produise une différence de position $\Delta_{dc} = \mathbf{p}_d - \mathbf{p}_c > 0$. La trajectoire corrigée peut donc être calculée en respectant l'équation dynamique (3.5) :

$$\ddot{\mathbf{p}}_c = \ddot{\mathbf{p}}_d - \mathbf{M}_c^{-1}\left(-\mathbf{f}_e - \mathbf{B}_c\dot{\Delta}_{dc} - \mathbf{K}_c\Delta_{dc}\right) \tag{3.6}$$

La vitesse $\dot{\mathbf{p}}_c$ et la position $\mathbf{p}_c$ corrigées sont calculées par intégration numérique de l'équation (3.6). Étant donné que le modèle de l'environnement ne tient pas compte de la force de friction, l'impédance aura un effet seulement sur l'axe normal à la surface de contact. Ainsi, la loi d'impédance sera décrite uniquement par les paramètres scalaires m_c, b_c, et k_c qui seront définies dans l'axe z_e de la force de contact. De plus, on suppose que les matrices des paramètres d'impédances $\mathbf{M}_c$, $\mathbf{B}_c$ et $\mathbf{K}_c$ sont diagonales, c'est-à-dire que les modèles d'impédance de chacun des axes y et z sont indépendants. Ainsi, les matrices des paramètres sont définies comme suit :

$$\mathbf{M}_c^{-1} = \begin{bmatrix} 0 & 0 \\ 0 & \dfrac{1}{m_c} \end{bmatrix} \tag{3.7}$$

$$\mathbf{B}_c = \begin{bmatrix} 0 & 0 \\ 0 & b_c \end{bmatrix} \tag{3.8}$$

$$\mathbf{K}_c = \begin{bmatrix} 0 & 0 \\ 0 & k_c \end{bmatrix} \tag{3.9}$$

3.2 Dynamique et stabilité du système en boucle fermée

L'objectif de cette section est de déterminer la fonction de transfert en boucle fermée de la simulation idéale et d'en déduire les conditions de stabilité. Ensuite, à partir de ces résultats, on présente la méthode pour choisir les paramètres d'impédance.

3.2.1 Dynamique de l'erreur et stabilité de la boucle interne de position

La première étape consiste à établir la dynamique de l'erreur de la boucle interne de position. On commence par insérer l'équation (2.25) de la dynamique du robot virtuel non contraint dans l'équation (3.2) de la linéarisation dans l'espace articulaire pour obtenir ceci :

$$\begin{aligned} \mathbf{M}_s\mathbf{u} + \mathbf{F}_s - \mathbf{J}_s^T\mathbf{f}_e &= \mathbf{M}_s\ddot{\mathbf{q}}_s + \mathbf{F}_s - \mathbf{J}_s^T\mathbf{f}_e \\ \Rightarrow \mathbf{u} &= \ddot{\mathbf{q}}_s \end{aligned} \tag{3.10}$$

Ensuite, on insère l'équation (3.10) dans l'équation (3.3) de la linéarisation dans l'espace de la tâche et sachant que la dérivée seconde de la pose $\ddot{\mathbf{p}}_s = \mathbf{J}_s\ddot{\mathbf{q}}_s + \dot{\mathbf{J}}_s\dot{\mathbf{q}}_s$, on obtient:

$$\begin{aligned} \ddot{\mathbf{q}}_s &= \mathbf{J}_s^{-1}\left(\mathbf{u}_p - \dot{\mathbf{J}}_s\dot{\mathbf{q}}_s\right) \\ \Rightarrow \mathbf{u}_p &= \mathbf{J}_s\ddot{\mathbf{q}}_s + \dot{\mathbf{J}}_s\dot{\mathbf{q}}_s \\ \Rightarrow \mathbf{u}_p &= \ddot{\mathbf{p}}_s \end{aligned} \tag{3.11}$$

Donc, tel que décrit par (Chiaverini, Siciliano et Villani, 1999), on peut ensuite insérer l'équation (3.11) dans l'équation (3.4) du contrôleur linéaire PD pour obtenir la dynamique de l'erreur $\mathbf{e}_s = \mathbf{p}_c - \mathbf{p}_s$ de la boucle interne de position :

$$\begin{aligned} \ddot{\mathbf{p}}_s &= \ddot{\mathbf{p}}_c + \mathbf{K}_{p_s}\left(\mathbf{p}_c - \mathbf{p}_s\right) + \mathbf{K}_{d_s}\left(\dot{\mathbf{p}}_c - \dot{\mathbf{p}}_s\right) \\ \Rightarrow \ddot{\mathbf{e}}_s &+ \mathbf{K}_{d_s}\dot{\mathbf{e}}_s + \mathbf{K}_{p_s}\mathbf{e}_s = 0 \end{aligned} \tag{3.12}$$

La stabilité exponentielle de l'erreur de suivi est donc assurée lorsque les matrices $\mathbf{K}_{d_s}$ et $\mathbf{K}_{p_s}$ sont symétriques et définies positives.

3.2.2 Fonction de transfert et stabilité du système en boucle fermée

Dans cette section, une analyse du système en boucle fermée dans le domaine de Laplace est présentée. En appliquant la transformée de Laplace sur la dynamique de l'erreur de la boucle interne de position décrit par l'équation (3.12) on obtient (Lawrence, 1988):

$$\left(s^2\mathbf{I}+\mathbf{K}_{d_s}s+\mathbf{K}_{p_s}\right)\left(\mathbf{p}_c-\mathbf{p}_s\right)=0 \tag{3.13}$$

Tel que décrit à la section 3.1.4, $\boldsymbol{\Delta}_{dc}=\mathbf{p}_d-\mathbf{p}_c$ et donc :

$$\mathbf{p}_c=\mathbf{p}_d-\boldsymbol{\Delta}_{dc} \tag{3.14}$$

En appliquant la transformée de Laplace au modèle d'impédance,

$$\boldsymbol{\Delta}_{dc}=-\left(\mathbf{M}_c s^2+\mathbf{B}_c s+\mathbf{K}_c\right)^{-1}\mathbf{f}_e \tag{3.15}$$

En insérant successivement les équations (3.15) dans (3.14) et (3.14) dans (3.13) on obtient le résultat suivant:

$$\left(s^2\mathbf{I}+\mathbf{K}_{d_s}s+\mathbf{K}_{p_s}\right)\left(\mathbf{p}_d+\left(\mathbf{M}_c s^2+\mathbf{B}_c s+\mathbf{K}_c\right)^{-1}\mathbf{f}_e-\mathbf{p}_s\right)=0 \tag{3.16}$$

En reformulant l'équation (3.16) et sachant que l'équation dynamique de l'environnement dans le domaine de Laplace est $\mathbf{f}_e=-\left(\mathbf{B}_e s+\mathbf{K}_e\right)\mathbf{p}_s$ (voir l'équation (2.12)), on trouve la fonction du système en boucle fermée :

$$\mathbf{p}_s=\left(\mathbf{M}_c s^2+\left(\mathbf{B}_c+\mathbf{B}_e\right)s+\mathbf{K}_c+\mathbf{K}_e\right)^{-1}\left(\mathbf{M}_c s^2+\mathbf{B}_c s+\mathbf{K}_c\right)\mathbf{p}_d \tag{3.17}$$

où les matrices $\mathbf{B}_e = \text{diag}(0, b_e)$ et $\mathbf{K}_e = \text{diag}(0, k_e)$ puisque l'environnement est défini uniquement selon l'axe des Z dans le repère de la tâche. Étant donné que le système est multi-variable et est constitué de deux systèmes de $2^{\text{ème}}$ ordre découplés, tous les coefficients des deux polynômes caractéristiques doivent être positifs pour qu'il soit stable. Donc, les conditions de stabilité sont les suivantes :

$$\mathbf{M}_c > 0 \ , \ \mathbf{B}_e + \mathbf{B}_c > 0 \ , \ \mathbf{K}_e + \mathbf{K}_c > 0 \tag{3.18}$$

3.2.3 Choix des paramètres d'impédances

Les paramètres d'impédance $\mathbf{M}_c$, $\mathbf{B}_c$ et $\mathbf{K}_c$ doivent être choisis de façon à respecter un comportement désiré. Premièrement, on pose $\mathbf{M}_c = \text{diag}(0, m_c)$ de façon à obtenir une valeur d'inertie physiquement cohérente. Deuxièmement, on calculera $\mathbf{K}_c = \text{diag}(0, k_c)$ de façon à obtenir une force désirée en régime permanent. Finalement, $\mathbf{B}_c = \text{diag}(0, b_c)$ sera choisi de façon à assurer la stabilité du système ainsi que des résultats de simulation appropriés.

Sachant que le modèle de l'environnement dans le domaine de Laplace est $\mathbf{f}_e = -(\mathbf{B}_e s + \mathbf{K}_e)\mathbf{p}_s$, à partir de (3.17) on obtient :

$$\mathbf{f}_e = -(\mathbf{B}_e s + \mathbf{K}_e)\left(\mathbf{M}_c s^2 + (\mathbf{B}_c + \mathbf{B}_e)s + \mathbf{K}_c + \mathbf{K}_e\right)^{-1}\left(\mathbf{M}_c s^2 + \mathbf{B}_c s + \mathbf{K}_c\right)\mathbf{p}_d \tag{3.19}$$

En appliquant le théorème de la valeur finale (Leigh, 2004) à la fonction de transfert (3.19), on trouve que :

$$\mathbf{f}_{e\infty} = -\mathbf{K}_e\left(\mathbf{K}_c + \mathbf{K}_e\right)^{-1}\mathbf{K}_c\mathbf{p}_{d\infty} = -\mathbf{K}\mathbf{p}_{d\infty} \tag{3.20}$$

où $\mathbf{K}$ représente la rigidité équivalente du système, $\mathbf{p}_{d\infty}$ représente la pose du robot virtuel en régime permanent et, $\mathbf{f}_{e\infty}$ représente le vecteur des forces d'interaction avec l'environnement en régime permanent. Sachant que le modèle de l'environnement est défini uniquement selon l'axe Z du repère de la tâche, en manipulant l'équation (3.20), on obtient la solution suivante pour $\mathbf{K}_c$:

$$\mathbf{K}_c = \begin{bmatrix} 0 & 0 \\ 0 & k_c \end{bmatrix} = \mathbf{K}\left[\mathbf{K}_e - \mathbf{K}\right]^{-1}\mathbf{K}_e = \begin{bmatrix} 0 & 0 \\ 0 & \dfrac{kk_e}{k_e - k} \end{bmatrix} \tag{3.21}$$

où k est la rigidité équivalente selon l'axe z, k_c est la constante de rigidité de l'impédance désirée selon l'axe z et k_e est la constante de rigidité de l'environnement selon l'axe z. Donc, à partir d'une force désirée en régime permanent $f_{e\infty z}$ et de la trajectoire désirée en régime permanent $p_{d\infty z}$, on peut facilement déduire la rigidité équivalente désirée $k = f_{e\infty z} / p_{d\infty z}$. La rigidité de l'impédance désirée peut alors être calculée à l'aide de l'équation (3.21).

Finalement, la constante d'amortissement du modèle d'impédance a été déterminée. En inspectant la fonction de transfert du système en boucle fermée décrit par l'équation (3.17), on remarque que ce paramètre intervient autant au numérateur qu'au dénominateur. Par conséquent, il n'est pas trivial de déduire ce paramètre de façon à imposer des caractéristiques transitoires. Donc, $\mathbf{B}_c = \mathrm{diag}\left(0, b_c\right)$ est choisi par essais et erreurs de façon à obtenir une réponse acceptable tout en respectant le critère de stabilité décrit par (3.18).

Dans ce chapitre, les différents éléments requis pour la simulation idéale ont été présentés. Ils formeront la base de comparaison pour la simulation avec matériel dans la boucle présentée au prochain chapitre. Notamment, il a été montré que le robot virtuel sera simulé par intégration numérique de l'accélération du modèle dynamique non contraint. Aussi, le développement du contrôleur par impédance du robot virtuel a été présenté. Finalement, la

preuve de stabilité du système en boucle fermée a été faite et la méthode de sélection des paramètres d'impédances a été expliquée.

CHAPITRE 4

SIMULATION AVEC MATÉRIEL DANS LA BOUCLE

Le but premier de simuler le robot virtuel est de connaître sa réaction face à un environnement donné. Cependant, le modèle de l'environnement n'est pas toujours connu ou il peut être imprécis. La simulation avec matériel dans la boucle, qui est le sujet du présent chapitre, est utilisée pour résoudre ce problème. Dans ce contexte, l'environnement virtuel est remplacé par un environnement et un robot réels. Pour fins de comparaison, le contrôleur du robot virtuel est identique à celui développé au chapitre précédent. Le schéma de simulation du robot virtuel proposé précédemment et illustré à la Figure 3.1 est alors remplacé par celui de la Figure 4.1.

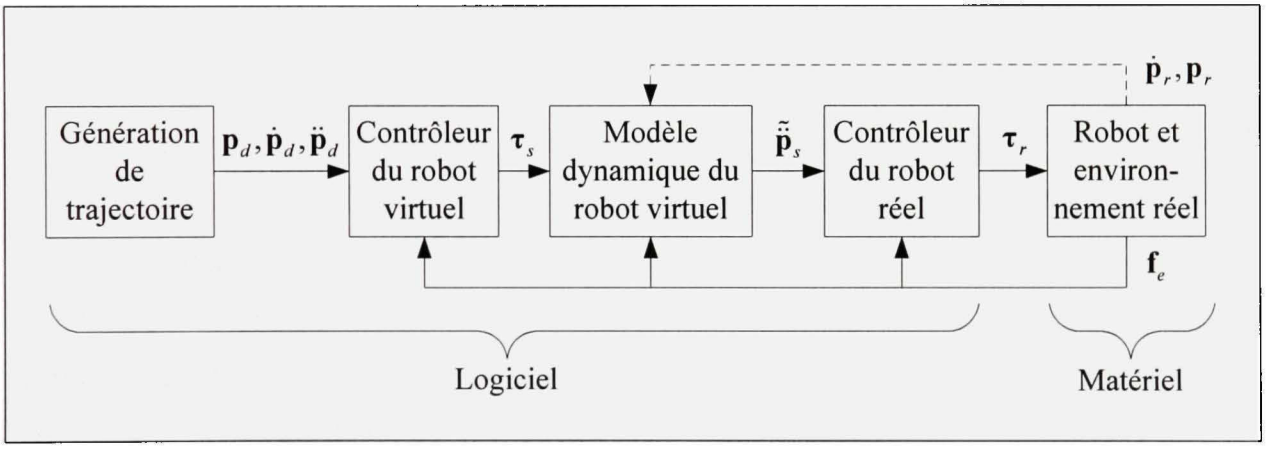

Figure 4.1 ***Schéma de simulation avec matériel dans la boucle.***

Le robot réel doit être commandé de façon à ce qu'il reproduise la dynamique du robot virtuel. Ainsi, tel que représenté à la Figure 4.2, le robot simulé va interagir virtuellement avec l'environnement réel. La réalisation de cet objectif nécessite le développement d'une loi de commande pour le robot réel, qui sera présenté à la section 4.1. Sur la Figure 4.1, on remarque que les mesures de vitesse et de position du robot réel sont requises pour simuler le robot virtuel. Cette rétroaction est requise seulement pour un des deux schémas de simulation qui seront présentés à la section 4.2.

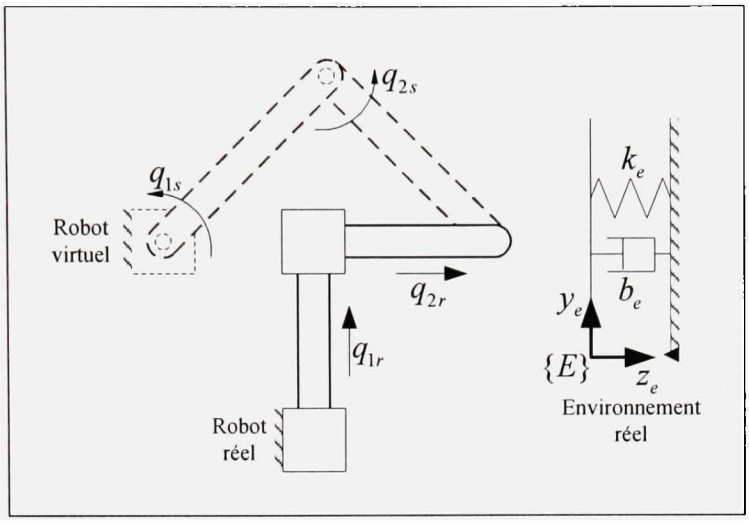

Figure 4.2 ***Schéma du robot réel et du robot virtuel en interaction avec l'environnement réel.***

4.1 Contrôleur du robot réel

Le contrôleur du robot réel utilisé a été initialement proposé par (Aghili et Piedboeuf, 2006). Ce contrôleur a été validé dans un contexte de robotique spatiale qui est significativement différent de celui d'un robot de meulage (voir la section 1.3). Pour cette raison, dans le chapitre 5, cette approche sera validée dans ce nouveau contexte. Mais d'abord, l'approche proposée par (Aghili et Piedboeuf, 2006) est présentée en détails dans le reste de ce chapitre. De plus, une extension de l'approche proposée par (Aghili et Piedboeuf, 2006), qui consiste à incorporé un contrôleur PID dans le schéma de contrôle, sera présentée.

L'objectif formulé par (Aghili et Piedboeuf, 2006) consiste à déterminer la loi de commande de couple à appliquer aux joints du robot réel de façon à ce qu'elle respecte les deux conditions suivantes :

C.1) Le robot réel et le robot virtuel ont la même pose exprimée dans le repère de l'environnement, c'est-à-dire $\mathbf{p}_s = \mathbf{p}_r$;

C.2) Le robot réel et le robot virtuel sont soumis aux mêmes forces d'interaction avec l'environnement.

La condition C.1 est traduite par l'application de la contrainte rhéonomique (2.29), c'est-à-dire :

$$\mathbf{\Phi}\left(\mathbf{q}_s, \mathbf{q}_r(t)\right) = \mathbf{p}_r\left(\mathbf{q}_r(t)\right) - \mathbf{p}_s\left(\mathbf{q}_s\right) = 0 \tag{4.1}$$

La dérivée seconde par rapport au temps de la contrainte (4.1) est obtenue et ré-exprimée grâce à la relation (2.22) :

$$\ddot{\mathbf{\Phi}} = \ddot{\mathbf{p}}_r - \ddot{\mathbf{p}}_s = 0$$

$$\ddot{\mathbf{\Phi}} = \ddot{\mathbf{p}}_r - \left(\mathbf{J}_s\ddot{\mathbf{q}}_s + \dot{\mathbf{J}}_s\dot{\mathbf{q}}_s\right) = 0 \tag{4.2}$$

Ensuite, en isolant l'accélération des coordonnées généralisées du robot virtuel dans (4.2) on obtient :

$$\ddot{\mathbf{q}}_s = \mathbf{J}_s^{-1}\left(\ddot{\mathbf{p}}_r - \dot{\mathbf{J}}_s\dot{\mathbf{q}}_s\right) \tag{4.3}$$

En insérant l'accélération (4.3) dans le modèle dynamique contraint du robot virtuel (2.31), on peut déterminer la solution des multiplicateurs de Lagrange :

$$\mathbf{M}_s\mathbf{J}_s^{-1}\left(\ddot{\mathbf{p}}_r - \dot{\mathbf{J}}_s\dot{\mathbf{q}}_s\right) + \mathbf{F}_s - \mathbf{J}_s^T\boldsymbol{\lambda} = \boldsymbol{\tau}_s$$

$$\Rightarrow \boldsymbol{\lambda} = \mathbf{J}_s^{-T}\left(\mathbf{M}_s\mathbf{J}_s^{-1}\left(\ddot{\mathbf{p}}_r - \dot{\mathbf{J}}_s\dot{\mathbf{q}}_s\right) + \mathbf{F}_s - \boldsymbol{\tau}_s\right) \tag{4.4}$$

Puis, le modèle dynamique du robot réel (2.8) peut être reformulé de la façon suivante :

$$\ddot{\mathbf{q}}_r = \mathbf{M}_r^{-1}\left(\boldsymbol{\tau}_r + \mathbf{J}_r^T\mathbf{f}_e - \mathbf{F}_r\right) \tag{4.5}$$

En se basant sur la méthode du couple pré-calculé dans l'espace de la tâche (Aghili et Piedboeuf, 2006; J. Craig, 2004), on insère (4.5) dans la dérivée seconde de la pose du robot réel $\ddot{\mathbf{p}}_r = \mathbf{J}_r\ddot{\mathbf{q}}_r + \dot{\mathbf{J}}_r\dot{\mathbf{q}}_r$ et on obtient :

$$\ddot{\mathbf{p}}_r = \mathbf{J}_r\mathbf{M}_r^{-1}\left(\boldsymbol{\tau}_r + \mathbf{J}_r^T\mathbf{f}_e - \mathbf{F}_r\right) + \dot{\mathbf{J}}_r\dot{\mathbf{q}}_r \tag{4.6}$$

En imposant $\boldsymbol{\lambda} = \mathbf{f}_e$, afin de respecter la condition C.2, et en insérant (4.6) dans (4.4), le résultat suivant est obtenu :

$$\mathbf{f}_e = \boldsymbol{\lambda} = \mathbf{J}_s^{-T}\left(\mathbf{M}_s\mathbf{J}_s^{-1}\left(\mathbf{J}_r\mathbf{M}_r^{-1}\left(\boldsymbol{\tau}_r + \mathbf{J}_r^T\mathbf{f}_e - \mathbf{F}_r\right) + \dot{\mathbf{J}}_r\dot{\mathbf{q}}_r - \dot{\mathbf{J}}_s\dot{\mathbf{q}}_s\right) + \mathbf{F}_s - \boldsymbol{\tau}_s\right) \tag{4.7}$$

Finalement, on isole $\boldsymbol{\tau}_r$ dans (4.7) afin d'obtenir le couple du robot réel qui respecte les conditions C.1 et C.2. Après simplifications, on obtient que :

$$\boldsymbol{\tau}_r = \mathbf{M}_r\mathbf{J}_r^{-1}\left(\tilde{\ddot{\mathbf{p}}}_\mathbf{s} - \dot{\mathbf{J}}_r\dot{\mathbf{q}}_r\right) + \mathbf{F}_r - \mathbf{J}_r^T\mathbf{f}_e \tag{4.8}$$

où $\tilde{\ddot{\mathbf{p}}}_\mathbf{s} = \mathbf{J}_s\mathbf{M}_s^{-1}\left(\boldsymbol{\tau}_s - \mathbf{F}_s + \mathbf{J}_s^T\mathbf{f}_e\right) + \dot{\mathbf{J}}_s\dot{\mathbf{q}}_s$ représente l'estimation de l'accélération de la pose, c'est-à-dire l'accélération cartésienne dans le repère de la tâche, calculée à partir du modèle non contraint du robot virtuel. Étant donné que cette commande est basée uniquement sur l'accélération, elle va inévitablement dévier. Pour cette raison, dans (Aghili et Piedboeuf, 2006), on propose d'ajouter une commande proportionnel-dérivée (PD) afin d'obtenir la loi de commande suivante :

$$\boldsymbol{\tau}_r = \mathbf{M}_r\mathbf{J}_r^{-1}\left(\mathbf{u}_r - \dot{\mathbf{J}}_r\dot{\mathbf{q}}_r\right) + \mathbf{F}_r - \mathbf{J}_r^T\mathbf{f}_e \tag{4.9}$$

où l'entrée auxiliaire $\mathbf{u}_r$ est définie comme suit :

$$\mathbf{u}_r = \tilde{\ddot{\mathbf{p}}}_s - \mathbf{K}_{d_r}\left(\dot{\mathbf{p}}_r - \tilde{\dot{\mathbf{p}}}_s\right) - \mathbf{K}_{p_r}\left(\mathbf{p}_r - \tilde{\mathbf{p}}_s\right) \tag{4.10}$$

avec $\tilde{\ddot{\mathbf{p}}}_s$ l'estimation de l'accélération de la pose calculée à partir du modèle non contraint du robot virtuel :

$$\tilde{\ddot{\mathbf{p}}}_s = \mathbf{J}_s\mathbf{M}_s^{-1}\left(\boldsymbol{\tau}_s - \mathbf{F}_s + \mathbf{J}_s^T\mathbf{f}_e\right) + \dot{\mathbf{J}}_s\dot{\mathbf{q}}_s \tag{4.11}$$

Dans l'équation (4.10), $\mathbf{K}_{p_r}$ et $\mathbf{K}_{d_r}$ sont respectivement les gains sur les erreurs de position et de vitesse. Il est démontré dans (Aghili et Piedboeuf, 2006) que la dynamique de l'erreur $\ddot{\mathbf{e}}_r = \ddot{\mathbf{p}}_r - \tilde{\ddot{\mathbf{p}}}_s$ sous l'application de cette loi de commande est décrite par :

$$\ddot{\mathbf{e}}_r + \mathbf{K}_{d_r}\dot{\mathbf{e}}_r + \mathbf{K}_{p_r}\mathbf{e}_r = 0 \tag{4.12}$$

Donc, en appliquant la loi de commande représentée par les équations (4.9), (4.10) et (4.11), les conditions C.1 et C.2 citées précédemment sont respectées. Aussi, la stabilité exponentielle de l'erreur de suivi est assurée lorsque les matrices $\mathbf{K}_{d_r}$ et $\mathbf{K}_{p_r}$ sont symétriques et définies positives.

4.1.1 Ajout de l'action intégrale

Dans le cadre de ce mémoire, l'ajout d'une action intégrale au contrôleur du robot réel est proposé. Ceci permettra de réduire les erreurs de suivi du contrôleur dans le cas de la simulation sans application de la contrainte. Donc, dans le cas du PID, l'entrée auxiliaire (4.10) devient :

$$\mathbf{u}_r = \tilde{\ddot{\mathbf{p}}}_s - \mathbf{K}_{d_r}\left(\dot{\mathbf{p}}_r - \tilde{\dot{\mathbf{p}}}_s\right) - \mathbf{K}_{p_r}\left(\mathbf{p}_r - \tilde{\mathbf{p}}_s\right) - \mathbf{K}_{i_r}\int\left(\mathbf{p}_r - \tilde{\mathbf{p}}_s\right) \tag{4.13}$$

La dynamique de l'erreur $\ddot{\mathbf{e}}_r = \ddot{\mathbf{p}}_r - \tilde{\ddot{\mathbf{p}}}_s$, définie précédemment par (4.12) devient :

$$\dddot{\mathbf{e}}_r + \mathbf{K}_{d_r}\ddot{\mathbf{e}}_r + \mathbf{K}_{p_r}\dot{\mathbf{e}}_r + \mathbf{K}_{i_r}\mathbf{e}_r = 0 \tag{4.14}$$

Une condition nécessaire pour assurer la stabilité exponentielle de l'erreur de suivi, mais non suffisante dans le cas du PID, est lorsque les matrices $\mathbf{K}_{d_r}$, $\mathbf{K}_{p_r}$ et $\mathbf{K}_{i_r}$ sont symétriques et définies positives. En supposant que les matrices de gains sont diagonales, les dynamiques d'erreurs sont découplées. Ainsi, la stabilité est assurée lorsque les gains sont choisis de façon à ce que chacune des équations caractéristiques soit un polynôme de Hurwitz[3] (Leigh, 2004).

4.2 Schémas de simulation

Les deux schémas de simulation qui sont proposés dans (Aghili et Piedboeuf, 2006) sont considérés. Dans les deux cas, la loi de commande développée précédemment est utilisée. La différence réside plutôt dans la façon de simuler le robot virtuel.

4.2.1 Simulation avec application de la contrainte

Le premier schéma de simulation (voir Figure 4.3) consiste à appliquer la contrainte rhéonomique au simulateur du robot virtuel, c'est-à-dire que la pose du robot virtuel est synchronisée en temps réel avec celle du robot réel. Cette contrainte est appliquée au niveau des positions et des vitesses, mais aussi au niveau des accélérations via la loi de commande du robot réel. En effet, cette dernière est basée sur la dérivée seconde de la contrainte, soit

[3] Un polynôme ayant des coefficients réels et dont toutes les racines sont à parties réelles négatives est considéré comme un polynôme de Hurwitz.

l'accélération de la pose du robot réel et virtuel. L'objectif est donc de calculer l'état du robot virtuel ($\mathbf{q}_s, \dot{\mathbf{q}}_s$) en fonction des états du robot réel ($\mathbf{q}_r, \dot{\mathbf{q}}_r$).

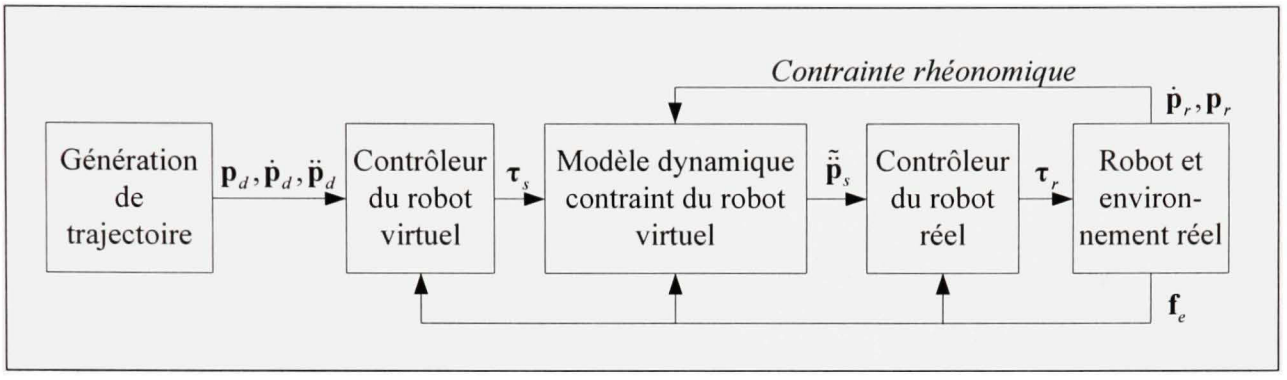

Figure 4.3 ***Schéma de simulation avec application de la contrainte.***

Dans le cadre de ce mémoire, les deux robots possèdent le même nombre de degrés de liberté, donc *n=m*. Ainsi, la simulation du robot virtuel est purement algébrique. Premièrement, $\dot{\mathbf{q}}_s$ peut être déterminé en utilisant la dérivée première de la contrainte (2.29). En d'autres mots, ceci revient à utiliser la cinématique différentielle inverse (2.23) :

$$\begin{aligned} \dot{\mathbf{\Phi}} &= \dot{\mathbf{p}}_s - \dot{\mathbf{p}}_r = 0 \\ \Rightarrow \mathbf{J}_s \dot{\mathbf{q}}_s &= \mathbf{J}_r \dot{\mathbf{q}}_r \\ \Rightarrow \dot{\mathbf{q}}_s &= \mathbf{J}_s^{-1} \mathbf{J}_r \dot{\mathbf{q}}_r \end{aligned} \tag{4.15}$$

Ensuite, sachant que $\mathbf{p}_s(\mathbf{q}_s) = \mathbf{p}_r(\mathbf{q}_r)$, on peut déduire $\mathbf{q}_s$ à partir de la cinématique inverse décrite à la section 2.3.2.

Il est important de remarquer que le couple de commande du robot virtuel n'apparaît pas directement dans les équations permettant de simuler le robot avec l'application de la contrainte. Ce couple de commande τ_s intervient par l'entremise de la contrainte d'accélération (4.2), pour ensuite se retrouver dans la loi de commande du robot réel définie par les équations (4.9) à (4.11).

4.2.1.1 Cas général

Dans le cadre de ce mémoire, le robot virtuel possède le même nombre de degrés de liberté que le robot réel. Ainsi, le robot virtuel est complètement contraint et la simulation de ses états peut être effectuée directement à partir de la cinématique inverse et de la cinématique différentielle inverse. Par contre, dans un cas plus général où le robot virtuel possède plus de degrés de liberté que le robot réel ($n < m$), le robot virtuel n'est que partiellement contraint. Ainsi, on doit simuler les coordonnées indépendantes (non-contraintes) et dépendantes (contraintes) du robot virtuel. En d'autres mots, on doit résoudre le système DAE (« Differential-Algebraic Equations ») formé par les équations (2.31) et (2.32).

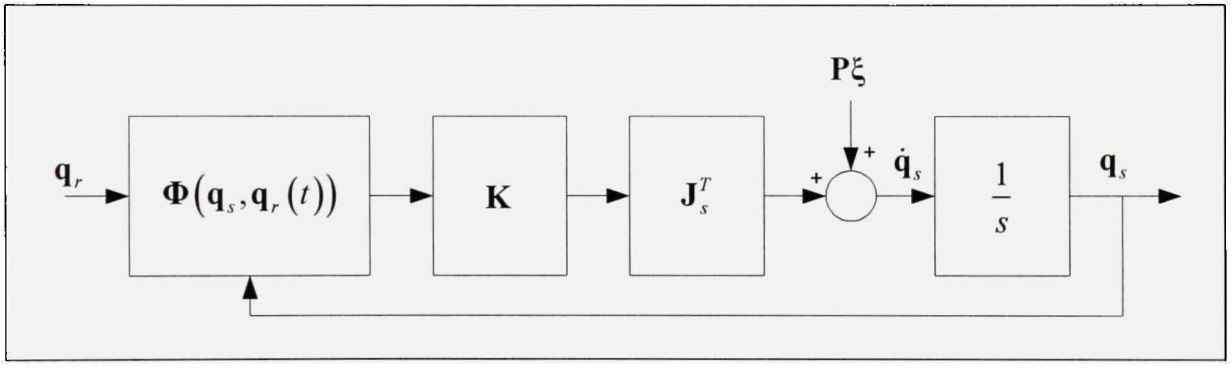

Figure 4.4 ***Schéma de simulation des états dépendants et indépendants du robot virtuel basé sur la méthode CLIK.***

Pour ce faire, (Aghili et Piedboeuf, 2006) propose une solution numérique basée sur la méthode CLIK (Wolovich et Elliott, 1984). À partir du schéma de la Figure 4.4, la solution de $\dot{\mathbf{q}}_s$ est obtenue par :

$$\dot{\mathbf{q}}_s = \mathbf{P}\boldsymbol{\xi} + \mathbf{J}_s^T \mathbf{K} \boldsymbol{\Phi}(\mathbf{q}_s, \mathbf{q}_r) \tag{4.16}$$

où $\mathbf{K}$ est une matrice de gain définie positive, $\mathbf{P}$ est une matrice de projection dont les colonnes annulent la contrainte ($\mathbf{J}_s\mathbf{P} = 0$) et $\boldsymbol{\xi} \in \mathbb{R}^{m-n}$ est le vecteur de vitesse des états

indépendants du robot qui représentent les mouvements non-contraints. Ensuite, l'intégration numérique de $\dot{\mathbf{q}}_s$ permet de déterminer $\mathbf{q}_s$. Il est montré que les états indépendants du robot ξ sont obtenus par intégration numérique de $\dot{\xi}$ qui lui est obtenu en calculant la solution de l'équation dynamique suivante (Aghili et Piedboeuf, 2006) :

$$\mathbf{P}^T\mathbf{M}_s\mathbf{P}\dot{\xi} = \mathbf{P}^T\mathbf{F}_s + \mathbf{P}^T\mathbf{M}_s\mathbf{J}_s^+\left(\ddot{\mathbf{p}}_r - \dot{\mathbf{J}}_s\dot{\mathbf{q}}_s\right) \tag{4.17}$$

où $\mathbf{J}_s^+ = \mathbf{J}_s^T\left(\mathbf{J}_s\mathbf{J}_s^T\right)^{-1}$ est la pseudo-inverse de la matrice jacobienne. En inspectant l'équation (4.17), on remarque le fait important que la mesure de l'accélération de la pose du robot réel doit être disponible pour calculer les états indépendants du robot virtuel.

4.2.2 Simulation sans application de la contrainte

Le second schéma de simulation (voir Figure 4.5) ne fait pas appel à la contrainte rhéonomique.

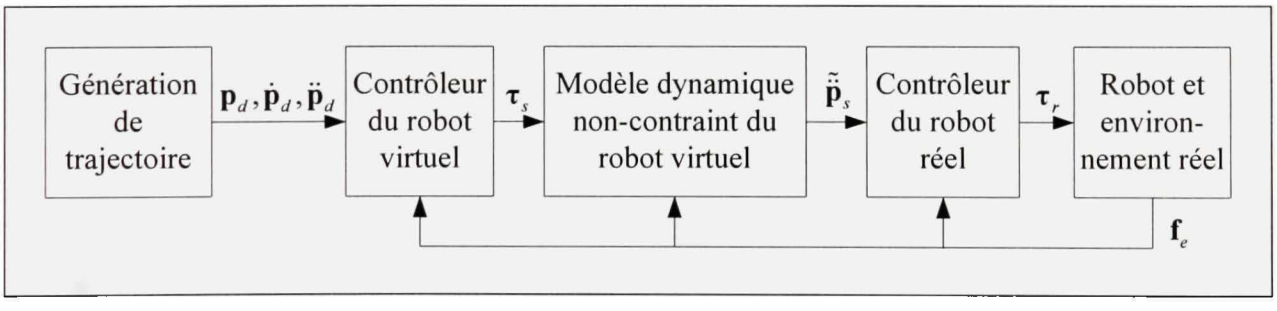

Figure 4.5 ***Schéma de simulation sans application de la contrainte.***

Le modèle dynamique non-contraint (2.25) est utilisé et les états du robot sont obtenus par intégration numérique de l'accélération :

$$\ddot{\mathbf{q}}_s = \mathbf{M}_s^{-1}\left(\boldsymbol{\tau}_s - \mathbf{F}_s + \mathbf{J}_s^T\mathbf{f}_e\right) \tag{4.18}$$

où $\mathbf{f}_e$ représente les forces mesurées par le capteur installé à l'effecteur du robot réel. Il est démontré (Aghili et Piedboeuf, 2006) qu'en appliquant la loi de commande PD décrite à la section 4.1 et en simulant le robot à l'aide de l'équation (4.18), la dynamique de l'erreur de la contrainte est décrite par :

$$\ddot{\mathbf{\Phi}} + \mathbf{K}_{d_r}\dot{\mathbf{\Phi}} + \mathbf{K}_{p_r}\mathbf{\Phi} = 0 \tag{4.19}$$

La stabilité exponentielle de l'erreur sur la contrainte est donc assurée lorsque les matrices $\mathbf{K}_{d_r}$ et $\mathbf{K}_{p_r}$ sont symétriques et définies positives. En appliquant la version PID de la loi de commande, décrite à la section 4.1.1, la dynamique de l'erreur de contrainte (4.19) devient :

$$\dddot{\mathbf{\Phi}} + \mathbf{K}_{d_r}\ddot{\mathbf{\Phi}} + \mathbf{K}_{p_r}\dot{\mathbf{\Phi}} + \mathbf{K}_{i_r}\mathbf{\Phi} = 0 \tag{4.20}$$

Dans le cas du PID, le fait que les matrices $\mathbf{K}_{d_r}$, $\mathbf{K}_{p_r}$ et $\mathbf{K}_{i_r}$ soient symétriques et définies positives n'est pas une condition suffisante pour assurer la stabilité exponentielle de l'erreur sur la contrainte. Étant donné que les matrices de gains sont diagonales, les dynamiques d'erreurs sont découplées. La stabilité est donc assurée lorsque les gains sont choisis de façon à ce que chacune des équations caractéristiques soit un polynôme de Hurwitz (Leigh, 2004).

Dans ce chapitre, la simulation avec matériel dans la boucle a été présentée. Comme dans le cas de la simulation idéale, le robot virtuel est commandé par une loi d'impédance. Cependant, le modèle de l'environnement a été remplacé par un robot et un environnement réels. Ainsi, il n'est pas nécessaire d'avoir un modèle mathématique de l'interaction avec l'environnement pour effectuer la simulation. La loi de commande du robot réel a été développée de façon à ce que la simulation avec matériel dans la boucle soit équivalente à la simulation idéale. Pour ce faire, une contrainte rhéonomique est posée de façon à ce que le robot réel reproduise la dynamique du robot virtuel. Ainsi, le robot simulé interagit virtuellement avec l'environnement réel. Finalement, deux schémas de simulation sont proposés. Le premier utilise le modèle dynamique contraint du robot virtuel et nécessite

l'application de la contrainte rhéonomique. Le second ne fait pas appel à la contrainte, donc il se résume à appliquer la loi de commande sur le robot réel et à simuler le robot virtuel avec l'intégration numérique du modèle dynamique non contraint. Le prochain chapitre présentera les résultats expérimentaux de la simulation avec matériel dans la boucle et les comparera avec les résultats de la simulation idéale.

CHAPITRE 5

RÉSULTATS EXPÉRIMENTAUX

Les résultats expérimentaux ont comme objectif de démontrer la validité de la simulation avec matériel dans la boucle pour simuler des robots en contact avec un environnement réel. À cet effet, un robot réel, muni de moteurs à entraînement direct et en interaction avec un environnement réel très rigide, est utilisé pour reproduire la dynamique d'un robot simulé. De cette façon, le robot simulé va interagir virtuellement avec l'environnement réel. Afin de valider cette approche, l'équivalence entre la simulation avec matériel dans la boucle et la simulation idéale doit être vérifiée. Étant donné que la simulation idéale requiert le modèle de l'environnement, il devra exceptionnellement être modélisé. Cependant, puisque l'objectif futur est justement d'utiliser ce système avec un environnement non modélisable, ce modèle est requis uniquement dans ce contexte de validation. Donc, la simulation idéale présentée au chapitre 3 sera comparée à la simulation avec matériel dans la boucle du chapitre 4. Les bases de cette comparaison seront expliquées en détails dans le plan d'expérimentation de la section 5.7.1.

Avant de présenter les résultats expérimentaux, il est nécessaire de décrire le banc d'essai ainsi que la procédure de prototypage rapide qui a été utilisée pour la mise en œuvre du contrôleur du robot réel. De plus, pour avoir une connaissance la plus exacte que possible du robot réel et de son environnement, une identification doit être effectuée au préalable. Finalement, avant de passer aux résultats proprement dit, il faut aussi présenter le calcul des gains des contrôleurs et des paramètres d'impédance.

5.1 Description du banc d'essai

5.1.1 Montage physique

Le banc d'essai utilisé pour effectuer les expérimentations est composé d'un robot et d'un environnement réel (voir Figure 5.1). L'environnement, considéré comme très rigide, est constitué d'une plaque d'acier coulée dans un bloc de béton.

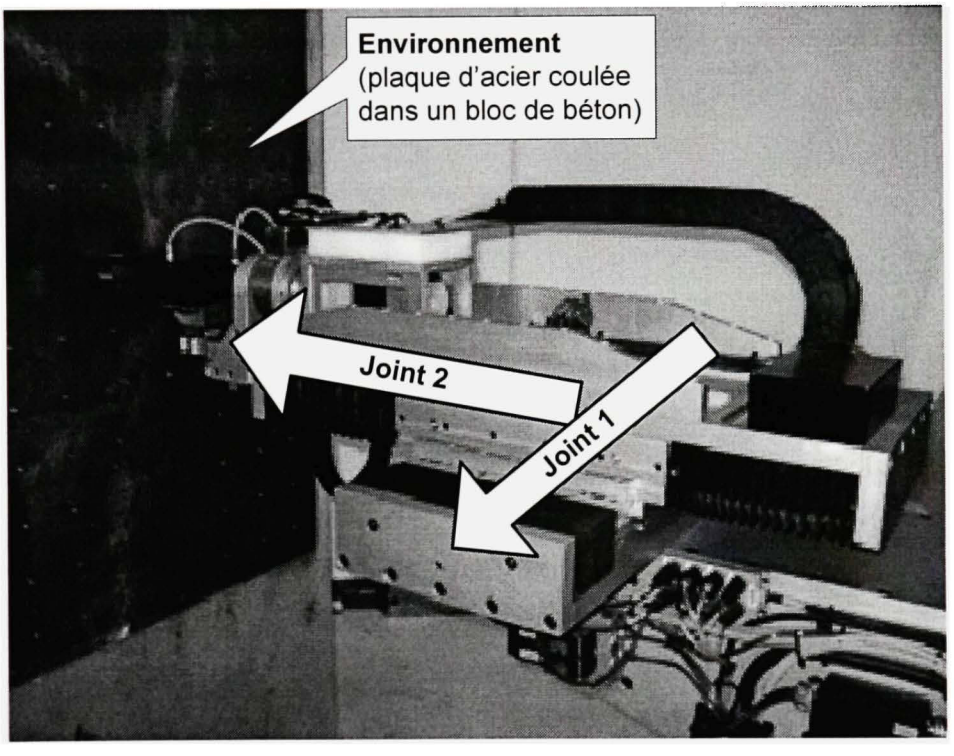

Figure 5.1 ***Photo du robot réel et de l'environnement.***

Le robot est de type sériel et possède six degrés de liberté. Cependant, dans le cadre de ce mémoire, seulement deux des six joints sont utilisés; les autres sont désactivés. Ainsi, tel qu'il a été modélisé à la section 2.1, le banc d'essai est principalement composé d'un robot planaire à deux joints prismatiques à entraînement direct. Les moteurs linéaires qui entraînent les deux joints sont de marque Parker 2000-5 avec des bobines en série. Ces moteurs peuvent produire une force allant jusqu'à 1125N et peuvent maintenir 356N en continu. Ils sont montés sur des guides linéaires à billes de haute performance, donc la friction est minime.

Aussi, un capteur de force de type ATI Delta SI-660-60 est installé sur l'effecteur du robot. Des informations plus détaillés sur le robot sont disponibles dans (Lemieux, Beaudry et Blain, 2006).

5.1.2 Architecture de contrôle

Cette section est consacrée à la description de l'architecture de contrôle du robot, c'est-à-dire la configuration logiciel et matériel qui permettent d'asservir le système.

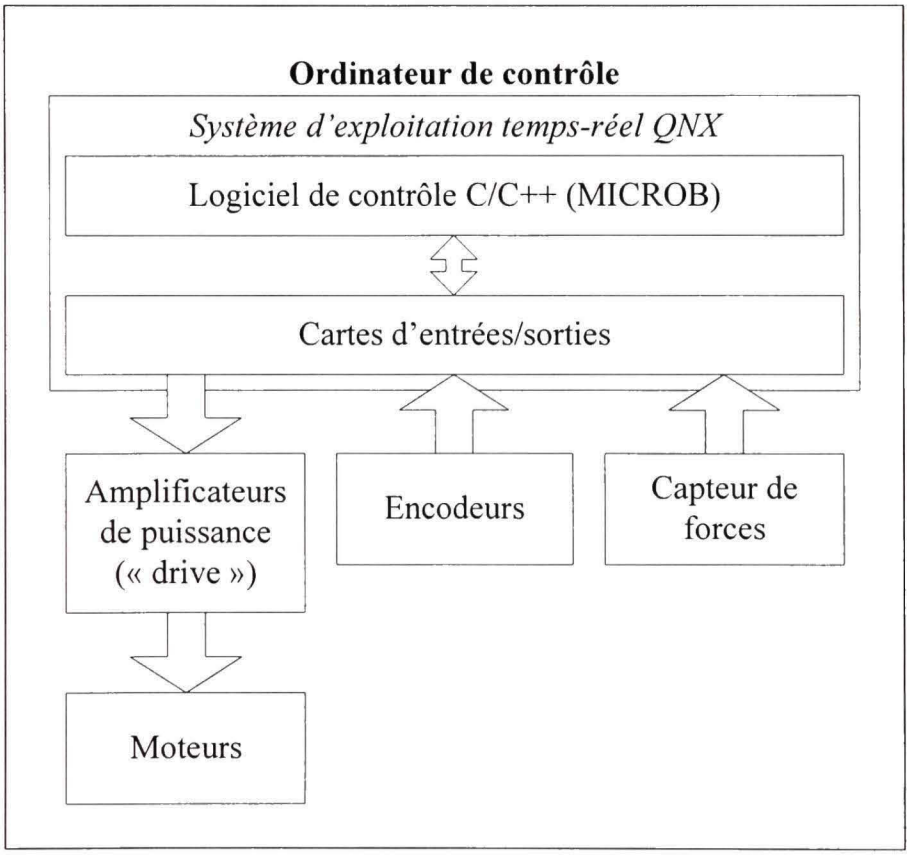

Figure 5.2 ***Architecture de contrôle du robot réel.***

Tel qu'illustré à la Figure 5.2, l'ordinateur de contrôle utilise le système d'exploitation temps-réel QNX (QNX, 2007). Le logiciel de contrôle est programmé en langage C/C++ et utilise la librairie robotique MICROB de l'Institut de Recherche d'Hydro-Québec (IREQ, 2007). Ce dernier assure entre autre la lecture et l'écriture des signaux via les cartes

d'entrées/sorties, l'enregistrement en temps-réel des données et la sécurité du système. Parmi les signaux gérés par le logiciel de contrôle, on compte notamment l'écriture de la commande à envoyer aux amplificateurs de puissance, la lecture des encodeurs et la lecture des signaux du capteur de force.

Pour effectuer la simulation avec matériel dans la boucle, le simulateur du robot virtuel (contrôleur et modèle du robot) ainsi que la loi de commande du robot réel doivent être mises en œuvre dans le logiciel de contrôle du robot réel. Typiquement, chacune de ces équations doivent être traduites en langage C/C++. Dans le cadre de ce mémoire, une procédure complète de prototypage rapide a été développée de façon à éviter entre autre la transcription manuelle de ces équations en langage C/C++. L'architecture de contrôle du robot réel, présentée à la Figure 5.2, sera bonifiée par l'ajout d'une librairie dynamique externe qui contiendra toutes les équations requises à la simulation avec matériel dans la boucle. La section 5.2 présente cette approche automatisée qui permet de générer automatiquement la librairie externe qui sera appelée en temps-réel par le logiciel de contrôle actuel.

La loi de commande du robot réel, qui a été développée à la section 4.1, permet de calculer la commande de force à envoyer aux articulations du robot réel. Or, tel qu'illustré à la Figure 5.2, la commande calculée par le logiciel de contrôle doit passer par une carte d'entrées/sorties, l'amplificateur de puissance et le moteur avant d'être traduite en force appliquée à l'articulation. Ainsi, un gain doit être considéré dans le but de convertir les unités numériques en tension à l'entrée des amplificateurs puis en forces aux articulations. Ce gain devra donc inclure le gain de la carte d'entrées/sorties, de l'amplificateur de puissance et du moteur. La section 5.4.2 présente les détails du calcul et l'identification de ce gain.

5.2 Procédure de prototypage rapide

Afin de réaliser la simulation avec matériel dans la boucle, plusieurs équations doivent être calculées, en particulier les modèles dynamiques et les lois de commande. De plus, la mise en œuvre du logiciel de contrôle du robot réel requiert la traduction de ces équations en langage

de programmation C/C++ (voir Figure 5.2). Cette traduction manuelle est souvent longue et devient une source d'erreur non négligeable. Dans le cadre de ce mémoire, une procédure de prototypage rapide a été développée de façon à minimiser les interventions manuelles.

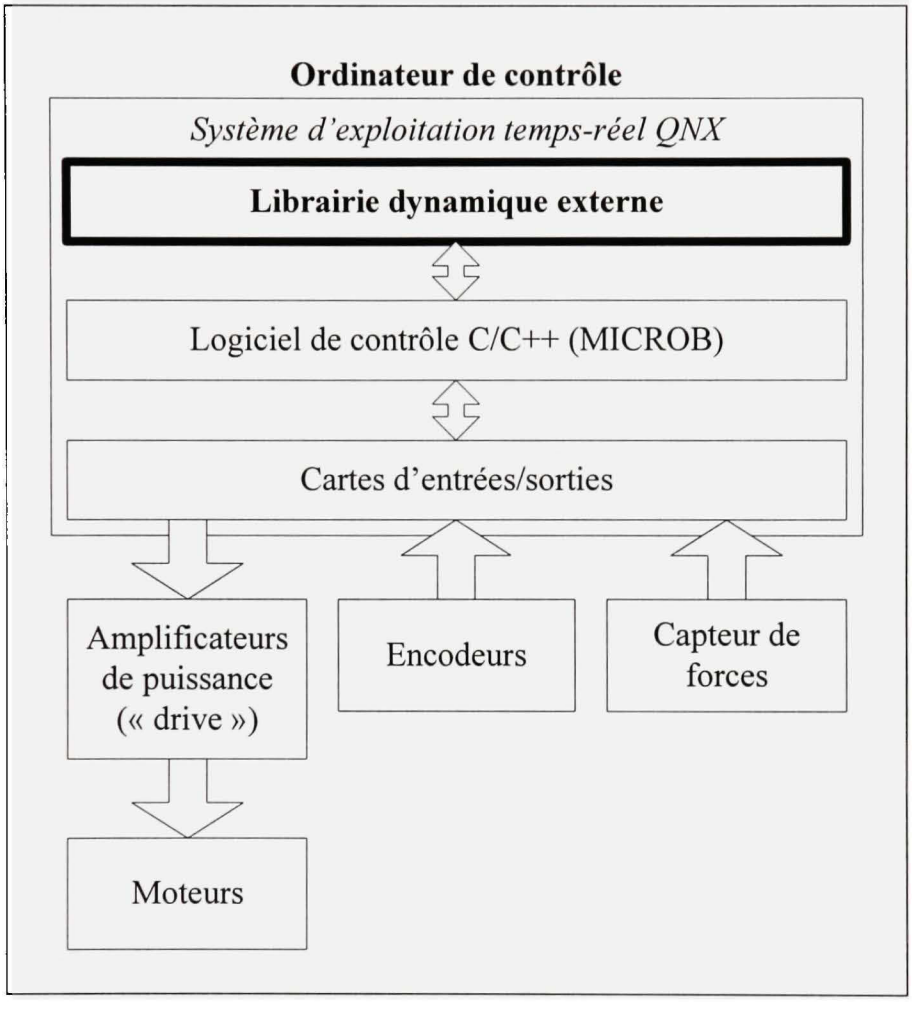

Figure 5.3 ***Architecture de contrôle du robot réel avec procédure de prototypage rapide.***

L'objectif final de cette procédure est d'obtenir une librairie dynamique[4] dans laquelle sera encapsulé un modèle Simulink qui contient toutes les équations requises à la simulation avec matériel dans la boucle (voir Figure 4.1), dont :

- la génération de trajectoire du robot virtuel;
- le contrôleur du robot virtuel;
- le modèle dynamique du robot virtuel;
- la loi de commande du robot réel.

Cette procédure permet non seulement de générer cette librairie à partir d'un modèle Simulink, mais aussi d'automatiser le calcul des équations qui la constitue (comme par exemple les modèles dynamiques et les lois de commande). Tel qu'illustré à la Figure 5.3, une librairie dynamique externe est donc ajoutée à l'architecture de contrôle initiale (voir Figure 5.2) du robot réel. À chaque itération, le logiciel de contrôle du robot réel fera appel à cette librairie externe. De cette façon, le modèle Simulink, c'est-à-dire la librairie dynamique externe, n'interagit pas directement avec les services de bas niveaux du robot. Cette méthode assure la robustesse du système, puisque toutes les tâches critiques sont effectuées par MICROB en langage C/C++ natif à QNX.

Au préalable, les logiciels Matlab et Simulink (MathWorks, 2007) doivent être installés, car ils constituent l'environnement de travail utilisé pour l'ensemble de la procédure. Les modules additionnels suivants doivent également être installés :

- Matlab Symbolic Toolbox (MathWorks, 2007);

---

[4] Une librairie dynamique est un ensemble de fonctions compilées et regroupées dans un même fichier (.dll sous Windows, .so sous QNX). Le terme dynamique signifie que la librairie peut être chargée pendant l'exécution, c'est-à-dire qu'elle n'a pas à être compilée en même temps que le programme qui l'utilise.

- Simulink Real Time Workshop (MathWorks, 2007);
- Robotics Toolbox de Peter Corke (Corke, 1996).

La Figure 5.4 illustre les principales étapes de la procédure afin de générer la librairie dynamique utilisable par le logiciel de contrôle du robot réel. Chacune de ces étapes seront expliquées en détails dans les prochaines sous-sections. De plus, l'ANNEXE II présente un exemple avec un robot planaire à deux degrés de liberté et une commande par couple pré-calculé dans laquelle les lignes ont été numérotées afin d'appuyer l'explication des différentes étapes de la procédure.

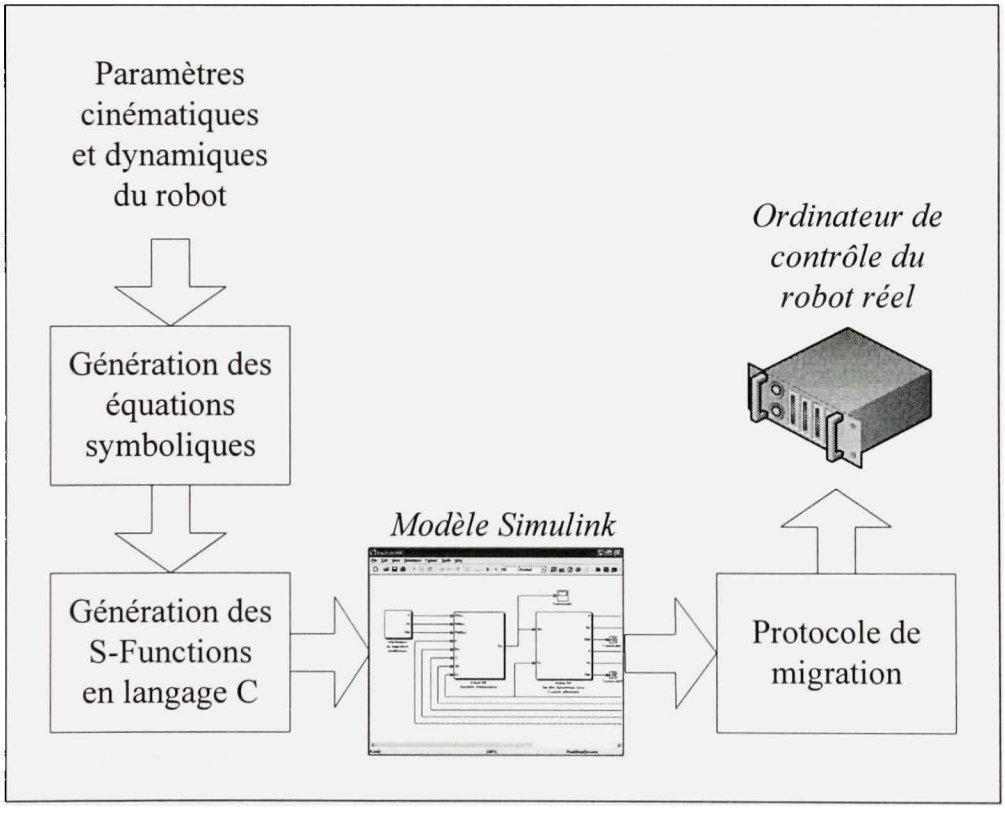

Figure 5.4 ***Schéma bloc de la procédure de prototypage rapide.***

5.2.1 Génération des équations symboliques

Initialement, on considère que les paramètres cinématiques et dynamiques des robots sont connus. Ainsi, cette première étape consiste à générer les équations symboliques requises

pour la simulation à partir de ces paramètres. À ce moment-ci, on ne fait pas encore appel à la structure de la simulation. C'est-à-dire que d'un point de vue schéma bloc, les équations symboliques représentent le contenu des blocs et non la façon dont ils sont interconnectés. Aussi, les opérations génériques comme les gains dans les contrôleurs seront insérées dans le schéma bloc générique de Simulink discuté à la section 5.2.3. Le calcul des équations est effectué à l'aide du « Robotics Toolbox » de Peter Corke (Corke, 1996). À l'origine, cette librairie ne supporte pas les variables symboliques du « Symbolic Toolbox » de Matlab. Donc, quelques modifications mineures ont été faites à la librairie de façon à corriger cette lacune. Cette étape de la procédure se divise en deux sous-étapes : l'insertion des paramètres et le calcul des équations.

Premièrement, les paramètres cinématiques et dynamiques des robots doivent être déclarés en respectant les règles d'utilisation du « Robotics Toolbox » de Peter Corke. En se référant à l'exemple de l'ANNEXE II, les lignes 1 à 73 constituent l'ensemble des déclarations et l'initialisation des paramètres des robots.

Deuxièmement, une fois que tous les paramètres sont déclarés et initialisés, les équations symboliques doivent être calculées. À cet effet, la librairie de Peter Corke offre plusieurs fonctions utiles pour formuler les modèles et les lois de commande. Le Tableau 5.1 présente quelques unes des fonctions disponibles qui sont utilisées dans l'exemple présenté à l'ANNEXE II aux lignes 74 à 138.

Tableau 5.1

Exemples de fonction du Robotics Toolbox de Peter Corke

Fonction	**Description**
fkine	Calcul la cinématique directe
inertia	Calcul la matrice d'inertie
coriolis	Calcul le vecteur des forces de Coriolis et centrifuges
jtraj	Génération de trajectoire dans l'espace articulaire

Par exemple, en combinant des fonctions, on arrive à calculer automatiquement le modèle dynamique d'un robot et la loi de commande linéarisante. Donc, après cette étape, toutes les équations mathématiques requises à la simulation sous forme symbolique dans Matlab sont obtenues.

5.2.2 Génération des S-Functions

Une fois que toutes les équations requises à la simulation sont obtenues sous forme symbolique dans Matlab, elles doivent être transformées en langage C. Plus particulièrement, elles sont converties en S-Functions de façon à être utilisables directement dans Simulink. À cet effet, la fonction Maple `Fonction2sfile` développée pour le cours de Systèmes robotiques en contact (SYS827) de l'ÉTS (Bigras, 2007) est utilisée. Puisque le Symbolic Toolbox de Matlab contient un noyau (« kernel ») Maple, il n'est pas nécessaire d'avoir le logiciel Maple proprement dit. La fonction Matlab `GenSFunction.m` permet de faire le pont entre Matlab et le fichier source Maple `Fonction2sfile.maple`. Le contenu de la fonction ainsi que ses dépendances sont présentés à l'ANNEXE III. L'appel de cette fonction se fait comme suit :

```
params = GenSFunction(expression, name, X)
```

où :

- `expression` est la variable qui contient l'équation sous forme symbolique;
- `name` est le nom qu'on désire donner à la S-Function;
- `X` est le vecteur des variables d'entrées;
- `params` est la variable de retour qui contient la liste des paramètres, c'est-à-dire toutes les variables symboliques, contenues dans l'expression, mais qui ne font pas partie du vecteur des variables à l'entrée X.

Les lignes 143 à 146 de l'ANNEXE II présentent des exemples de génération de S-Functions. Pendant l'exécution de cette fonction, plusieurs fichiers sources en langage C sont

créés. Il faut ensuite compiler ces fichiers sources avec la fonction `mex` de Matlab pour être en mesure d'utiliser les S-Functions dans Simulink. Les lignes 152 à 154 présentent des exemples de compilation de S-Functions. Il est important de remarquer que l'algorithme de génération des S-Functions ajoute le suffixe « ssm » au nom de la fonction et l'appel de la fonction `mex` doit en tenir compte.

5.2.3 Modèle Simulink

À cette étape, toutes les équations symboliques requises à la simulation ont été encapsulées dans des S-Functions, c'est-à-dire que le code en langage C de chacune des équations a été généré et compilé. Il faut maintenant bâtir le modèle Simulink avec la structure de base dans laquelle les S-Functions seront insérées. En effet, puisque les calculs qui dépendent des paramètres cinématiques et dynamiques des robots sont effectués à l'intérieur des S-Functions, le modèle Simulink devient générique. Pour ajouter une S-Function à un modèle Simulink, on doit utiliser le bloc « Simulink/User-Defined Functions/S-Function ». Le Tableau 5.2 présente la description et des exemples pour la configuration des paramètres du bloc.

Tableau 5.2

Description et exemples des paramètres de la S-Function

Nom du paramètre	**Description**	**Exemple**
S-function name	Nom de la S-Function incluant le suffixe « ssm »	`CinDirectessm`
S-function parameters	Vecteur des paramètres constants, soit la valeur de retour `params` de la fonction `GenSFunction`	`[L1 L2]`
S-function modules	Le paramètre par défaut doit rester inchangé.	`''`

Dans le cas particulier de la simulation idéale, il n'est pas nécessaire d'encapsuler tous les sous-systèmes dans une librairie dynamique puisqu'il n'y a pas d'interaction avec une partie matérielle (ex: le robot réel). Ainsi, les étapes subséquentes ne sont pas requises, car le modèle Simulink actuel permet d'effectuer la simulation idéale sous Windows.

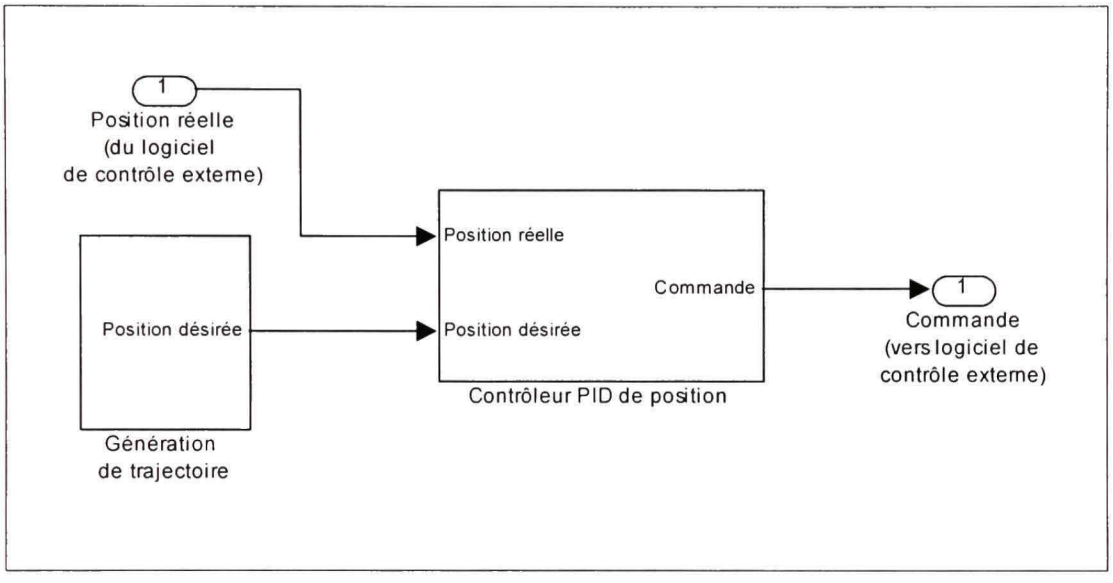

Figure 5.5 ***Exemple simple de modèle Simulink avec une entrée et une sortie externes.***

En ce qui concerne la simulation avec matériel dans la boucle, on doit ajouter des ports d'entrées/sorties externes au schéma Simulink principal. La Figure 5.5 présente un exemple de modèle Simulink dans lequel un port d'entrée et un port de sortie externes ont été ajoutés. L'ajout de ces ports permet au protocole de migration, présenté à la prochaine section, d'identifier les signaux d'entrées/sorties externes qui seront accessibles au logiciel de contrôle du robot réel via la librairie dynamique externe.

5.2.4 Protocole de migration

Le protocole de migration consiste à encapsuler un modèle Simulink dans une librairie dynamique afin d'être en mesure de l'utiliser sur la plate-forme cible, soit l'ordinateur de contrôle du robot réel sur lequel est installé le système d'exploitation QNX. À cet effet, le

protocole de migration développé par Vincent Poiré (Poiré, 2006; Poire et al., 2006) est utilisé. L'utilisation de la cible « Real Time Workshop » fournie par le protocole permet de traduire le modèle Simulink en langage C compatible à QNX et ce, dans une structure de librairie dynamique. Ces fichiers sources sont ensuite copiés et compilés sur l'ordinateur de contrôle du robot réel afin d'obtenir un fichier de librairie dynamique (extension .so sous QNX). Le logiciel de contrôle du robot réel peut donc, à chaque itération, faire appel à cette librairie afin de calculer le modèle de simulation du robot virtuel et la commande du robot réel.

5.3 Identification de l'environnement

L'identification de l'environnement, tel que montré à la Figure 5.1, consiste à déterminer la valeur des paramètres du modèle développé à la section 2.2, soit la constante d'amortissement b_e et la constante de rigidité k_e. Selon l'équation (2.12), la fonction de transfert du modèle de l'environnement est exprimée par :

$$\frac{f_z}{z_r} = -\left(k_e + b_e s\right) \tag{5.1}$$

Ensuite, si on fait l'hypothèse que $\lim_{t\to\infty} z_r$ existe, est borné et différent de zéro, le théorème de la valeur finale permet de trouver que :

$$\frac{f_{z\infty}}{z_{r\infty}} = \lim_{t\to\infty} \frac{f_z}{z_r} = \left.\frac{f_z}{z_r}\right|_{s=0} = -k_e \tag{5.2}$$

où $f_{z\infty}$ et $z_{r\infty}$ représentent respectivement la force et la position mesurées en régime permanent selon l'axe Z du repère de l'environnement. Ainsi, à partir de l'équation (5.2), on obtient la relation suivante qui permet de déterminer la constante de rigidité à partir des valeurs en régime permanent :

$$k_e = -\frac{f_{z\infty}}{z_{r\infty}} \tag{5.3}$$

Afin d'identifier ce paramètre, le robot réel a été positionné à la surface de la plaque et une suite de trois échelons de tension de 0.5, 1.0 et 1.5 Volt a été envoyée à l'entrée de l'amplificateur du moteur du joint 2, c'est-à-dire celui dans l'axe normal de l'environnement. Donc, à l'aide de l'équation (5.3), une constante de rigidité peut être calculée pour chacune des trois réponses échelons en utilisant les valeurs en régime permanent.

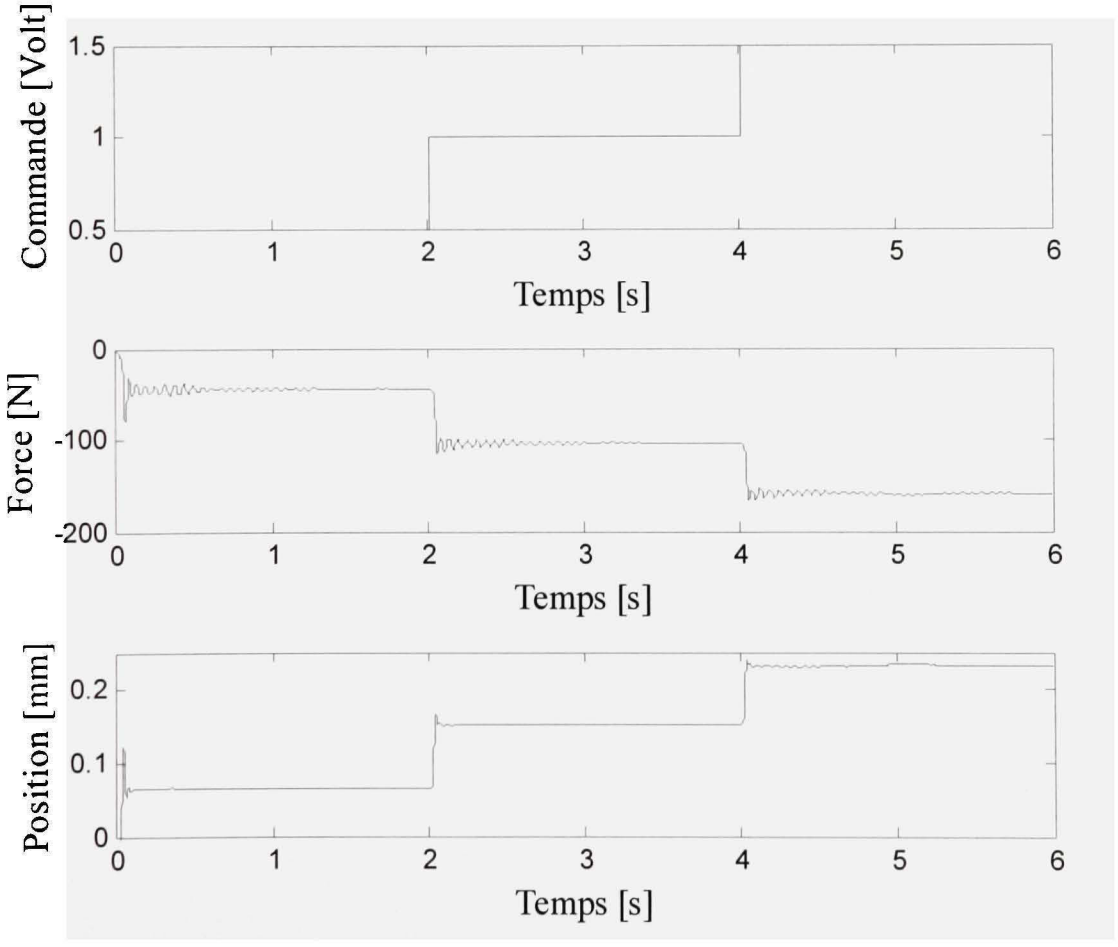

Figure 5.6 ***Réponse échelon pour l'identification de l'environnement.***

Tableau 5.3

Résultats de l'identification de la constante de rigidité de l'environnement

Commande [Volt]	**Force en régime permanent** [N]	**Position en régime permanent** [m]	**Constante de rigidité** [N/m]
0,5	-44,85	6,54E-5	k_{e1}= 685815,7
1,0	-104,36	1,51E-4	k_{e2}= 691608,4
1,5	-159,00	2,31E-4	k_{e3}= 687128,7
	Constante de rigidité moyenne :		k_e= 688184,3

La Figure 5.6 illustre la réponse du système face à ces trois niveaux d'échelon et le

Tableau 5.3 présente un résumé des résultats obtenus. On remarque que les trois constantes de rigidité calculées sont très similaires et donc que l'environnement peut effectivement être représenté par le modèle linéaire (5.1) dans la plage de fonctionnement ciblée. En pratique, la constante de rigidité moyenne suivante a été calculée et sera utilisée pour les simulations :

$$k_e = \frac{k_{e1} + k_{e2} + k_{e3}}{3} = \frac{685815,7 + 691608,4 + 687128,7}{3} = 688184,3 \text{ N/m}$$

Dans le contexte de cette expérimentation, le robot est en contact avec un environnement rigide et donc la vitesse du robot réel est très faible. Ainsi, la constante d'amortissement a une influence peu significative sur le système et la précision des signaux n'est pas suffisante pour l'estimer convenablement. Cette valeur n'a donc pas été identifiée en expérimentation, mais plutôt imposée de façon à obtenir une réponse en boucle fermée bien amortie:

$$b_e = \frac{k_e}{200} = \frac{688184,3}{200} = 3440,9 \text{ N/m/s}$$

5.4 Identification du robot réel

5.4.1 Identification des masses des membrures

Le modèle dynamique du robot réel, tel qu'il a été développé à la section 2.1.5, dépend de la masse des membrures 1 et 2. La masse de la membrure 1 a été déterminée à partir du modèle théorique 3D du robot qui a été dessiné dans le logiciel Catia V5. Pour ce qui est de la masse de la membrure 2, la valeur théorique donnée par le logiciel de dessin a été additionnée à la masse de l'outil, qui a été pesé à l'aide d'une balance. Les valeurs suivantes ont été obtenues :

$$m_{1s} = 26.7 \text{ kg}, \; m_{2s} = 58.6 \text{ kg}$$

5.4.2 Identification du gain des actionneurs

Le contrôleur du robot réel suppose que la commande envoyée au robot est en Newton. Or, le robot est actionné par des moteurs linéaires qui eux sont contrôlés par un amplificateur de puissance. Par conséquent, il est nécessaire de déterminer le gain qui permet de convertir une commande τ' en force τ_r sur la membrure et vice-versa. La Figure 5.7 illustre le schéma bloc de l'actionneur pour un des deux joints.

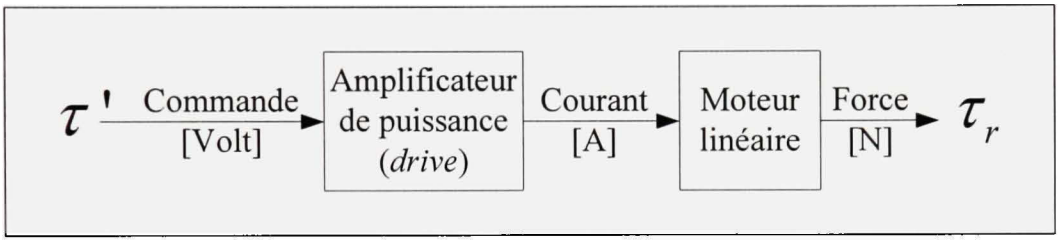

Figure 5.7 ***Schéma bloc de l'actionneur d'un joint du robot réel.***

L'objectif est donc d'identifier le gain k_a de l'actionneur :

$$k_a\ [\text{N/Volt}] = k_d[\text{A/Volt}] \times k_m[\text{N/A}]$$

où k_d est le gain de l'amplificateur de puissance et k_m le gain du moteur. Étant donné que les deux joints sont actionnés par le même type de moteur et d'amplificateur de puissance, un seul gain sera identifié pour les deux. L'identification du gain de l'actionneur est effectuée à partir des mêmes résultats (voir Figure 5.6) qui ont servis à l'identification de l'environnement. Pour chaque niveau de tension, le gain du moteur a été calculé et la moyenne de ces trois gains est utilisée. Ainsi, le gain de l'amplificateur peut être calculé comme suit :

$$k_{a1} = \frac{-f_{\infty 1}}{\tau'_1} = \frac{44.85\ \text{N}}{0.5\ \text{Volt}} = 89.7\ \text{N/Volt}$$

$$k_{a2} = \frac{-f_{\infty 2}}{\tau'_2} = \frac{104.36 \text{ N}}{1.0 \text{ Volt}} = 104.36 \text{ N/Volt}$$

$$k_{a3} = \frac{-f_{\infty 3}}{\tau'_3} = \frac{159.00 \text{ N}}{1.5 \text{ Volt}} = 106 \text{ N/Volt}$$

$$k_a = \frac{k_{a1} + k_{a2} + k_{a3}}{3} = 100.02 \text{ N/Volt}$$

où f_∞ est la force mesurée en régime permanent et τ' est la commande échelon envoyée à l'amplificateur de puissance. Étant donné que le gain du moteur varie très peu en fonction de la tension appliquée, et bien il a pu être identifié à partir d'une seule réponse à l'échelon. La commande de force τ_r, calculée à partir du contrôleur du robot réel, devra donc être transformée de la façon suivante avant d'être envoyée à l'amplificateur de puissance :

$$\tau' \text{ [Volt]} = \frac{\tau_s \text{ [N]}}{k_a \text{ [N/Volt]}} = 0.01\,\tau_s$$

5.5 Paramètres du contrôleur d'impédance

5.5.1 Gains du contrôleur d'impédance

Les gains $\mathbf{K}_{d_s}$ et $\mathbf{K}_{p_s}$ du contrôleur d'impédance du robot virtuel doivent être déterminés. Pour ce faire, les valeurs propres de la dynamique de l'erreur (3.12) seront imposées. En utilisant des valeurs propres réelles et négatives, on s'assure que le système est stable. La représentation d'état de cette équation dynamique peut s'écrire comme suit :

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} = \begin{bmatrix} 0 & 1 \\ -\mathbf{K}_{p_s} & -\mathbf{K}_{d_s} \end{bmatrix} \mathbf{x} \tag{5.4}$$

où $\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 \end{bmatrix}^T = \begin{bmatrix} \mathbf{e}_s & \dot{\mathbf{e}}_s \end{bmatrix}^T$ est le vecteur des états du système et $\mathbf{A}$ est la matrice de la dynamique. Les valeurs propres du système solutionnent l'équation caractéristique suivante :

$$\det\left(\mathbf{A}-\boldsymbol{\lambda}\mathbf{I}\right)=\boldsymbol{\lambda}^2+\mathbf{K}_{d_s}\boldsymbol{\lambda}+\mathbf{K}_{p_s}=0 \tag{5.5}$$

En imposant une valeur propre double λ_0 on obtient l'équation caractéristique désirée suivante :

$$\left(\boldsymbol{\lambda}-\boldsymbol{\lambda}_0\right)^2=\boldsymbol{\lambda}^2-2\boldsymbol{\lambda}\boldsymbol{\lambda}_0+\boldsymbol{\lambda}_0^2 \tag{5.6}$$

L'identification terme à terme de (5.5) avec (5.6) donne :

$$\mathbf{K}_{p_s}=\boldsymbol{\lambda}_0^2 \tag{5.7}$$

$$\mathbf{K}_{d_s}=-2\boldsymbol{\lambda}_0 \tag{5.8}$$

Ainsi, puisque le temps de réponse à 5% est lié à la double valeur propre λ_0 par la relation $T_r=-4,73/\lambda_0$, en imposant toutes les valeurs propres à -47.30, on obtient un temps de réponse de 100ms et l'ensemble de gains suivants :

$$\mathbf{K}_{d_s}=\begin{bmatrix}2237.3 & 0\\ 0 & 2237.3\end{bmatrix},\ \mathbf{K}_{p_s}=\begin{bmatrix}94.6 & 0\\ 0 & 94.6\end{bmatrix}$$

La valeur de 100ms a été choisie par essai erreur de façon à obtenir des performances suffisantes dans notre contexte d'expérimentation.

5.5.2 Paramètres de l'impédance désirée

Comme il a été présenté à la section 3.1.4, les paramètres de l'impédance désirée sont déterminés en fonction de la position finale p_∞, de la force finale f_∞ et de la constante de rigidité de l'environnement k_e. Dans le cadre de ce mémoire, la position finale $p_\infty=3\text{mm}$ et

la force finale $f_\infty = -100\text{N}$ sont imposées. De plus, la constante de rigidité de l'environnement $k_e = 688184,3\ \text{N/m}$ est connue. Ainsi, la constante de rigidité de l'impédance désirée peut être calculée à partir de (3.21) :

$$k_c = \frac{-f_\infty k_e}{k_e p_\infty - f_\infty} = 31793.4\ \text{N/m} \tag{5.9}$$

Ensuite, la masse de l'impédance désirée est imposée à $m_c = 100\text{kg}$. Cette valeur peut paraître élevée à priori, mais elle permet d'obtenir un comportement amorti avec un environnement très rigide. Finalement, la constante d'amortissement a été choisie expérimentalement de façon à obtenir un comportement stable et amorti :

$$b_c = 3757.3\ \text{N/m/s} \tag{5.10}$$

5.6 Gains du contrôleur du robot réel

La loi de commande du robot réel a été calculée dans un premier temps avec une commande linéaire PD. Ensuite, une version PID a été proposée avec l'ajout de l'action intégrale. Ainsi, les gains doivent être déterminés indépendamment pour chacune de ces configurations.

5.6.1 Commande linéaire PD

Dans le cas de la commande linéaire PD, les gains $\mathbf{K}_{d_r}$ et $\mathbf{K}_{p_r}$ doivent être calculés. De façon analogue à ce qui a été fait pour les gains du contrôleur du robot virtuel, les valeurs propres de la dynamique de l'erreur (4.12) seront imposées. La représentation d'état de cette équation dynamique peut s'écrire comme suit :

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} = \begin{bmatrix} 0 & 1 \\ -\mathbf{K}_{p_r} & -\mathbf{K}_{d_r} \end{bmatrix} \mathbf{x} \tag{5.11}$$

où $\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 \end{bmatrix}^T = \begin{bmatrix} \mathbf{e}_r & \dot{\mathbf{e}}_r \end{bmatrix}^T$ est le vecteur d'état du système et $\mathbf{A}$ est la matrice d'état. Les valeurs propres du système solutionnent l'équation caractéristique suivante :

$$\det\left(\mathbf{A} - \boldsymbol{\lambda}\mathbf{I}\right) = \boldsymbol{\lambda}^2 + \mathbf{K}_{d_r}\boldsymbol{\lambda} + \mathbf{K}_{p_r} = 0 \tag{5.12}$$

En imposant une valeur propre double λ_0 on obtient l'équation caractéristique désirée suivante :

$$\left(\boldsymbol{\lambda} - \boldsymbol{\lambda}_0\right)^2 = \boldsymbol{\lambda}^2 - 2\boldsymbol{\lambda}\boldsymbol{\lambda}_0 + \boldsymbol{\lambda}_0^2 \tag{5.13}$$

L'identification terme à terme de (5.11) avec (5.13) permet de trouver que :

$$\mathbf{K}_{p_r} = \boldsymbol{\lambda}_0^2 \tag{5.14}$$

$$\mathbf{K}_{d_r} = -2\boldsymbol{\lambda}_0 \tag{5.15}$$

Donc, sachant que le temps de réponse à 5% est lié à la double valeur propre λ_0 par la relation $T_r = -4,73/\lambda_0$, on obtient un temps de réponse de 125ms en imposant toutes les valeurs propres à -37.84 et les gains obtenus sont :

$$\mathbf{K}_{p_r} = \begin{bmatrix} 1431.87 & 0 \\ 0 & 1431.87 \end{bmatrix}, \mathbf{K}_{d_r} = \begin{bmatrix} 75.68 & 0 \\ 0 & 75.68 \end{bmatrix}$$

Le temps de réponse de la loi de commande d'impédance a été sélectionné par essai erreur. Pour la loi de commande du robot réel, il n'est pas recommandé de choisir n'importe quel temps de réponse, à défaut d'avoir des problèmes de stabilité lors de l'implantation sur le banc d'essai. Le choix du temps de réponse de 125ms a permis d'obtenir un bon compromis entre la stabilité et les performances obtenues.

5.6.2 Commande linéaire PID

Comme dans le cas des gains du contrôleur du robot virtuel et de la version PD du contrôleur du robot réel, la méthode d'imposition des valeurs propres a été utilisée pour calculer les gains $\mathbf{K}_{d_r}$, $\mathbf{K}_{p_r}$ et $\mathbf{K}_{i_r}$. Cependant, l'ajout de l'action intégrale à la partie linéaire de la commande du robot réel va donner un résultat différent de ce que qu'il a été obtenu précédemment, car une troisième variable sera ajoutée à l'équation d'état. La dynamique de l'erreur représentée par l'équation (4.14) peut s'exprimer sous la représentation d'état suivante :

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -\mathbf{K}_{i_r} & -\mathbf{K}_{p_r} & -\mathbf{K}_{d_r} \end{bmatrix} \mathbf{x} \tag{5.16}$$

où $\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 \end{bmatrix}^T = \begin{bmatrix} \mathbf{e}_r & \dot{\mathbf{e}}_r & \ddot{\mathbf{e}}_r \end{bmatrix}^T$ est le vecteur d'état du système et $\mathbf{A}$ est la matrice d'état. Les valeurs propres du système solutionnent l'équation caractéristique suivante :

$$\det\left(\mathbf{A} - \boldsymbol{\lambda}\mathbf{I}\right) = \boldsymbol{\lambda}^3 + \mathbf{K}_{d_r}\boldsymbol{\lambda}^2 + \mathbf{K}_{p_r}\boldsymbol{\lambda} + \mathbf{K}_{i_r} = 0 \tag{5.17}$$

En imposant une valeur propre triple λ_0, on obtient l'équation caractéristique désirée suivante :

$$\left(\boldsymbol{\lambda} - \boldsymbol{\lambda}_0\right)^3 = \boldsymbol{\lambda}^3 - 3\boldsymbol{\lambda}_0\boldsymbol{\lambda}^2 + 3\boldsymbol{\lambda}_0^2\boldsymbol{\lambda} - \boldsymbol{\lambda}_0^3 \tag{5.18}$$

L'identification terme à terme de (5.17) avec (5.18) permet de trouver :

$$\mathbf{K}_{p_r} = 3\boldsymbol{\lambda}_0^2 \tag{5.19}$$

$$\mathbf{K}_{d_r} = -3\boldsymbol{\lambda}_0 \tag{5.20}$$

$$\mathbf{K}_{i_r} = -\boldsymbol{\lambda}_0^3 \tag{5.21}$$

Ainsi, puisque le temps de réponse à 5% est lié à la triple valeur propre λ_0 par la relation $T_r = -6,27/\lambda_0$, en imposant toutes les valeurs propres à -50.16, on obtient un temps de réponse de 125ms et les gains suivants :

$$\mathbf{K}_{d_r} = \begin{bmatrix} 150.48 & 0 \\ 0 & 150.48 \end{bmatrix}, \mathbf{K}_{p_r} = \begin{bmatrix} 7548.08 & 0 \\ 0 & 7548.08 \end{bmatrix}, \mathbf{K}_{i_r} = \begin{bmatrix} 126203.84 & 0 \\ 0 & 126203.84 \end{bmatrix}$$

Tel qu'expliqué à la section 5.6.1, le choix du temps de réponse de 125ms a permis d'obtenir un bon compromis entre la stabilité du système réel et sa performance.

5.7 Simulation avec matériel dans la boucle

5.7.1 Plan d'expérimentation

Les résultats expérimentaux de la simulation avec matériel dans la boucle permettront de vérifier la validité de la méthode pour simuler un robot virtuel en interaction avec un environnement réel. Pour ce faire, les résultats de la simulation avec matériel dans la boucle seront comparés avec ceux de la simulation idéale. Dans le cadre de ce mémoire, deux schémas de simulation ont été présentés : celui avec et celui sans application de la contrainte rhéonomique (voir sections 4.2.1 et 4.2.2). De plus, dans le cas de la méthode sans application de la contrainte, l'ajout d'une action intégrale a été proposé pour améliorer les performances. Ainsi, l'étude expérimentale permettra de comparer les trois cas suivants :

1. Simulation avec application de la contrainte et la version PD du contrôleur du robot réel;
2. Simulation sans application de la contrainte et la version PD du contrôleur du robot réel;

3. Simulation sans application de la contrainte et la version PID du contrôleur du robot réel.

Le robot virtuel est commandé par une loi d'impédance ayant une double rétroaction, c'est-à-dire une boucle externe à rétroaction de force accompagnée d'une boucle interne à rétroaction de position. Par conséquent, ces deux mesures serviront à l'évaluation des performances de la simulation avec matériel dans la boucle. De plus, étant donné que le système est en boucle fermée, les couples de commande du robot virtuel sont ajoutés comme base de comparaison afin de démontrer l'équivalence dynamique. Ainsi, on dira que la simulation avec matériel dans la boucle est équivalente à la simulation idéale si les sorties (position du robot virtuel et force de contact avec l'environnement) et l'entrée (commande envoyée au robot virtuel) du système sont similaires. À partir de l'évaluation de ces critères, on pourra ensuite déterminer laquelle des trois méthodes proposées ci-haut est la plus performante dans ce contexte.

Pour que cette comparaison soit possible, ces trois méthodes doivent être étudiées dans un même contexte de simulation. Premièrement, le robot virtuel est contrôlé par la loi de commande d'impédance présentée à la section 3.1 avec les paramètres (gains et impédance désirée) calculés à la section 5.5. Deuxièmement, la trajectoire désirée, illustrée à la Figure 5.8, est commune aux trois cas de simulation. Elle est constituée d'une trajectoire linéaire, où la première seconde est dans l'espace libre, suivi d'une partie sinusoïdale pour mettre à l'épreuve le système. Le contact du robot avec l'environnement devrait donc se faire au temps $t = 1$ seconde. Finalement, même si les gains du contrôleur du robot réel sont différents selon le type de commande linéaire utilisé (version PD ou PID), on imposera toujours un temps de réponse de 125ms à l'aide des valeurs propres multiples.

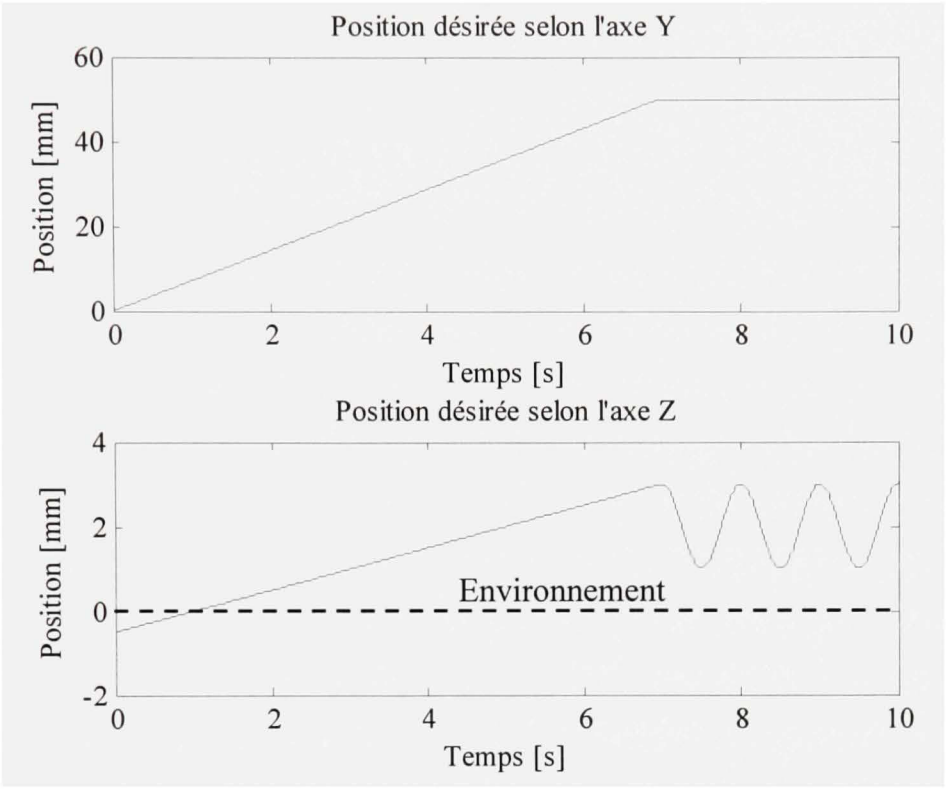

Figure 5.8 ***Position désirée du robot virtuel.***

5.7.2 Simulation avec application de la contrainte

Cette section présente les résultats expérimentaux de la simulation avec matériel dans la boucle d'un robot virtuel selon la méthode présentée à la section 4.2.1, soit avec l'application de la contrainte rhéonomique. Ainsi, à chaque itération, la position et la vitesse du robot virtuel sont synchronisées avec la position et la vitesse mesurées du robot réel. Les vitesses généralisées du robot virtuel sont obtenues par l'équation de cinématique différentielle inverse (4.15). Sachant que la contrainte impose $\mathbf{p}_s(\mathbf{q}_s) = \mathbf{p}_r(\mathbf{q}_r)$, les coordonnées généralisées du robot virtuel peuvent être obtenues en calculant la cinématique inverse présentée à la section 2.3.2.

Le robot virtuel est commandé par la loi d'impédance présentée à la section 3.1 avec les paramètres (gains et impédance désirée) calculés à la section 5.5. La trajectoire désirée du robot virtuel est illustrée à la Figure 5.8. Le robot réel est contrôlé par la loi de commande

présentée à la section 4.1, c'est-à-dire la version sans action intégrale. Les gains de ce contrôleur sont calculés à la section 5.6.1 et ont été obtenus en imposant un temps de réponse de 125ms à l'aide d'une valeur propre double.

Dans le cas de la simulation avec application de la contrainte, une oscillation est présente. En effet, la Figure 5.9 montre que, pendant la partie linéaire de la trajectoire, la force de contact oscille avec une amplitude d'environ 8N. Cette erreur entre la force obtenue par la simulation idéale et celle obtenue par la simulation avec matériel dans la boucle est causée principalement par la friction de coulomb des actionneurs. Ce phénomène est beaucoup moins présent dans la partie sinusoïdale de la trajectoire, c'est-à-dire où les vitesses sont plus élevées. Ceci peut s'expliquer en partie par l'importance relative de la force de frottement par rapport à la force nécessaire pour assurer le suivi de la trajectoire. En effet, la force de frottement de Coulomb varie avec le signe de la vitesse, de sorte qu'elle n'est pas influencée par son amplitude tandis que l'amplitude de la commande de suivi de trajectoire est directement liée à l'amplitude de la trajectoire. De plus, puisque la friction de Coulomb s'oppose de façon constante au mouvement, elle aide l'actionneur à stopper le mouvement avant d'aller en direction inverse, c'est-à-dire à chaque crête négative et positive du sinus. Malgré cette variation de force et étant donné que la force finale désirée est de 100N, on peut dire que l'erreur est acceptable et que la force de contact est suffisamment bien reproduite dans la simulation avec matériel dans la boucle.

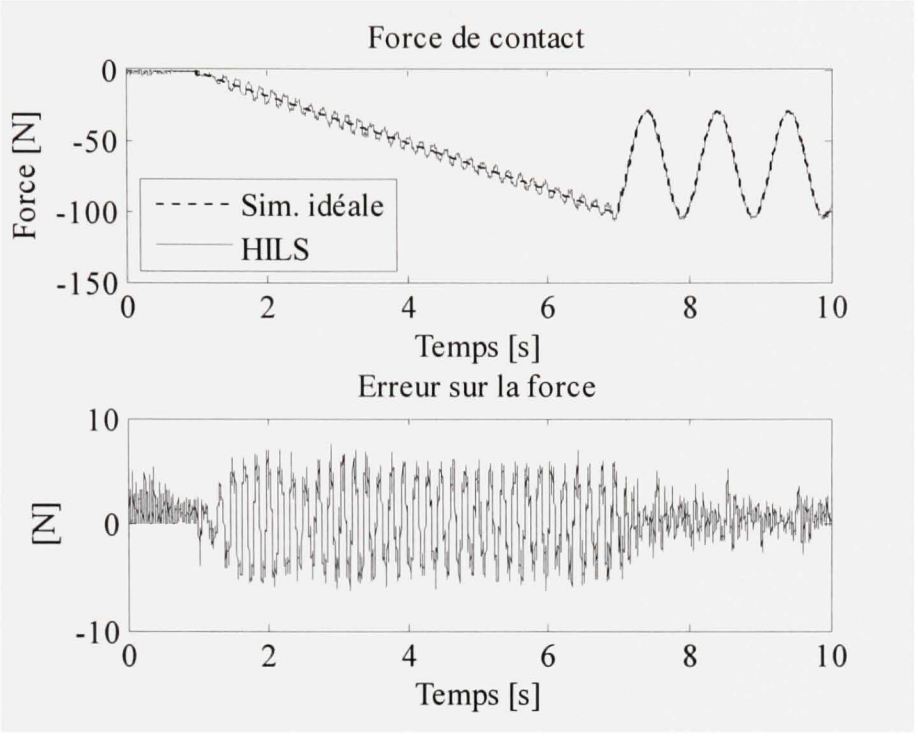

Figure 5.9 ***Force de contact de la simulation avec application de la contrainte.***

La seconde mesure de performance de la simulation avec matériel dans la boucle consiste à comparer la position du robot virtuel avec celle de la simulation idéale. La Figure 5.10 et la Figure 5.11 représentent respectivement la position du robot virtuel selon les axes Y et Z dans le repère de la tâche. Le contact avec l'environnement selon l'axe Z produit une force de friction sur l'axe Y du robot. Même si cette friction n'est pas compensée, le contrôleur du robot réel réussi à suivre avec une grande précision sa trajectoire selon cet axe. Cette précision est assurée par les gains relativement élevés du contrôleur de position du robot réel, calculés à la section 5.6.1. Dans les parties discontinues de la trajectoire (au niveau des vitesses), c'est-à-dire lors du départ et de la transition à la $7^{ème}$ seconde, l'erreur selon l'axe Y atteint un maximum de 15µm.

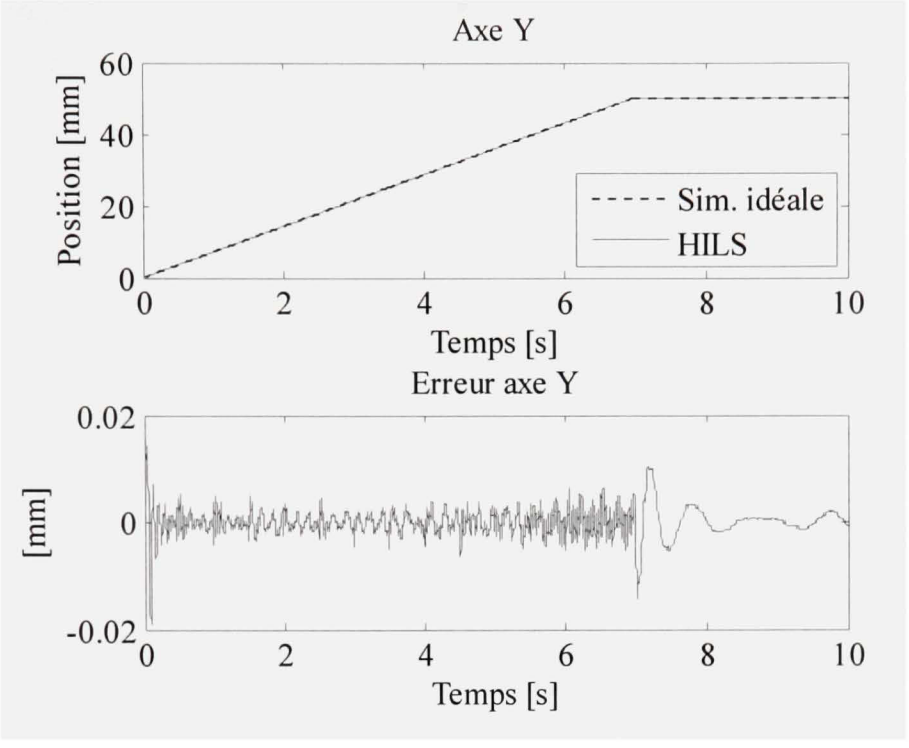

Figure 5.10 ***Position du robot virtuel (axe Y) de la simulation avec application de la contrainte.***

En ce qui concerne la position du robot virtuel selon l'axe Z, plusieurs raisons expliquent l'erreur entre la simulation idéale et la simulation avec matériel dans la boucle. Tout d'abord, pendant la première seconde, c'est-à-dire la période durant laquelle le robot n'est pas encore en contact avec l'environnement, l'erreur de position est due en partie au biais sur le capteur de force. En effet, si on observe la force lue durant cette période (voir Figure 5.9), on constate qu'elle n'est pas nulle même si le robot n'est pas encore en contact avec l'environnement. Ainsi, puisque le contrôleur du robot réel comporte un terme pour la compensation des forces de l'environnement, cette erreur sur la mesure de force apporte nécessairement une erreur de position. De plus, le contrôleur d'impédance modifie la trajectoire en fonction de ce biais. Ensuite, tel que vu sur la mesure de force de la Figure 5.9, la friction de coulomb des actionneurs crée un effet d'oscillation qui est également perceptible sur la position. Pour les raisons expliquées dans le paragraphe précédent, cette perturbation est dominante durant la partie linéaire de la trajectoire, soit lorsque le robot se déplace à faibles vitesses. Lorsque le robot est en contact avec l'environnement, l'erreur de

position est inférieure à 20µm selon l'axe Z. De plus, on doit considérer que le modèle de l'environnement identifié comporte des incertitudes. Considérant que l'environnement est très rigide, une telle différence de position entre la simulation idéale et la simulation avec matériel dans la boucle est très acceptable.

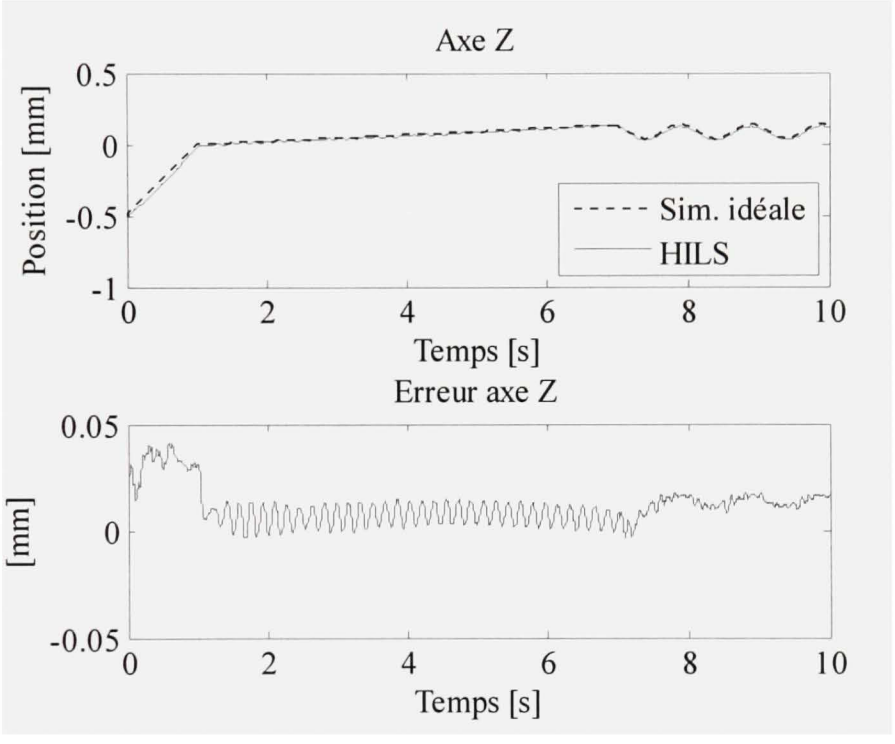

Figure 5.11 ***Position du robot virtuel (axe Z) de la simulation avec application de la contrainte.***

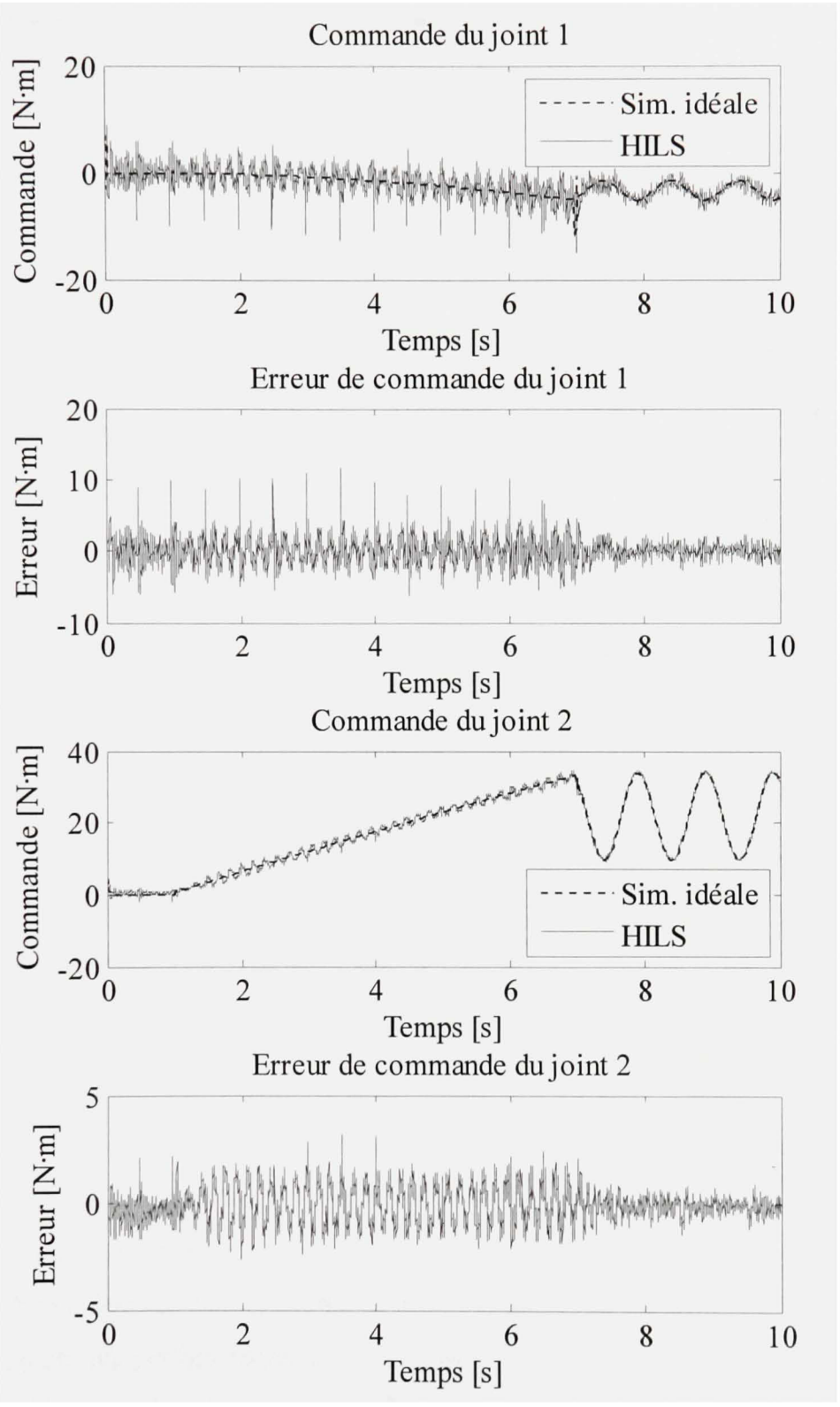

Figure 5.12 ***Commande du robot virtuel de la simulation avec application de la contrainte.***

La troisième mesure de performance de la simulation avec matériel dans la boucle consiste à comparer les couples de commande envoyés au robot virtuel. Comme dans le cas de la force de contact et de la position, l'effet d'oscillation apparaît sur la Figure 5.12. Aussi, le bruit du capteur de force influence la commande étant donné qu'elle est composée en grande partie de la compensation des forces de friction causées par le contact avec l'environnement. Pour ce qui est du joint 1, l'erreur maximale de 10N·m est non négligeable comparé à l'amplitude même de la commande. L'erreur relative est donc très élevée. Sur le joint 2, une erreur de 4N·m est obtenue par rapport à une amplitude de commande qui atteint environ 40N·m. L'erreur relative obtenue est alors d'environ 10%, ce qui est acceptable.

Sommes toutes, cette méthode basée sur la simulation avec application de la contrainte permet de reproduire raisonnablement la simulation idéale, particulièrement aux niveaux des forces et des positions. La friction de coulomb présente sur les actionneurs à entraînement direct crée un effet de cycle-limite visible sur toutes les mesures du système (force, position et commande). La version sans action intégrale du contrôleur du robot réel a néanmoins réussi à limiter les erreurs en régime permanent afin d'obtenir une mesure de position précise à l'intérieur de quelques dizaines de micromètres. On verra à la prochaine section que la simulation sans application de la contrainte n'est pas aussi efficace lorsque le robot réel est commandé par la version PD du contrôleur.

5.7.3 Simulation sans application de la contrainte

Cette section présente les résultats expérimentaux de la simulation sans application de la contrainte qui a été décrite à la section 4.2.2. Pour cette méthode, les variables d'état du robot virtuel sont calculées à partir de l'intégration numérique du modèle dynamique représenté par l'équation (4.18). Dans ce cas-ci, la pose du robot virtuel n'est pas synchronisée avec celle du robot réel. Ainsi, les performances de la simulation avec matériel dans la boucle dépendent principalement des performances du contrôleur du robot réel, c'est-à-dire de la capacité du robot réel à suivre la trajectoire du robot virtuel. Comme pour la première méthode, le robot virtuel est commandé par la loi d'impédance présentée à la section 3.1 avec les paramètres

(gains et impédance désirée) calculés à la section 5.5. La trajectoire désirée du robot virtuel est une fois de plus celle illustrée à la Figure 5.8. La prochaine sous-section présente la simulation sans application de la contrainte en utilisant la version PD du contrôleur du robot réel. Ensuite, les résultats de la version modifiée par l'ajout de l'action intégrale de l'erreur de position seront présentés.

5.7.3.1 Contrôleur linéaire proportionnel-dérivé (PD)

Cette sous-section présente les résultats expérimentaux de la méthode sans application de la contrainte et plus particulièrement la version PD du contrôleur du robot réel. Les gains du contrôleur du robot réel sont calculés à la section 5.6.1 et ont été déterminés en imposant un temps de réponse de 125ms à l'aide d'une valeur propre double.

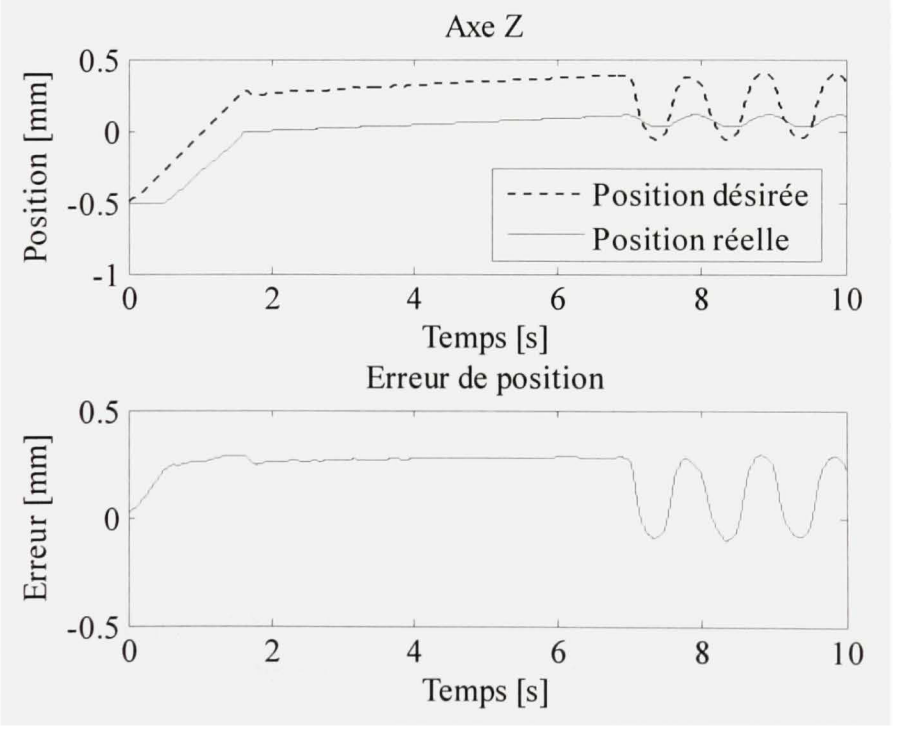

Figure 5.13 ***Position du robot réel (axe Z) de la simulation sans application de la contrainte (PD).***

Tel que mentionné dans la présentation des résultats pour la méthode avec application de la contrainte, une friction de coulomb est présente sur les actionneurs à entraînement direct. Dans le cas de la simulation sans application de la contrainte combinée au contrôleur PD du robot réel, cette perturbation se traduit principalement par une erreur de suivi de trajectoire du robot réel. En effet, la Figure 5.13 montre qu'il y a une erreur non négligeable entre la position désirée (position du robot virtuel) et la position réelle du robot. Tel qu'il sera constaté plus bas, cette erreur aura un impact sur toutes les mesures (force, position et commande) du système et affectera grandement les performances de la simulation avec matériel dans la boucle.

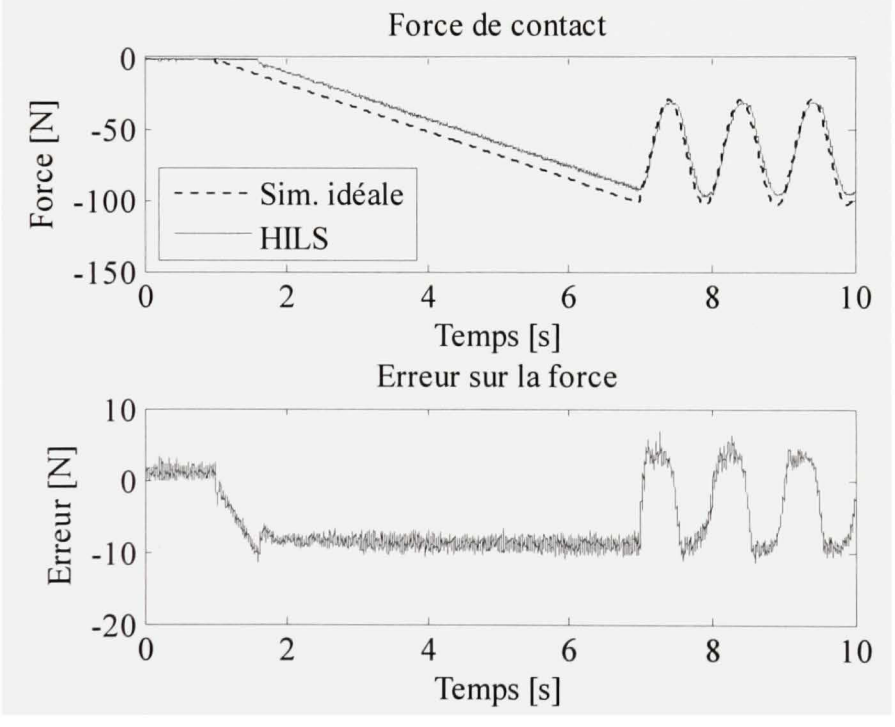

Figure 5.14 ***Force de contact de la simulation sans application de la contrainte (PD).***

Tout d'abord, comme le montre la Figure 5.14, une erreur constante d'environ 10N est présente sur la force de contact pendant la partie linéaire de la trajectoire. Par contre, pendant la partie sinusoïdale, la friction de coulomb est moins significative par rapport à l'effort de commande nécessaire pour suivre la trajectoire. En comparant l'erreur sur la force de la

Figure 5.14 à celle du suivi du robot réel de la Figure 5.13, on constate qu'il y a une similarité qui confirme que les performances de la simulation avec matériel dans la boucle sont directement dépendantes de la capacité du robot réel à suivre la position du robot virtuel.

Tel qu'illustré à la Figure 5.15, l'erreur de position du robot virtuel selon l'axe Y est négligeable. La pose du robot virtuel n'est pas synchronisée avec celle du robot réel; la force mesurée est donc la seule rétroaction du modèle du robot virtuel. Ainsi, puisqu'il n'y a aucune force mesurée selon l'axe Y, l'erreur de position est plutôt due à des erreurs de calcul numérique provenant notamment des incertitudes des termes non-linéaires du modèle.

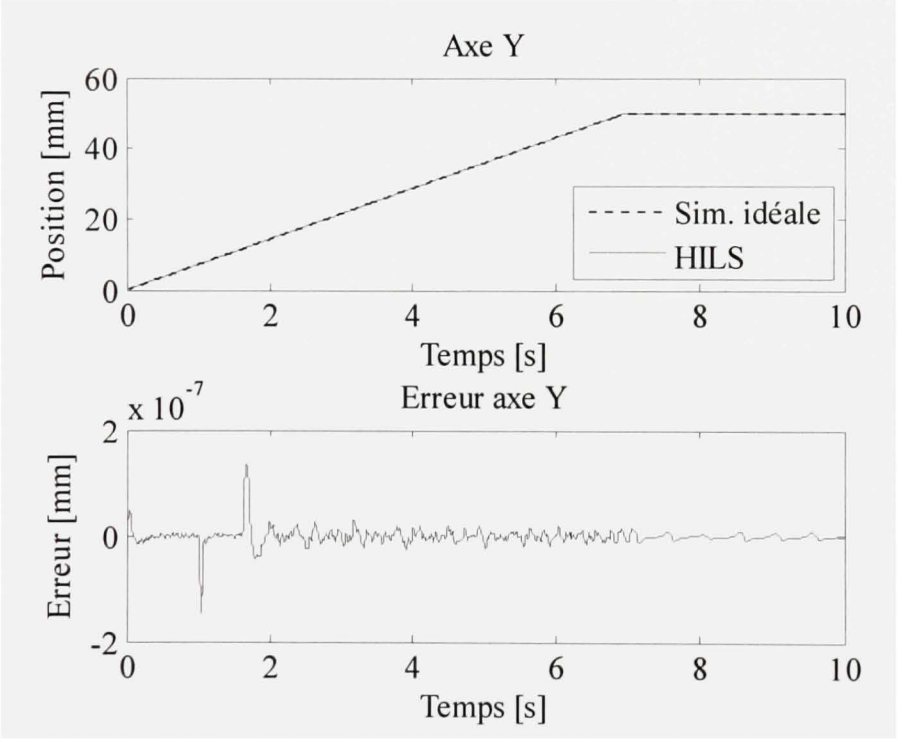

Figure 5.15 ***Position du robot virtuel (axe Y) de la simulation sans application de la contrainte (PD).***

Par contre, on ne peut pas en dire autant de la position du robot virtuel selon l'axe Z présentée à la Figure 5.16. En effet, puisque le robot réel n'arrive pas très bien à suivre la trajectoire du robot virtuel, ce dernier voit l'environnement beaucoup plus loin qu'il ne l'est en réalité et ceci cause une erreur de position importante. Encore un fois, en comparant

l'erreur de position du robot virtuel de la Figure 5.16 à l'erreur de suivi du robot réel de la Figure 5.13 on peut facilement en déduire la causalité.

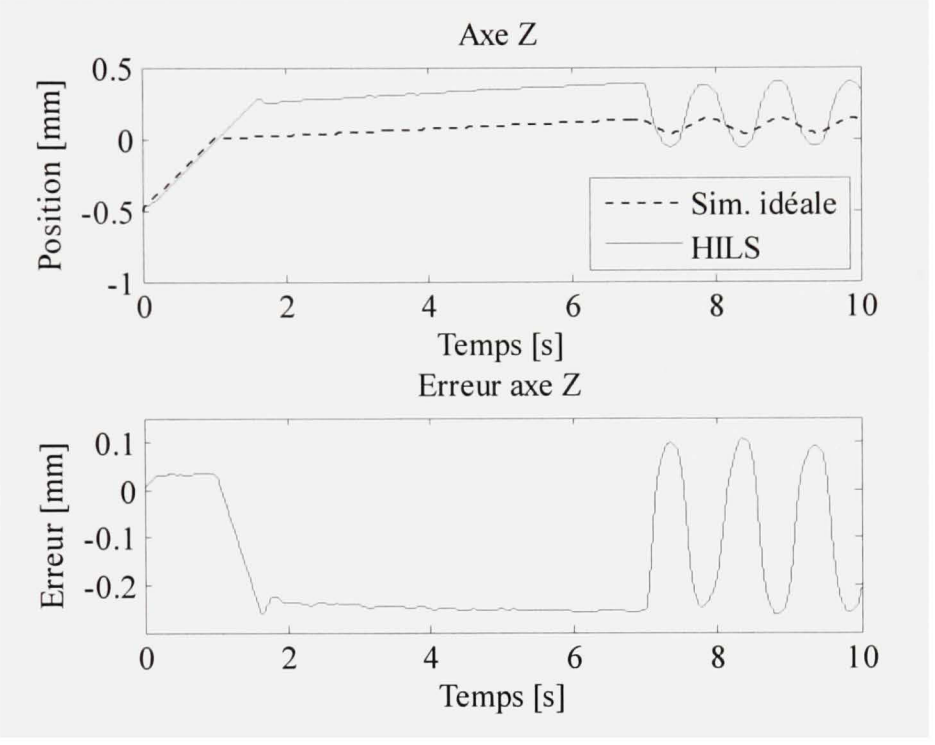

Figure 5.16 ***Position du robot virtuel (axe Z) de la simulation sans application de la contrainte (PD).***

Il a été déterminé, jusqu'à maintenant, que l'erreur de suivi de trajectoire du robot réel produit des erreurs importantes sur les mesures de force de position. Les couples de commande envoyés au robot virtuel ne font pas exception. En effet, on retrouve sur la Figure 5.17 des erreurs de commande de 1 N·m sur le joint 1 et de 4 N·m sur le joint 2. Les forces de contact sont compensées majoritairement par le joint 2 et par conséquent, l'erreur sur la force de contact montré à la Figure 5.14 est plus particulièrement visible sur l'erreur de commande de ce joint. Toutefois, l'erreur de commande est moins bruitée qu'avec l'application de la contrainte étant donné que l'effet de cycle-limite n'est pas présent.

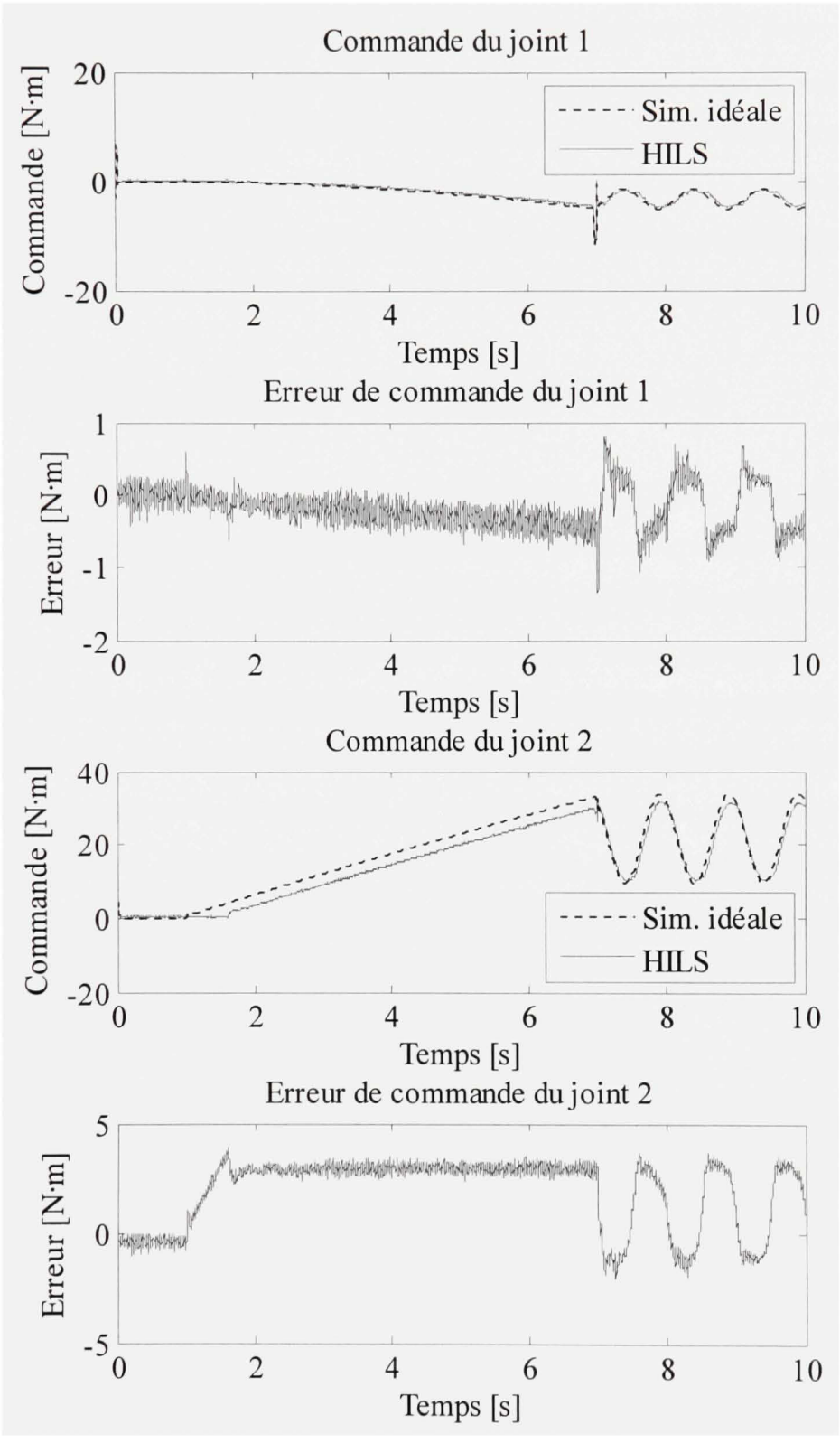

Figure 5.17 ***Commande du robot virtuel de la simulation sans application de la contrainte (PD).***

Dans cette section, il a été constaté que la simulation avec matériel dans la boucle sans application de la contrainte produit des différences importantes avec la simulation idéale. Plus particulièrement, la version PD du contrôleur du robot réel n'arrive pas à compenser totalement les effets de la friction de coulomb des actionneurs, qui est l'élément perturbateur principal du système. Pour contrer ce problème, la solution retenue est d'ajouter une action intégrale au contrôleur du robot réel. Cette solution est présentée dans la prochaine section.

5.7.3.2 Contrôleur linéaire proportionnel-intégral-dérivé (PID)

Cette section présente la simulation avec matériel dans la boucle sans application de la contrainte, mais en ajoutant une action intégrale au niveau du contrôleur de position du robot réel. Tel que vu à la section précédente, la friction de coulomb des actionneurs dégrade les performances du suivi de trajectoire du robot réel. Cette erreur de suivi se traduit en une différence importante entre la simulation idéale et la simulation avec matériel dans la boucle et ce, sur toutes les mesures du système (position, force et commande). Afin de compenser cette perturbation, on propose d'ajouter une action intégrale au contrôleur du robot réel. L'ajout d'une action intégrale à la partie linéaire du contrôleur du robot réel modifie l'ensemble des gains. Cependant, la procédure pour les calculer est la même, c'est-à-dire qu'on impose un temps de réponse de 125ms et à l'aide d'une valeur propre de multiplicité 3. La section 5.6.2 présente en détails le calcul des gains PID.

Tel que prévu, l'ajout de l'action intégrale a permis de diminuer l'erreur de suivi de trajectoire du robot réel. En effet, la Figure 5.18 montre que l'erreur de position selon l'axe Z converge rapidement à l'intérieur de 13μm pendant la partie linéaire de la trajectoire. Dans la partie sinusoïdale, cette erreur atteint des maximums d'environ 35μm. Cette erreur de suivi est minime considérant que le robot est en interaction avec un environnement rigide.

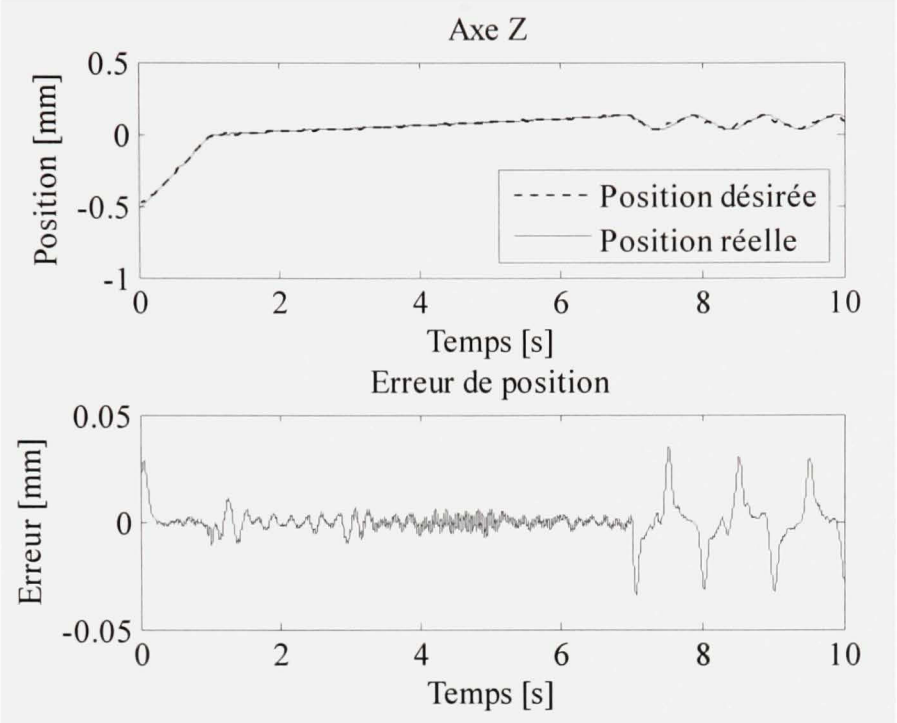

Figure 5.18 ***Position du robot réel (axe Z) de la simulation sans application de la contrainte (PID).***

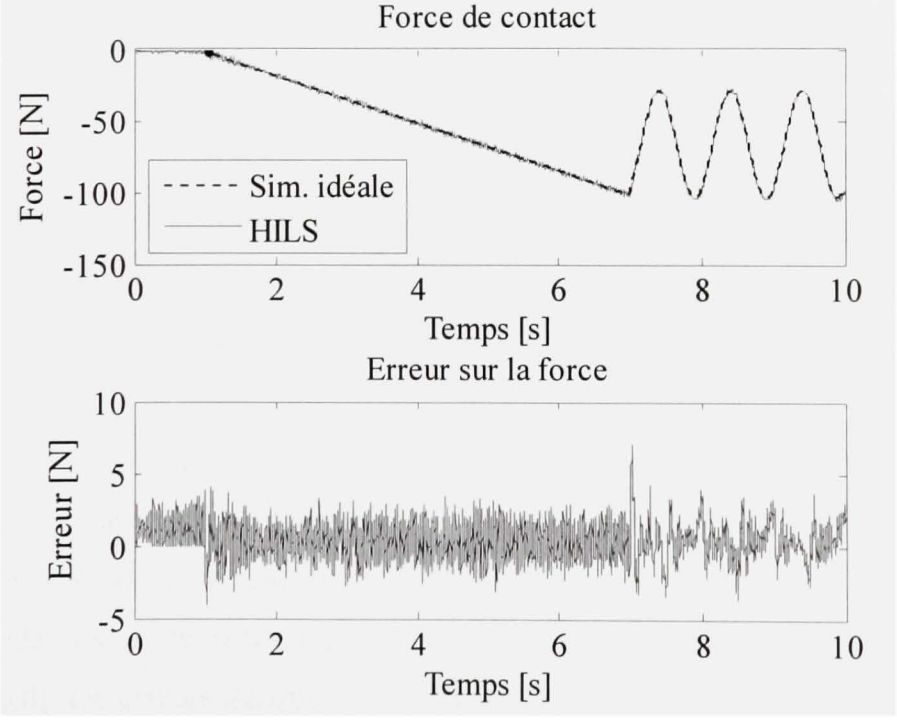

Figure 5.19 ***Force de contact de la simulation sans application de la contrainte (PID).***

L'amélioration des performances de suivi de trajectoire du contrôleur du robot réel a un impact important sur les capacités de la simulation avec matériel dans la boucle à être équivalente à la simulation idéale. La première mesure à en bénéficier est la force de contact illustrée à la Figure 5.19. Mise à part un maximum produit par la discontinuité de la vitesse désirée à la 7ème seconde, l'erreur sur la force est bornée entre ±5N et une bonne partie de cet erreur provient du bruit du capteur de force.

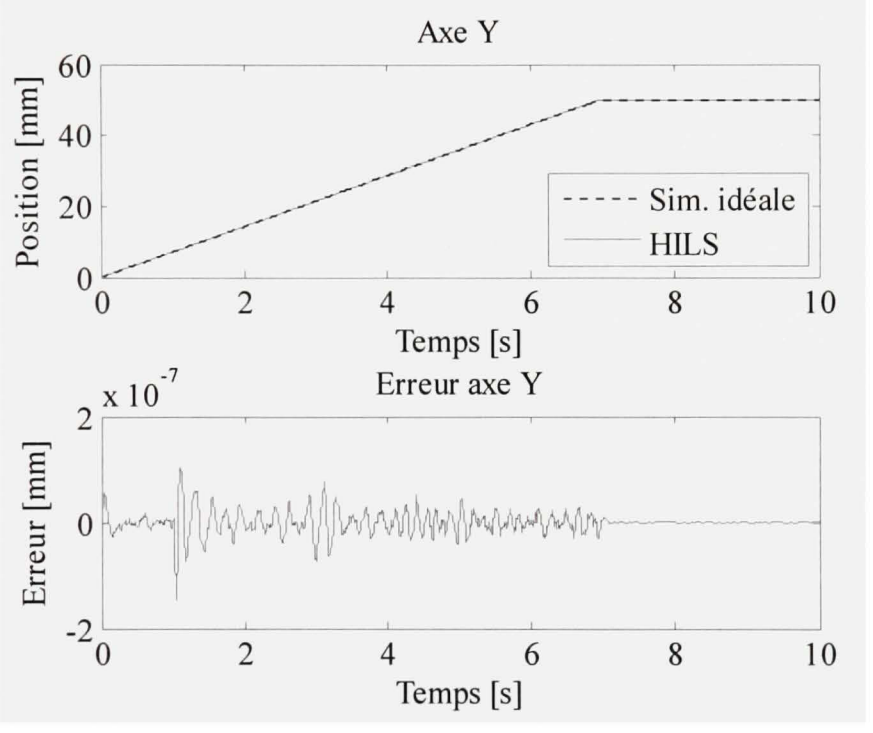

Figure 5.20 ***Position du robot virtuel (axe Y) de la simulation sans application de la contrainte (PID).***

Ensuite, en ce qui concerne la position du robot virtuel selon l'axe Y (voir Figure 5.20), les résultats sont similaires à ceux obtenus sans action intégrale. En effet, la pose du robot virtuel n'est pas synchronisée avec celle du robot réel et il n'y a aucune force mesurée selon cet axe. Ainsi, la différence entre la simulation idéale et la simulation avec matériel dans la boucle provient plutôt des erreurs de calcul numérique provenant entre autres des incertitudes des termes non-linéaires du modèle dynamique.

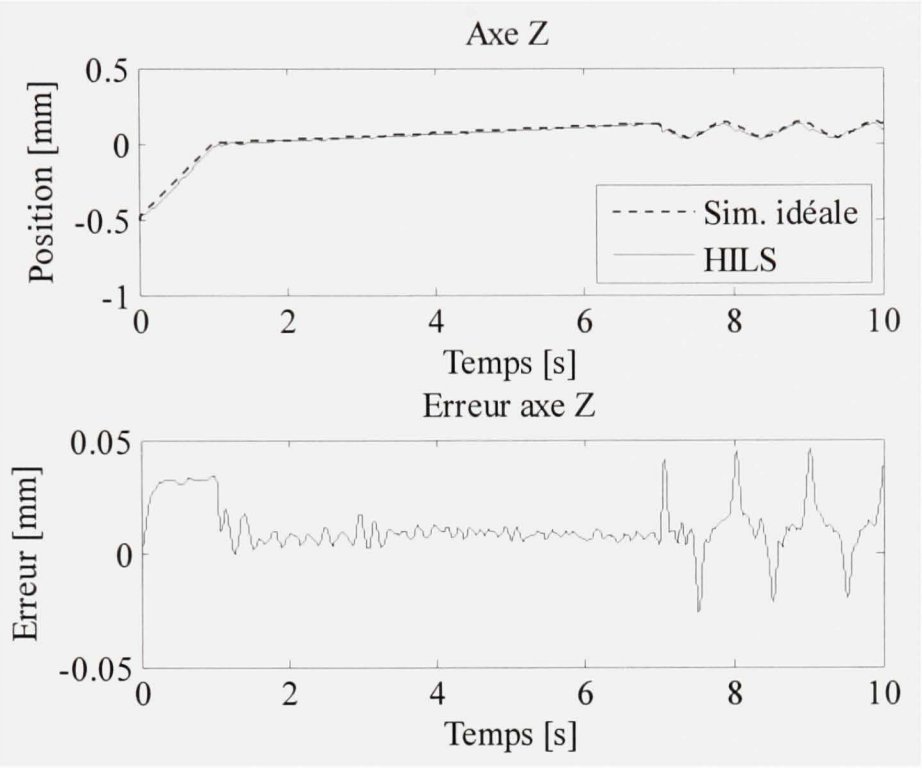

Figure 5.21 ***Position du robot virtuel (axe Z) de la simulation sans application de la contrainte (PID).***

En observant la position du robot virtuel selon l'axe Z illustrée à la Figure 5.21, on remarque que l'ajout de l'action intégrale a diminué considérablement l'erreur entre la simulation idéale est la simulation avec matériel dans la boucle. En effet, pendant la partie linéaire de la trajectoire, l'erreur converge à l'intérieur de ±20μm. Dans la partie sinusoïdale, cette erreur atteint un peu moins de 50μm. L'erreur de position présente dans la première seconde de la simulation, c'est-à-dire lorsque le robot n'est pas encore en contact avec l'environnement, est due au biais sur le capteur de force. En effet, on remarque sur la Figure 5.19 que la force de contact n'est pas tout à fait nulle lorsque le robot est dans l'espace libre.

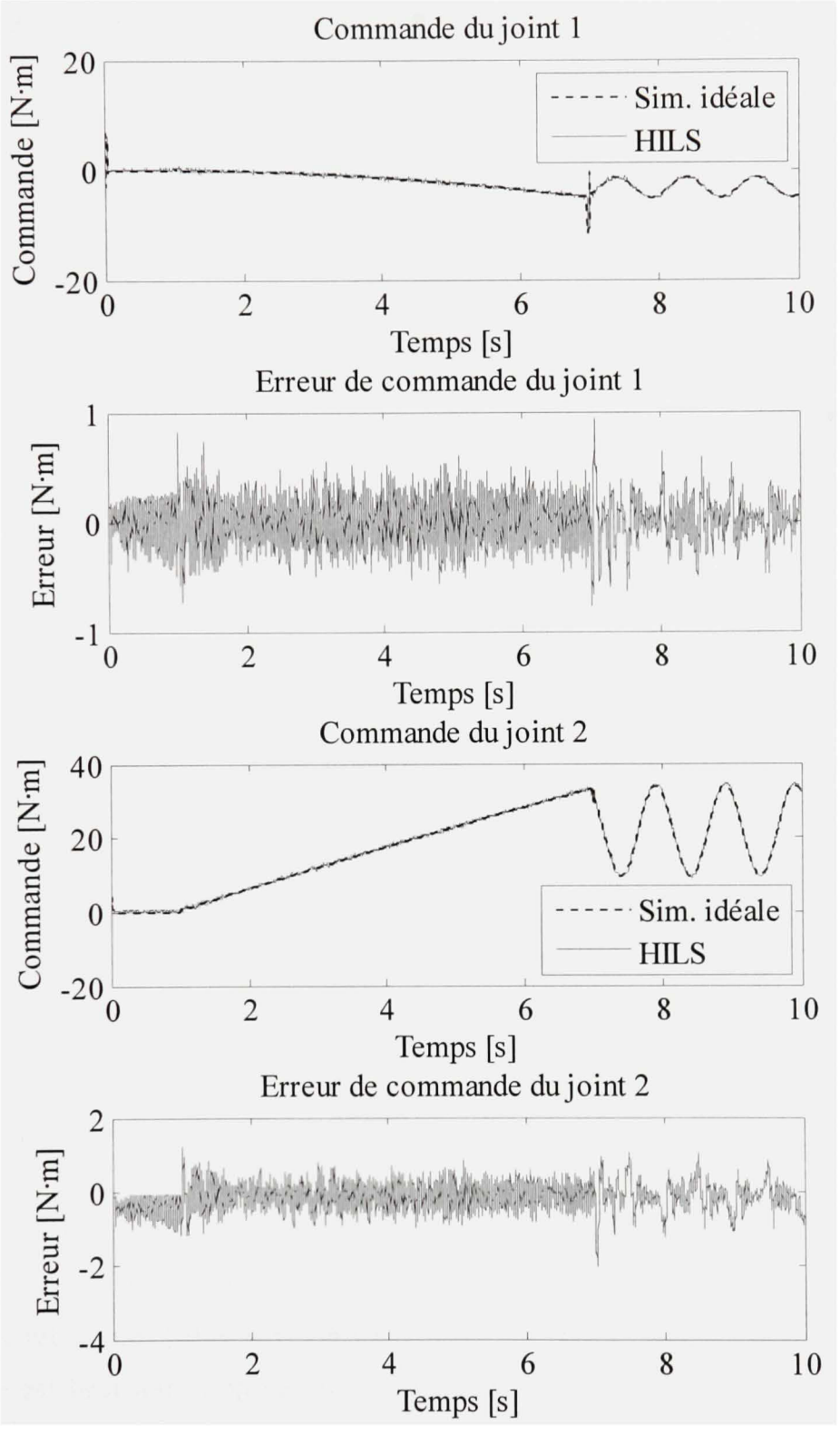

Figure 5.22 ***Commande du robot virtuel de la simulation sans application de la contrainte (PID).***

La dernière mesure utile à la validation de la simulation avec matériel dans la boucle est la commande de couple envoyée au robot virtuel. Tel que présenté à la Figure 5.22, les erreurs sur les commandes ont grandement diminué par rapport à la version sans action intégrale. En effet, on obtient des erreurs maximales de ±1N·m pour le joint 1 et de ±2N·m pour le joint 2. Par rapport aux amplitudes absolues des commandes, elles représentent respectivement des erreurs relatives d'environ 10% et 6%. Ces résultats sont très acceptables considérant qu'une bonne partie de cette erreur provient du bruit engendré par le capteur de force.

Cette section a présenté l'ajout de l'action intégrale qui a permis d'améliorer considérablement les performances de la simulation avec matériel dans la boucle sans application de la contrainte. En effet, l'intégrateur permet de compenser en grande partie les effets perturbateurs de la friction de coulomb présente sur les actionneurs linéaires à entraînement direct et ainsi, d'avoir un impact positif sur toutes les mesures du système.

5.8 Discussion des résultats de simulation

Dans ce chapitre, les deux schémas de simulation avec matériel dans la boucle ont été comparés: l'un avec et l'autre sans application de la contrainte rhéonomique. Pour la méthode sans application de la contrainte, il a été proposé d'ajouter une action intégrale afin de compenser les perturbations causées par la friction de coulomb des actionneurs. Donc, en réalité, on désire connaître le meilleur schéma de simulation HIL parmi les trois différents cas étudiés.

Premièrement, la méthode avec application de la contrainte a donné des résultats satisfaisant sur presque toutes les mesures. L'erreur sur la force est dans une plage acceptable et l'erreur de position est la plus petite obtenue parmi tous les cas étudiés. Cependant, l'erreur sur la commande est beaucoup trop élevée. De plus, les résultats obtenus montrent un effet de cycle-limite causé par la friction de coulomb des actionneurs et cela perturbe considérablement les résultats. Malgré tout, il est plausible d'utiliser cette méthode dans un

contexte où les couples de commande du robot virtuel ne nécessitent pas une grande précision et où l'effet de cycle-limite est peu nuisible.

Ensuite, la méthode sans application de la contrainte, jumelée à la version PD du contrôleur du robot réel, a montré une performance considérablement dégradée par la présence de friction de coulomb sur les actionneurs du robot réel. En effet, les mesures de force et de position présentent des biais importants, ce qui rend la simulation peu précise.

Finalement, l'ajout de l'action intégrale à la méthode sans application de la contrainte a permis d'améliorer grandement les performances de ce schéma de simulation. La mesure de force et les couples de commande sont très proches de la simulation idéale et dépassent les performances des deux autres cas étudiés. Pour ce qui de la mesure de position, elle est grandement améliorée par rapport au même schéma de simulation, mais sans action intégrale. L'erreur est toutefois légèrement plus grande que dans le cas de la simulation avec application de la contrainte. En effet, en appliquant la contrainte rhéonomique, la pose du robot virtuel est synchronisée avec celle du robot réel, ce qui permet de minimiser l'erreur de position entre la simulation idéale et la simulation HIL.

Le Tableau 5.4 présente un résumé des erreurs maximales et RMS[5] pour chacune des méthodes étudiées. Sommes toutes, on observe que la méthode sans application de la contrainte avec la version PID du contrôleur du robot réel se distingue largement des autres méthodes. Elle offre des erreurs inférieures sur les mesures de force et de couples de commande. Pour ce qui est de l'erreur de position, elle est légèrement plus grande qu'avec la simulation avec l'application de la contrainte. Cette augmentation de l'erreur représente cependant une différence peu significative. Aussi, comparativement au schéma de simulation

[5] L'erreur RMS est la racine carrée de la moyenne des erreurs au carré.

avec application de la contrainte, l'approche sans contrainte est beaucoup plus intuitive. Elle peut être décrite par un simulateur de robot virtuel avec un robot réel qui tente de suivre la trajectoire de son outil. De plus, l'application de la contrainte sur le modèle du robot virtuel nécessite en général beaucoup plus de calculs, notamment lorsque le robot virtuel possède plus de degrés de liberté que le robot réel.

Tableau 5.4

Résumé des erreurs pour chacune des méthodes de simulation avec matériel dans la boucle

		Méthode avec application de la contrainte (version PD)	Méthode sans application de la contrainte (version PD)	Méthode sans application de la contrainte (version PID)
Erreur de force [N]	Max	7,5010	11,5168	7,0392
	RMS	2,8796	7,4001	1,2068
Erreur de position selon l'axe Y [mm]	Max	0,0191	1,4678e-7	1,4573e-7
	RMS	0,0026	1,6259e-8	2,0333e-8
Erreur de position selon l'axe Z [mm]	Max	0,0417	0,2634	0,0458
	RMS	0,0144	0,2060	0,0151
Erreur de commande sur le joint 1 [N·m]	Max	1,7341	1,3575	0,9469
	RMS	1,5609	0,3342	0,1833
Erreur de commande sur le joint 2 [N·m]	Max	3,2178	3,9107	2,0397
	RMS	0,7856	2,4908	0,3613

CONCLUSION

Dans ce mémoire, le problème de la simulation avec matériel dans la boucle d'un robot virtuel en contact avec un environnement réel a été abordé. Le contexte de simulation proposé pour effectuer la preuve de concept est composé d'un robot planaire à deux degrés de liberté en interaction avec un environnement très rigide. Afin d'étudier la dynamique en boucle fermée au niveau de la force et de la position, une loi d'impédance commande le robot virtuel. La méthode à l'étude propose d'utiliser un robot réel afin d'assurer le contact avec l'environnement réel tout en reproduisant la dynamique du robot virtuel. Ainsi, le robot simulé interagit virtuellement avec l'environnement réel. Cette méthode permet donc de simuler, par le biais d'un robot réel, un robot virtuel en interaction avec un environnement réel dont on ne connait pas le modèle. Une autre particularité importante de ce projet est que le robot réel utilisé pour les expérimentations possède deux joints prismatiques à entraînement direct. Ce type d'actionneur est reconnu pour avoir une excellente bande passante par rapport aux actionneurs munis de réducteurs de vitesse. Cette caractéristique permet de reproduire plus fidèlement la dynamique du robot virtuel.

Deux méthodes ont été étudiées pour la simulation du robot en contact : la méthode avec et la méthode sans application de la contrainte. De plus, pour la méthode sans application de la contrainte, une variante au niveau du contrôleur a été proposée pour améliorer les résultats. Ces trois cas sont évalués dans un contexte où l'environnement réel est simple et connu, de façon à pouvoir les comparer à la simulation idéale. Ainsi, l'étude expérimentale a permis d'établir clairement la performance de chacun. Tout d'abord, la méthode avec application de la contrainte a donné de bons résultats. Les erreurs au niveau de la force et de la position sont petites. Cependant, l'erreur au niveau de la commande est plus significative, ce qui permet de douter de la performance de cette méthode lorsque cette mesure est importante. Ensuite, la méthode sans application de la contrainte, avec la version PD du contrôleur, a donné des résultats peu enviables. La friction de coulomb des articulations du robot réel a causé une erreur importante de suivi au niveau du contrôleur du robot réel. Cette erreur a nécessairement provoqué une différence importante entre la simulation idéale et la simulation

HIL. L'ajout de l'action intégrale, qui représente le troisième cas à l'étude, a permis de réduire cette erreur de suivi et par conséquent, d'améliorer considérablement la performance de la simulation HIL. Les erreurs au niveau de la force et de la commande ont été les plus petites de tous les essais, tandis que l'erreur de position est sensiblement la même qu'avec la méthode qui consiste à appliquer la contrainte rhéonomique. Étant donné que la méthode sans application de la contrainte est beaucoup plus intuitive que la méthode avec application de la contrainte et qu'elle nécessite généralement moins de manipulation mathématique, elle semble être à priori l'approche la plus favorable.

Dans ce mémoire, une contribution indirecte, mais d'une grande utilité, est le développement d'une méthode de prototypage rapide pour la simulation HIL. Cette procédure a permis de générer automatiquement toutes les équations symboliques requises à la simulation et ce, à partir des paramètres cinématiques et dynamiques des robots. Ainsi, en connaissant les paramètres d'un robot virtuel, il est possible en quelques minutes d'étudier son comportement lorsqu'il est en interaction avec un environnement réel.

Même si l'étude expérimentale a démontré clairement la validité de la simulation HIL pour la simulation de robots en contact, certaines recommandations peuvent êtes faites pour améliorer les performances. Tout d'abord, la friction de coulomb présente sur les actionneurs du robot réel pourrait être mieux compensée de façon à réduire les erreurs de suivi du contrôleur du robot réel. Ensuite, les paramètres dynamiques du robot réel pourraient être identifiés de façon plus précise afin de réduire les incertitudes de la loi de commande linéarisante. Finalement, une autre possibilité serait d'améliorer la calibration du capteur de force, de façon à éliminer le biais qui semble dégrader la validité de la simulation à certains moments. Sommes toutes, ces recommandations permettraient d'améliorer de façon directe ou indirecte les mesures de performance de la simulation HIL, soit la position, la force et la commande.

La suite logique de ce travail serait d'évaluer l'approche dans un contexte de meulage à sec dans les laboratoires de l'IREQ. L'application de la méthode présentée dans ce mémoire

devrait se faire assez facilement, étant donné que le modèle du manipulateur est déjà très bien connu. De plus, l'excellente capacité de suivi de trajectoire du contrôleur du robot réel permettrait sans aucun doute d'obtenir des résultats réalistes. Aussi, cette méthode pourrait être appliquée au meulage sous-marin. Cependant, dans ce contexte certaines limitations pourraient dégrader les résultats. Par exemple, le mouvement du robot réel sous l'eau engendrera des perturbations qui ne seront pas mesurées par le capteur de force. Puisque la géométrie du robot réel et généralement différente de celle du robot virtuel, les perturbations subites par le robot réel ne seront pas nécessairement celles que subirait le robot virtuel. Toutefois, dans le cas du procédé de meulage sous l'eau les vitesses peuvent être relativement petites et les perturbations non mesurées peuvent être négligeables par rapport aux forces engendrées par le procédé de meulage. Il est clair que dans ce contexte une validation expérimentale est requise.

ANNEXE I

DÉVELOPPEMENT DU MODÈLE DYNAMIQUE DU ROBOT VIRTUEL

Le modèle dynamique du robot virtuel est développé à l'aide de l'équation de Lagrange (Goldstein, 1980) suivante :

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{\mathbf{q}}_s} - \frac{\partial T}{\partial \mathbf{q}_s} + \frac{\partial V}{\partial \mathbf{q}_s} = \mathbf{J}_s^T \mathbf{f}_e + \boldsymbol{\tau}_s \tag{6.1}$$

où T est l'énergie cinétique totale, V est l'énergie potentielle totale, $\mathbf{q}_s$ est le vecteur des coordonnées généralisées, $\boldsymbol{\tau}_s \in \mathbb{R}^m$ le vecteur des forces généralisées, $\mathbf{J}_s \in \mathbb{R}^{n \times m}$ la matrice jacobienne et $\mathbf{f}_e$ les forces externes provenant de l'interaction avec l'environnement. Dans ce cas-ci, les axes de rotation des joints sont alignés avec le vecteur de gravité. Ainsi, celle-ci n'intervient pas dans la dynamique du robot et l'énergie potentielle V est nulle. Le problème se résume donc à déterminer l'énergie cinétique totale T pour ensuite calculer l'équation dynamique représentée par (6.1). Pour un système à m corps rigide, l'énergie cinétique est donnée par :

$$T = \sum_{i=1}^{m} \frac{1}{2} m_i \mathbf{v}_{c_i}^T \mathbf{v}_{c_i} + \frac{1}{2} {}^{c_i}\mathbf{w}_{c_i}^T \mathbf{I}_i {}^{c_i}\mathbf{w}_{c_i} \tag{6.2}$$

où m_i est la masse de la membrure du $i^{\text{ème}}$ joint, $\mathbf{v}_{c_i}$ est le vecteur des vitesses linéaires du centre de gravité i par rapport au repère de base, ${}^{c_i}\mathbf{w}_{c_i}$ est le vecteur des vitesses angulaires du centre de gravité i par rapport au repère de base projeté dans le repère du joint i et $\mathbf{I}_i$ est le tenseur d'inertie de la membrure i. Aussi, les centres de gravité sont placés au centre de chacune des membrures et que les tenseurs d'inerties sont exprimés par des matrices diagonales.

À partir des données cinématiques du chapitre 2, on détermine la position des centres de gravités par rapport au repère de base :

$$\begin{bmatrix} {}^0\mathbf{c}_1 \\ 1 \end{bmatrix} = \begin{bmatrix} c_{1x} \\ c_{1y} \\ c_{1z} \\ 1 \end{bmatrix} = {}^0_1\mathbf{T} \begin{bmatrix} {}^1\mathbf{c}_1 \\ 1 \end{bmatrix} = \begin{bmatrix} c_{1s} & -s_{1s} & 0 & 0 \\ s_{1s} & c_{1s} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{l_1}{2} \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{l_1 c_{1s}}{2} \\ \frac{l_1 s_{1s}}{2} \\ 0 \\ 1 \end{bmatrix} \tag{6.3}$$

$$\begin{bmatrix} {}^0\mathbf{c}_2 \\ 1 \end{bmatrix} = \begin{bmatrix} c_{2x} \\ c_{2y} \\ c_{2z} \\ 1 \end{bmatrix} = {}^0_2\mathbf{T} \begin{bmatrix} {}^2\mathbf{c}_2 \\ 1 \end{bmatrix} = \begin{bmatrix} c_{1s2s} & -s_{1s2s} & 0 & l_1 c_{1s} \\ s_{1s2s} & c_{1s2s} & 0 & l_1 s_{1s} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{l_2}{2} \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{l_2 c_{1s2s}}{2} + l_1 c_{1s} \\ \frac{l_2 s_{1s2s}}{2} + l_1 s_{1s} \\ 0 \\ 1 \end{bmatrix} \tag{6.4}$$

Ensuite, on calcul la matrice jacobienne des positions des centres de gravités exprimée par les équations (6.3) et (6.4) :

$$J_{c_1} = \begin{bmatrix} \frac{\partial c_{1x}}{\partial q_{1s}} & \frac{\partial c_{1x}}{\partial q_{2s}} \\ \frac{\partial c_{1y}}{\partial q_{1s}} & \frac{\partial c_{1y}}{\partial q_{2s}} \\ \frac{\partial c_{1z}}{\partial q_{1s}} & \frac{\partial c_{1z}}{\partial q_{2s}} \end{bmatrix} = \begin{bmatrix} \frac{-l_1 s_{1s}}{2} & 0 \\ \frac{l_1 c_{1s}}{2} & 0 \\ 0 & 0 \end{bmatrix} \tag{6.5}$$

$$J_{c_2} = \begin{bmatrix} \frac{\partial c_{2x}}{\partial q_{1s}} & \frac{\partial c_{2x}}{\partial q_{2s}} \\ \frac{\partial c_{2y}}{\partial q_{1s}} & \frac{\partial c_{2y}}{\partial q_{2s}} \\ \frac{\partial c_{2z}}{\partial q_{1s}} & \frac{\partial c_{2z}}{\partial q_{2s}} \end{bmatrix} = \begin{bmatrix} -\frac{l_2 s_{1s2s}}{2} - l_1 s_{1s} & \frac{-l_2 s_{1s2s}}{2} \\ \frac{l_2 c_{1s2s}}{2} - l_1 c_{1s} & \frac{l_2 c_{1s2s}}{2} \\ 0 & 0 \end{bmatrix} \tag{6.6}$$

Les matrices jacobiennes (6.5) et (6.6) permettent de déterminer les vitesses linéaires des centres de gravités par rapport au repère de base :

$$v_{c_1} = \begin{bmatrix} v_{c_1x} \\ v_{c_1y} \\ v_{c_1z} \end{bmatrix} = J_{c_1}{}^0 c_1 = \begin{bmatrix} \frac{-l_1 s_{1s} \dot{q}_{1s}}{2} \\ \frac{l_1 c_{1s} \dot{q}_{1s}}{2} \\ 0 \end{bmatrix} \tag{6.7}$$

$$v_{c_2} = \begin{bmatrix} v_{c_2x} \\ v_{c_2y} \\ v_{c_2z} \end{bmatrix} = J_{c_2}{}^0 c_2 = \begin{bmatrix} \frac{-l_2 s_{1s2s} \dot{q}_{1s}}{2} - l_1 s_{1s} \dot{q}_{1s} - \frac{l_2 s_{1s2s} \dot{q}_{2s}}{2} \\ \frac{l_2 c_{1s2s} \dot{q}_{1s}}{2} - l_1 c_{1s} \dot{q}_{1s} - \frac{l_2 c_{1s2s} \dot{q}_{2s}}{2} \\ 0 \end{bmatrix} \tag{6.8}$$

En ce qui concerne les vitesses angulaires des centres de gravités, elles sont déterminées par inspection :

$${}^{c_1}\mathbf{w}_{c_1} = \begin{bmatrix} 0 \\ 0 \\ \dot{q}_{1s} \end{bmatrix} \tag{6.9}$$

$$ {}^{c_2}\mathbf{w}_{c_2} = \begin{bmatrix} 0 \\ 0 \\ \dot{q}_{1s} + \dot{q}_{2s} \end{bmatrix} \tag{6.10} $$

Finalement, toutes les informations pour calculer l'énergie cinétique totale exprimée par l'équation (6.2) sont connues. Après simplifications, le résultat suivant est obtenu:

$$ \begin{aligned} T = & \left(1 + \frac{1}{8} m_{1s} l_1^2 + \frac{1}{8} m_{2s} l_2^2 + \frac{1}{2} m_{2s} l_1^2 + \frac{1}{2} m_{2s} l_1 l_2 c_{2s} \right) \dot{q}_{1s}^2 \\ & + \left(\frac{1}{2} + \frac{1}{8} m_{2s} l_2^2 \right) \dot{q}_{2s}^2 + \left(1 + \frac{1}{2} m_{2s} l_1 l_2 c_{2s} + \frac{1}{4} m_{2s} l_2^2 \right) \dot{q}_{1s} \dot{q}_{2s} \end{aligned} \tag{6.11} $$

Ensuite, en calculant l'équation de Lagrange (6.1) à partir de l'énergie cinétique obtenue à l'équation (6.11), on obtient l'équation dynamique suivante :

$$ \begin{bmatrix} \left(2 + \frac{5}{4} l_1^2 m_{2s} + l_1^2 c_{2s} m_{2s} + \frac{1}{4} l_1^2 m_{1s} \right) \ddot{q}_{1s} + \left(1 + \frac{1}{4} l_1^2 m_{2s} + \frac{1}{2} l_1^2 c_{2s} m_{2s} \right) \ddot{q}_{2s} \\ \quad - \frac{1}{2} l_1^2 m_{2s} s_{2s} \dot{q}_{2s} \left(2\dot{q}_{1s} + \dot{q}_{2s} \right) \\ \left(1 + \frac{1}{4} l_1^2 m_{2s} + \frac{1}{2} l_1^2 c_{2s} m_{2s} \right) \ddot{q}_{1s} + \left(1 + \frac{1}{4} l_1^2 m_{2s} \right) \ddot{q}_{2s} + \frac{1}{2} l_1^2 m_{2s} s_{2s} q_{1s}^2 \end{bmatrix} = \mathbf{J}_s^T \mathbf{f}_e + \boldsymbol{\tau}_s \tag{6.12} $$

L'équation dynamique (6.12) peut être ramenée à la forme générale suivante :

$$ \mathbf{M}_s \left(\mathbf{q}_s \right) \ddot{\mathbf{q}}_s + \mathbf{F}_s \left(\mathbf{q}_s, \dot{\mathbf{q}}_s \right) = \boldsymbol{\tau}_s + \mathbf{J}_s^T \mathbf{f}_e \tag{6.13} $$

où

$$ \mathbf{M}_s = \begin{bmatrix} 2 + \frac{5}{4} l_1^2 m_{2s} + l_1^2 c_{2s} m_{2s} + \frac{1}{4} l_1^2 m_{1s} & 1 + \frac{1}{4} l_1^2 m_{2s} + \frac{1}{2} l_1^2 c_{2s} m_{2s} \\ 1 + \frac{1}{4} l_1^2 m_{2s} + \frac{1}{2} l_1^2 c_{2s} m_{2s} & 1 + \frac{1}{4} l_1^2 m_{2s} \end{bmatrix} \tag{6.14} $$

$$\mathbf{F}_s = \begin{bmatrix} -\frac{1}{2} l_1^2 m_{2s} s_{2s} \dot{q}_{2s} \left(2\dot{q}_{1s} + \dot{q}_{2s} \right) \\ \\ \frac{1}{2} l_1^2 m_{2s} s_{2s} \dot{q}_1^2 \end{bmatrix} \tag{6.15}$$

La matrice jacobienne $\mathbf{J}_s$ est calculée à la section 2.3.3.

ANNEXE II

EXPROTORAPIDE.M : EXEMPLE DE PROTOTYPAGE RAPIDE DE CONTRÔLEURS DANS MATLAB

```
%-------------------------------------------------------------------------
% Auteur :      Philippe Hamelin
% Description : Exemple d'utilisation de la procédure de prototypage rapide
%               de contrôleurs avec un robot planaire à 2 DDL (Craig p.177)
%-------------------------------------------------------------------------

%-------------------------------------------------------------------------
% Initialisation des variables symboliques
%-------------------------------------------------------------------------

% Coordonnees généralisees
syms q1 q2 real;
Q = [q1 q2];

% Vitesses angulaires
syms q1p q2p real;
Qp = [q1p q2p];

% Accélérations angulaires
syms q1pp q2pp real;
Qpp = [q1pp q2pp];

% Couples aux joints
syms tau1 tau2 real;
Tau = [tau1 tau2];

%-------------------------------------------------------------------------
% Initialisation des paramètres cinématiques et dynamiques du robot
%-------------------------------------------------------------------------

% Longueur de la membrure 1 (m)
l1 = 1;
% Longueur de la membrure 2 (m)
l2 = 1;
% Masse de la membrure 1 (kg)
m1 = 1;
% Masse de la membrure 2 (kg)
m2 = 1;

%-------------------------------------------------------------------------
% Insertion des paramètres cinématiques et dynamiques dans la version
% modifiée du robotics toolbox de Peter Corke
%-------------------------------------------------------------------------

% Tableau des paramètres HD (version modifiée, Craig)
% Liens{i} = [alpha(i) a(i-1) theta(i) D(i)% Type(0=rotoïde,1=prismatique)]
Liens{1} = link([0 0 0 0 0], 'modified');
Liens{2} = link([0 l1 0 0 0], 'modified');

% Joint 1
Liens{1}.m = m1;                        % Masse
Liens{1}.I = zeros(3,3);                % Matrice d'inertie
Liens{1}.r = [l1 0 0];                  % Position du centre de gravité
Liens{1}.G = 1;                         % Ratio d'engrenage
Liens{1}.Jm = 0;                        % Inertie du moteur
Liens{1}.B = 0;                         % Friction visqueuse

% Joint 2
Liens{2}.m = m2;                        % Masse
```

```
60      Liens{2}.I = zeros(3,3);                % Matrice d'inertie
61      Liens{2}.r = [l2 0 0];                  % Position du centre de gravité
62      Liens{2}.G = 1;                         % Ratio d'engrenage
63      Liens{2}.Jm = 0;                        % Inertie du moteur
64      Liens{2}.B = 0;                         % Friction visqueuse
65
66      % Appel de la fonction de construction du robot du toolbox
67      Robot2DDL = robot(Liens);
68
69      % Gravite selon Y0
70      Robot2DDL.gravity = [0;9.81;0];
71
72      % Initialisation du repère de l'outil (translation en X de l2)
73      Robot2DDL.tool = transl([l2 0 0]);
74
75      %-------------------------------------------------------------------------
76      % Calcul de la cinematique directe par rapport au reper de base du robot
77      %-------------------------------------------------------------------------
78
79      CinDirecte = fkine(Robot2DDL,Q);
80
81      % On extrait seulement la partie position [x,y] de la matrice de
82      % transformation homogène
83      CinDirecte = CinDirecte(1:2,4);
84
85      %-------------------------------------------------------------------------
86      % Calcul de l'équation dynamique du robot (sans forces externes)
87      %-------------------------------------------------------------------------
88
89      % Matrice d'inertie
90      DynM = simple(inertia(Robot2DDL, Q));
91
92      % Vecteur des forces de coriolis et centrifuge
93      DynF = simple(coriolis(Robot2DDL, Q, Qp)');
94
95      % Vecteur des forces de gravité
96      DynG = gravload(Robot2DDL, Q)';
97
98      % Dynamique directe
99      DynQpp = simple(inv(DynM)*(Tau'-DynF-DynG));
100
101     %-------------------------------------------------------------------------
102     % Calcul de la loi de commande linéarisante dans l'espace articulaire
103     %-------------------------------------------------------------------------
104
105     syms u1 u2 real;
106     u = [u1 u2];
107
108     % tau = M*u+F, où u est l'entree auxiliaire
109     CmdLinearisante = simple(DynM*u'+DynF+DynG);
110
111     % Placement de pôles PD avec imposition du temps de réponse Tr (sec)
112     Tr = 0.1;
113     Lambda = 4.73/Tr;
114     Kp = ones(2,1)*Lambda*Lambda;
115     Kd = ones(2,1)*2*Lambda;
116
117     %-------------------------------------------------------------------------
118     % Génération de la trajectoire
119     %-------------------------------------------------------------------------
120
121     % Période d'échantillonnage
122     T = 0.001;
123
124     % Temps final
125     Tf = 1;
126
127     % Vecteur temps
128     t = 0:T:Tf;
```

```
129
130     % Position initiale
131     Qinit = [0 0];
132
133     % Position finale
134     Qfinal = [pi/4 pi/4];
135
136     % Trajectoire dans l'espace articulaire
137      [TrajQ TrajQp TrajQpp] = jtraj(Qinit, Qfinal, t);
138
139     %-------------------------------------------------------------------------
140     % Génération des S-Functions
141     %-------------------------------------------------------------------------
142
143     DynQpp_params = GenSFunction(DynQpp, 'DynQpp', [Tau Q Qp]);
144     CmdLinearisante_params = GenSFunction(CmdLinearisante, ...
145         'CmdLinearisante', [u Q Qp]);
146     CinDirecte_params = GenSFunction(CinDirecte, 'CinDirecte', Q);
147
148     %-------------------------------------------------------------------------
149     % Compilation des S-Functions
150     %-------------------------------------------------------------------------
151
152     mex DynQppssm.c;
153     mex CmdLinearisantessm.c;
154     mex CinDirectessm.c
```

ANNEXE III

FICHIERS POUR GÉNÉRATION DES S-FUNCTIONS À PARTIR D'UNE ÉQUATION SYMBOLIQUE DANS MATLAB

GenSFunction.m

```
%-------------------------------------------------------------------------
% Auteur :      Philippe Hamelin
%
% Date :        12 décembre 2007
%
% Description : Cette fonction permet de générer une S-Fonction a partir
%               d'une équation symbolique. Elle fait appel au fichier
%               source Maple 'Function2sfile.maple' qui est extrait du
%               toolbox de robotique de Pascal Bigras.
%               ref. : http://www.gpa.etsmtl.ca/cours/sys827/
%
% Utilisation:  params = GenSFunction(expression, name, X)
%
%               expression: variable contenant l'équation symbolique
%               name:       nom de la S-Function
%               X:          vecteur des variables d'entrées
%               params:     liste des parametres constants, soit les
%                           variables qui ne sont pas présentes à l'entrée
%-------------------------------------------------------------------------

function params = GenSFunction(expression, name, X)

% Extration du répertoire ou est situé ce fichier
[pathstr, filename, ext, versn] = fileparts(mfilename('fullpath'));
fctFullPath = strrep(pathstr,'\','/');

% Initialisation du nom de la S-Function
Output = name;
if isempty(Output), Output = 'Output'; end

% Initialisation du vecteur d'entrée (variables indépendantes)
InputStr = char(X,1);
InputStr = strrep(InputStr,'vector','');
maple(['X :=' InputStr]);

% Initialisation du chemin d'écriture des fichiers
maple('chemin := "./":');

% Initialisation du nom de la S-Function
maple(['modelName :=  ' Output ' :']);

% Initialisation de l'expression symbolique
maple(['F :=' char(expression)]);

% Exécution de la procédure Function2sfile
maple(['fctFullPath := `' fctFullPath '`:']);
maple('read cat(fctFullPath, /Function2sfile.maple )');
params = maple('par := par');

if(strcmp(params,'[pas_de_parametre]'))
    params = [];
end
```

Function2sfile.maple

```
# recupere les sources de la s-fonction en deux partie
fr := fopen(cat(fctFullPath,`/src/rb_statsl.c`),READ):
sFuncStat1_ := []:
strLine := readline(fr):
for ii from 1 by 1 while strLine <> 0 do
sFuncStat1_ := [op(sFuncStat1_),strLine]:
strLine := readline(fr):
od:
fclose(fr):

fr := fopen(cat(fctFullPath,`/src/rb_stats2.c`),READ):
sFuncStat2_ := []:
strLine := readline(fr):
for ii from 1 by 1 while strLine <> 0 do
sFuncStat2_ := [op(sFuncStat2_),strLine]:
strLine := readline(fr):
od:
fclose(fr):

# recupere les sources de la mex-fonction en deux partie
fr := fopen(cat(fctFullPath,`/src/rb_statml.c`),READ):
mFuncStat1_ := []:
strLine := readline(fr):
for ii from 1 by 1 while strLine <> 0 do
mFuncStat1_ := [op(mFuncStat1_),strLine]:
strLine := readline(fr):
od:
fclose(fr):

fr := fopen(cat(fctFullPath,`/src/rb_statm2.c`),READ):
mFuncStat2_ := []:
strLine := readline(fr):
for ii from 1 by 1 while strLine <> 0 do
mFuncStat2_ := [op(mFuncStat2_),strLine]:
strLine := readline(fr):
od:
fclose(fr):

#Dimention de F
if (type(F,matrix) = true) then
  nRow:=linalg[rowdim](F):
  nCol:=linalg[coldim](F):
else
  nRow:=linalg[vectdim](F):
  nCol:=1:
end if:

#Transformation de la matrice en vecteur colonne whise
Fv := evalm(convert(evalm(transpose(F)),vector)):

# Extraction des parametres du model
parTot := select(type,indets(convert(Fv,list)),name):
par := sort([op(parTot minus {op(X)})]):
if (nops(par) = 0) then
  par := [pas_de_parametre]:
end if:

# Generarion du code d'une procedure Maple qui calcul le vecteur Fv
fun := codegen[makeproc](Fv,[op(X),op(par)]):
fun := codegen[packparams](fun,X,x):
fun := codegen[packparams](fun,par,p):

############################ Fichier des dimentions ######################

file:=fopen(cat(chemin,"\\",modelName,`sLn.h`),WRITE):
```

```
fprintf(file,`\n`):
fprintf(file,`/* nombre d'entrees */ \n`):
fprintf(file,`#define NB_INPUT %d \n`,nops(convert(X,list))):
fprintf(file,`\n`):
fprintf(file,`/* nombre de sorties  */ \n`):
fprintf(file,`#define NB_OUTPUT %d \n`,nops(convert(Fv,list))):
fprintf(file,`\n`):
fprintf(file,`#define NBROW_OUTPUT %d \n`,nRow):
fprintf(file,`\n`):
fprintf(file,`#define NBCOL_OUTPUT %d \n`,nCol):
fprintf(file,`\n`):
fprintf(file,`/* nombre de parametres */ \n`):
fprintf(file,`#define NB_PARAM %d \n`,nops(convert(par,list))):
fclose(file):

########################## Fichier de la fonction du modele ###############

file:=fopen(cat(chemin,"\\",modelName,`Sta.c`),WRITE):

codegen[C](fun,optimized,ansi,filename=cat(chemin,"\\",modelName,`Sta.c`)):

par:

############################### s-function ####################################

# Ouvre le fichier c de la s-Fonction
file:=fopen(cat(chemin,"\\",modelName,`ssm.c`),WRITE):

# Copy la premiere partie du fichier c
for ii from 1 by 1 while ii <= nops(sFuncStat1_) do
  fprintf(file,"%s\n",sFuncStat1_[ii]):
od:

# genere la deuxieme partie du fichier c
fprintf(file,"#define S_FUNCTION_NAME %sssm\n",modelName):
fprintf(file,"#include \"%ssLn.h\"\n",modelName):
fprintf(file,"#include \"%sSta.c\"\n",modelName):

# Copy la derniere partie du fichier c
for ii from 1 by 1 while ii <= nops(sFuncStat2_) do
  fprintf(file,"%s\n",sFuncStat2_[ii]):
od:
fclose(file):

############################## mex-function ###################################

# Ouvre le fichier c de la mex-Fonction
file:=fopen(cat(chemin,"\\",modelName,`mex.c`),WRITE):

# Copy la premiere partie du fichier c
for ii from 1 by 1 while ii <= nops(mFuncStat1_) do
  fprintf(file,"%s\n",mFuncStat1_[ii]):
od:

# genere la deuxieme partie du fichier c
fprintf(file,"#include \"%ssLn.h\"\n",modelName):
fprintf(file,"#include \"%sSta.c\"\n",modelName):

# Copy la derniere partie du fichier c
for ii from 1 by 1 while ii <= nops(mFuncStat2_) do
  fprintf(file,"%s\n",mFuncStat2_[ii]):
od:
fclose(file):

par:
```

Rb_stats1.c

```
/*****************************************************************************
*                                                                            *
* Laboratoire de Calcul Avancé (LCA)                                         *
* Décanat des études supérieures et de la Recherche                          *
* École de Technologie Supérieure                                            *
*                                                                            *
* Description: S-function en c générée par la librairie de modélisation      *
* cinematique de robot Rb_cin.                                               *
*                                                                            *
*****************************************************************************/

/* nom de la fonction simulink et include du modele et des dimensions*/
```

Rb_stats2.c

```
#define S_FUNCTION_LEVEL 2
#include <stdio.h>
#ifdef MATLAB_MEX_FILE
#include "mex.h"
#endif

#include "simstruc.h"

/* Arguments suplementaire de la fonction */
#define PAR  ssGetSFcnParam(S,0)

/* Fonction simulink pour la definition des grandeur */
static void mdlInitializeSizes(SimStruct *S)
{
   /* parametres */
   ssSetNumSFcnParams(S,1); /*Number of expected parameters*/
   if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) return;
   if (ssGetErrorStatus(S) != NULL) return;

   /* input */
   if (!ssSetNumInputPorts(S, 1)) return;
   /* ssSetInputPortWidth(S, 0, NB_INPUT); */
   if(!ssSetInputPortMatrixDimensions(S, 0,NB_INPUT,1)) return;
   ssSetInputPortDirectFeedThrough(S, 0, 1);

   /* output */
   if ( !ssSetNumOutputPorts( S, 1 ) ) return;
   if(!ssSetOutputPortMatrixDimensions(S, 0,NBROW_OUTPUT,NBCOL_OUTPUT)) return;

   ssSetNumSampleTimes(S, 1);

}

/* specifications associees a la periode d'echantillonnage lorsque le systeme est discret */
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
}/* mdlInitializeSampleTimes */


/* Calcul des sorties */
static void mdlOutputs(SimStruct *S, int_T tid)
{
   InputRealPtrsType u = ssGetInputPortRealSignalPtrs(S,0);
   real_T            *y     = ssGetOutputPortRealSignal(S,0);
   const real_T      *par     = mxGetPr(PAR);
```

```
   fun((double *)(*u),(double *)par,(double *)y);
}

static void mdlTerminate(SimStruct *S)
{
    UNUSED_ARG(S); /* unused input argument */

} /* end mdlTerminate */


#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

Rb_statm1.c

```
/*****************************************************************************
*                                                                            *
* Laboratoire de Calcul Avance (LCA)                                         *
* Décanat des études supérieures et de la Recherche                          *
* École de Technologie Supérieure                                            *
*                                                                            *
* Description: Mex-function en c générée par la librairie de modélisation    *
* cinematique de robot Rb_cin.                                               *
*                                                                            *
*****************************************************************************/
```

Rb_statm2.c

```
#include "mex.h"

void mexFunction( int nlhs, mxArray *plhs[],
                  int nrhs, const mxArray *prhs[])
{
  double *x,*par,*y;
  int     status,mrows,ncols;

  if(nrhs!=2)
    mexErrMsgTxt("Two inputs required.");
  if(nlhs!=1)
    mexErrMsgTxt("One output required.");

  if ((mxGetM(prhs[0])*mxGetN(prhs[0])) != NB_INPUT)
    mexErrMsgTxt("The size of input is incorrect.");

  if ((mxGetM(prhs[1])*mxGetN(prhs[1])) != NB_PARAM)
    mexErrMsgTxt("The size of parameters vector is incorrect.");

  /*  Create a pointer to the input matrix x and par. */
  x = mxGetPr(prhs[0]);
  par = mxGetPr(prhs[1]);

  /*  Set the output pointer to the output matrix. */
  plhs[0] = mxCreateDoubleMatrix(NBROW_OUTPUT,NBCOL_OUTPUT, mxREAL);

  /*  Create a C pointer to a copy of the output matrix. */
  y = mxGetPr(plhs[0]);

  /*  Call the C subroutine. */
  fun(x,par,y);
}
```

BIBLIOGRAPHIE

Aghili, F., et J. C. Piedboeuf. 2000. « Hardware-in-loop simulation of robots interacting with environment via algebraic differential equation ». In. Vol. 3, p. 1590-1596. Coll. « IEEE International Conference on Intelligent Robots and Systems ». Takamatsu: Institute of Electrical and Electronics Engineers Inc.

Aghili, F., et J. C. Piedboeuf. 2006. « Emulation of robots interacting with environment ». Ieee-Asme Transactions on Mechatronics, vol. 11, n° 1 (Feb), p. 35-46.

Aghili, Farhad, et Jean-Claude Piedboeuf. 2002. « Contact dynamics emulation for hardware-in-loop simulation of robots interacting with environment ». In. Vol. 1, p. 523-529. Coll. « Proceedings - IEEE International Conference on Robotics and Automation ». Washington, DC, United States: Institute of Electrical and Electronics Engineers Inc. <http://dx.doi.org/10.1109/ROBOT.2002.1013412 >.

Baumgarte, J. 1972. « Stabilization of constraints and integrals of motion in dynamical systems ». Computer Methods in Applied Mechanics and Engineering, vol. 1, p. 1-16.

Bigras, Pascal. 2007. *Site web du cours SYS827 : Systèmes robotiques en contact*. En ligne. <http://www.gpa.etsmtl.ca/cours/sys827/>. Consulté le 12 décembre 2007.

Brinksmeier, E., J. C. Aurich, E. Govekar, C. Heinzel, H. W. Hoffmeister, F. Klocke, J. Peters, R. Rentsch, D. J. Stephenson, E. Uhlmann, K. Weinert et M. Wittmann. 2006. « Advances in Modeling and Simulation of Grinding Processes ». CIRP Annals - Manufacturing Technology, vol. 55, n° 2, p. 667-696.

Chiaverini, Stefano, Bruno Siciliano et Luigi Villani. 1999. « Survey of robot interaction control schemes with experimental comparison ». IEEE/ASME Transactions on Mechatronics, vol. 4, n° 3, p. 273-285.

Corke, P.I. 1996. « A Robotics Toolbox for MATLAB ». IEEE Robotics and Automation Magazine, vol. 3, p. 24-32.

de Carufel, J., E. Martin et J. C. Piedboeuf. 2000. « Control strategies for hardware-in-the-loop simulation of flexible space robots ». In *Iee Proceedings-Control Theory and Applications* (Nov). Article. Vol. 147, p. 569-579. 6. <<Go to ISI>://000167096400003 >.

Gilardi, G., et I. Sharf. 2002. « Literature survey of contact dynamics modelling ». Mechanism and Machine Theory, vol. 37, n° 10, p. 1213-1239.

Goldstein, Herbert. 1980. *Classical Mechanics*, Second Edition. Addison Wesley.

Hamelin, Philippe, Pascal Bigras, Julien Beaudry, Sylvain Lemieux et Michel Blain. 2008a. « Hardware-in-the-loop Simulation of an Impedance Controlled Robot Using a Direct-Drive Test Bench ». In *IEEE International Symposium on Industrial Electronics 2008* (Cambridge (UK), 30 juin au 2 juillet 2008).

Hamelin, Philippe, Pascal Bigras, Sylvain Lemieux et Michel Blain. 2008b. « Hardware-in-the-loop Simulation of an Impedance Controlled Robot Interacting with a Real Environment ». In *The 10th IASTED International Conference on Control and Applications* (Quebec City, Quebec, Canada, 26-28 mai 2008).

IREQ. 2007. *MICROB : Modules Intégrés pour le Contrôle de ROBots*. En ligne. <http://www.robotique.ireq.ca/microb/>. Consulté le 13 décembre 2007.

J. Craig, John. 2004. *Introduction to Robotics: Mechanics and Control*, 3rd.: Prentice Hall.

Kircanski, Nenad M., et Andrew A. Goldenberg. 1997. « An Experimental Study of Nonlinear Stiffness, Hysteresis, and Friction Effects in Robot Joints with Harmonic Drives and Torque Sensors ». The International Journal of Robotics Research, vol. 16, n° 2 (April 1, 1997), p. 214-239.

Krenn, Rainer, et Bernd Schaefer. 1999. « Limitations of hardware-in-the-loop simulations of space robotics dynamics using industrial robots ». In, 440. p. 681-686. Coll. « European Space Agency, (Special Publication) ESA SP ».

Lawrence, D. A. 1988. « Impedance control stability properties in common implementations ». In *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*. p. 1185-1190 vol.2.

Leigh, James Ronald. 2004. *Control Theory (2nd Edition)*. Institution of Engineering and Technology (IET), 297 p.

Leitner, J. 1996. « Space technology transition using hardware in the loop simulation ». In. Vol. vol.2, p. 303-11. Coll. « 1996 IEEE Aerospace Applications Conference. Proceedings (Cat. No.96CH35904) ». Aspen, CO, USA: IEEE. <http://dx.doi.org/10.1109/AERO.1996.495985>.

Lemieux, Sylvain, Julien Beaudry et Michel Blain. 2006. « Force Control Test Bench for Underwater Vehicle-Manipulator System Applications ». In *IECON 2006 - 32nd IEEE Annual Conference on Industrial Electronics*. p. 4036-4042.

Martin, Adrian, Eric Scott et M. Reza Emami. 2006. « Design and development of robotic hardware-in-the-loop simulation ». In., p. 4150216. Coll. « 9th International Conference on Control, Automation, Robotics and Vision, 2006, ICARCV '06 ». Singapore, Singapore: Institute of Electrical and Electronics Engineers Computer

Society, Piscataway, NJ 08855-1331, United States. <http://dx.doi.org/10.1109/ICARCV.2006.345412>.

MathWorks. 2007. *The MathWorks : MATLAB and Simulink for Technical Computing*. En ligne. <http://www.mathworks.com/>. Consulté le 12 décembre 2007.

Poiré, Vincent. 2006. « Migration de lois de contrôle d'un environnement de simulation vers un cadre d'application robotique ». Montréal, École de technologie supérieure, 180 p.

Poire, Vincent, Sylvain Lemieux, Pascal Bigras et Michel Blain. 2006. « Migration procedure of control laws from a graphical simulation environment to a robotic framework ». In. Vol. 2006, p. 165-170. Coll. « Proceedings of the Eight IASTED International Conference on Control and Applications ». Montreal, QC, Canada: Int. Assoc. of Science and Technology for Development, Calgery - Alberta, T3B OM6, Canada.

QNX. 2007. *QNX Real-Time Operating System (ROTS) Software*. En ligne. <http://www.qnx.com/>.

Short, Michael, et Michael J. Pont. 2005. « Hardware in the loop simulation of embedded automotive control systems ». In. Vol. 2005, p. 226-231. Coll. « IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC ». Vienna, Austria: Institute of Electrical and Electronics Engineers Inc., Piscataway, NJ 08855-1331, United States. <http://dx.doi.org/10.1109/ITSC.2005.1520052>.

Siciliano, Bruno, et Luigi Villani. 1999. *Robot Force Control*. Kluwer Academic Publishers, 164 p.

Wolovich, W. A., et H. Elliott. 1984. « A computational technique for inverse kinematics ». In *Decision and Control, 1984. The 23rd IEEE Conference on*, sous la dir. de Elliott, H. Vol. 23, p. 1359-1363.