ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MEMOIRE DE RECHERCHE PRÉSENTÉ A
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
A L'OBTENTION DE LA
MAITRISE EN TECHNOLOGIE DES SYSTEMES
M. Ing.

PAR
PATRICK LESSARD

CONTROLE DE LA DÉMARCHE DE HEADUS
ROBOT QUADRUPEDE DYNAMIQUE

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE, LE 24 MAI 2002

CE MEMOIRE DE RECHERCHE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

- M. Pascal Bigras, professeur-tuteur
  Département de génie de la production automatisée à l'École de technologie supérieure

- M. Thiagas S. Sankar, professeur-cotuteur
  Département de génie de la production automatisée à l'École de technologie supérieure

- M. Maarouf Saad, professeur
  Département de génie électrique à l'École de technologie supérieure

- M. Michel Lambert
  Département de robotique avancé chez Opal-RT technologies

IL A FAIT L'OBJET D'UNE PRÉSENTATION

DEVANT LE JURY ET UN PUBLIC

LE 24 AVRIL 2002

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

CONTRÔLE DE LA DÉMARCHE DE HAEDUS
ROBOT QUADRUPÈDE DYNAMIQUE

Patrick Lessard

SOMMAIRE

Les robots marcheurs font l'objet d'études depuis moins de 25 ans seulement. La revue de littérature contenue dans ce mémoire effectue un survol du problème de la démarche si complexe et des techniques de contrôle associées. La méthode présentée dans cet ouvrage concerne une partie du problème de la démarche, soit l'asservissement des membres du robot. Le contrôleur est basé sur un modèle dynamique. À partir d'une trajectoire assignée, une commande de couple pré-calculé est appliquée au modèle pour calculer les efforts nécessaires pour produire le mouvement du corps et des pattes. La répartition de ces efforts au niveau des articulations est effectuée par un calcul matriciel nommé pseudo-inverse. Dans l'approche proposée, la résolution de la dynamique inverse est simplifiée par deux transformations. La première effectue un changement de coordonnées, ce qui permet de contrôler le robot dans l'espace de travail. La deuxième est basée sur l'hypothèse que les pattes touchant le sol portent le robot et sont considérées sous contraintes. Ceci permet de réduire le modèle et de prendre en charge chaque configuration de patte à savoir si elles sont en contact avec le sol ou non. En somme, cette approche permet de simplifier la conception du contrôleur, réduit le temps de calcul et la complexité du programme. La simulation et l'expérimentation en temps réel sur le robot prototype Haedus ont permis de valider la méthode présentée.

WALKING CONTROL OF HAEDUS
A QUADRUPED DYNAMIC ROBOT

Patrick Lessard

ABSTRACT

Research on the different aspects in the design and development of walking robots has been reported only in the past 25 years. The literature review included in this thesis clearly shows that the gait and walking problems with related control techniques are the key issues in the robot design and construction. A method is developed and presented in this research thesis to address the walking and gait analysis and the associated motion control. The controller is defined and formulated based on a robust dynamic model of the robot. Sets of computed torques are exerted on the predefined trajectory of the model and the required forces and torques are derived to produce the various leg and body movements. The force and torque distributions at the joints are solved by a pseudo-inverse matrix calculation. Two transformations on the original joint space dynamic model are applied to simplify and to obtain the inverse dynamic solutions. The first transformation allows control of the robot in the working space by a coordinate system shift. The second transformation is based on the hypothesis that the legs touching the ground are under constraint and should support the body, hence preserve the boundary conditions. This consideration allows reduction in the complexity of the model and permits handling each dynamic configuration of the legs in an independently controllable fashion. Finally, the proposed approach also simplifies the conception of the controller and reduces the computational complexity. Validation of the control scheme has been demonstrated through simulation and real time laboratory experimentation with the prototype robot - Haedus. The approach presented gives an elegant technique for the design of motion control scheme for any quadruped walking machine and the results support the validity of the method proposed.

AVANT-PROPOS

Jusqu'à aujourd'hui, aucun robot marcheur à lui seul n'a démontré la capacité de marcher sur un terrain accidenté, de porter une charge importante et de courir à une bonne vitesse tout en maintenant son équilibre en plus de trouver son propre chemin. Ce sont les objectifs qui furent définis pour Capra[1], un robot marcheur dynamique et autonome.

Capra, c'est une plate-forme marchante dont les caractéristiques sont communes à plusieurs applications. Capra pourra servir entre autre à la défense nationale, pour des missions de reconnaissance et de surveillance ou encore, aux escouades tactiques de la Sûreté du Québec pour les interventions lors d'attentats à la bombe. Il pourrait aussi être assigné à des opérations de repérage et de sauvetage lors d'un incendie ou d'un désastre comme un déversement de produits chimiques, enfin tout types de catastrophes présentant un danger pour l'humain. L'intérêt de développer une plate-forme disposant d'autant de mobilité capable de servir dans ce genre de missions, a beaucoup

augmenté depuis l'événement du 11 septembre dernier. Également, la grande mobilité de Capra permettra de travailler en forêt ou dans les grandes canalisations souterraines des villes. Le futur des robots mobiles comme Capra est assuré, c'est pourquoi les méthodes de développement comme celle présentée dans ce travail de recherche prendront de plus en plus d'importance, notamment lorsqu'il s'agit de sécurité au niveau du public.

Afin d'atteindre les objectifs de Capra, diverses technologies doivent être maîtrisées. Entre autre, un bon système de coordination des pattes. Le contrôleur doit être en mesure de réaliser une démarche fluide et efficace énergétiquement, autant pour la marche que pour la course. Vu l'ampleur du défi, Haedus[2], un prototype simplifié de Capra, a été conçu. Cette étude préliminaire a permis de valider l'approche utilisée pour réaliser le système de commande présenté dans cet ouvrage.

---

[1] Capra est tiré de l'étymologie latine et signifie chèvre. La chèvre est un animal qui se déplace avec une fascinante agilité même sur les terrains les plus escarpés. Par respect pour cet animal qui inspire une extraordinaire mobilité, nous avons donné ce nom à notre projet.

[2] Haedus signifie chevreau en latin.

REMERCIEMENT

Le temps d'une pose entre deux lignes d'écriture, un regard perdu aux petites heures d'une nuit passée à modéliser le robot ou encore l'instant d'un songe à contempler les mouvements d'Haedus me permettent de penser à vous, qui m'avez supporté, félicité et encouragé tout au long de ce travail de recherche. Ce travail est certainement le plus long que j'ai entrepris jusqu'à ce jour, mais grâce à votre support et vos encouragements, il n'est certes pas le dernier. C'est pourquoi je prends ces quelques lignes pour vous remercier :

Mes chers parents, Réjean et Suzanne Lessard, qui avez toujours cru en moi et m'avez fourni les bonnes racines qui font de moi ce que je suis aujourd'hui. Mon cher frère Simon avec qui j'ai grandi et qui est appelé à accomplir de grands projets. Ma charmante copine Marie-Joëlle qui a été d'une écoute précieuse et d'un support moral intarissable. Dr. Thiagas Sankar qui avez toujours le mot juste au bon moment pour rétablir les situations les plus critiques. Dr. Pascal Bigras qui êtes d'un support technique incomparable et d'une disponibilité enviable. Et finalement, mes chers amis et collègues qui m'avez accompagné dans cette aventure et qui avez contribué aux réalisations

de Capra. De ceux-ci, j'aimerais particulièrement citer Jean-Christophe Demers, Alexandre Doin, Simon Lupien, Patrick Frenette et Guillaume Lambert.

Je tiens aussi à souligner le support financier du FCAR, du CRSNG provenant des fonds de recherches de Dr. Thiagas Sankar. Dans la même veine, je fais remarquer l'importance du support financier et technique des commanditaires de Capra sans quoi nous n'aurions pas pu nous procurer le matériel pour la réalisation du robot. Parmi ceux-ci, je dénote particulièrement : Opal-RT qui a fait don du logiciel RT-Lab et d'un support technique hors pair et l'agence spatiale canadienne (ASC) qui a fait don de SYMOFROS et le prêt des amplificateurs. Au sein de Capra, la conception et la fabrication de Haedus a été rendu possible grâce au travail de Guillaume Lambert, Carl Blouin, Hugo Drolet, Dominic Moreau et Dominik Désilets. Enfin, je tiens à remercier Anne-Lise Moreau pour la révision de la langue.

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS ET DES SIGLES

g	Accélération gravitationnelle, m/s^2
CAD	Computer Aided Design
DDL	Degrés De Liberté
NASA	National Aeronautics and Space Administration
PID	Contrôleur Proportionnel – Intégral – Dérivé
PD	Contrôleur Proportionnel – Dérivé
ZMP	Zero Momented Point
ε	Résistance spécifique attribuée à un déplacement
M	Matrice de masses du modèle dynamique
N	Vecteur non-linéaire du modèle dynamique
$\boldsymbol{\tau}$	Vecteur de couples articulaires
H	Transformation de l'espace articulaire à l'espace cartésien
$\boldsymbol{\chi}$	Coordonnées généralisées dans l'espace Cartésien
q	Coordonnées généralisées dans l'espace articulaire
Γ	Forces généralisées symbolisées
B	Matrice de transformation entre les efforts des actionneurs $\boldsymbol{\tau}$ et les forces généralisées Γ
F	Forces appliquées sur le corps et les extrémités des pattes libres définissant le mouvement décrit par la trajectoire assignée
u	Loi de commande linéarisée

Note. Les variables présentées ci-haut sont utilisées dans le texte en combinaison avec les indices de la page suivante.

Indicateurs et indices :

$\bar{}$ Indique la transformation de l'espace articulaire à l'espace cartésien

$=$ Indique la réduction de modèle

$_c$ Indice utilisé pour définir la partie qui se rapporte au corps

$_p$ Indice utilisé pour définir la partie qui se rapporte aux pattes

$_{pl}$ Indice utilisé pour définir la partie qui se rapporte aux pattes libres

$_{ps}$ Indice utilisé pour définir la partie qui se rapporte aux pattes au sol

CHAPITRE 1

INTRODUCTION ET PRÉSENTATION DU PROBLÈME

1.1 Présentation générale du sujet d'étude

Depuis des siècles l'homme est captivé par tout ce qu'il voit et touche. Cet être pensant, cet être d'exploration tente par tous les moyens de copier la nature, si imaginative, si créative à ses yeux. Fasciné par ce qui bouge, l'homme analyse les moindres mouvements des animaux pour mieux les comprendre et s'en inspirer pour ses propres créations. L'homme a même appris au cheval de nouvelles allures après avoir tant analysé les démarches naturelles de ce quadrupède. Pourtant, jusqu'à ce jour, la démarche des animaux demeure un défi à reproduire. Les robots mobiles d'aujourd'hui évoluent de plus en plus rapidement grâce à l'avancement de la technologie. N'en demeure pas moins que l'agilité dont ils font preuve est loin d'être à la hauteur de ce que dame Nature propose.

Sans toutefois prétendre reproduire tous les mécanismes conduisant à une allure gracieuse, le travail présenté dans ce mémoire permet de résoudre une partie du problème de la démarche pour ce qui concerne l'asservissement.

La revue de littérature effectue un survol de la problématique et des principes reliés à la démarche des robots mobiles. L'étude présentée porte une attention particulière au problème de coordination des pattes, à la répartition

des efforts et à la modélisation dynamique du robot. Une partie en simulation ainsi qu'une partie en expérimentation avec comparaison accompagnent le travail de recherche. Les résultats obtenus avec l'approche préconisée dans ce travail sont discutés dans le dernier chapitre.

La méthode exposée dans cet ouvrage utilise des outils de modélisation et un environnement de programmation temps réel de haut niveau qui permettent d'alléger ce travail et d'en assurer une plus grande qualité de résultats.

1.2 Revue de littérature

La revue de littérature est structurée de façon à répondre aux trois questions suivantes : pourquoi développer le robot marcheur? Qu'est-ce qu'un robot marcheur? Et comment réaliser le contrôle d'un robot marcheur?

1.2.1 L'intérêt de développer les technologies associées aux robots marcheurs

Plusieurs motivations sont à l'origine du développement des robots marcheurs. La robotique mobile en général vise à faire accomplir par des robots certaines tâches de façon autonome. La catégorie marcheur offre une plus grande mobilité de déplacement et une apparence plus animale. Les Japonais la préfèrent particulièrement pour cette qualité, qui selon eux, leur permettra de pénétrer le marché de masse plus facilement (Mandzel 2000).

Si l'on doit soustraire l'homme aux environnements nocifs ou à hauts risques, les robots doivent pouvoir se déplacer où l'homme peut normalement accéder. L'entretien et l'inspection des équipements et la gestion des déchets à

l'intérieur des centrales nucléaires ou encore les interventions d'urgence dans les usines de produits chimiques, sont deux exemples d'applications où l'homme devrait se retrouver derrière la machine à téléopérer celle-ci afin d'éviter de s'exposer aux dangers qu'encourent son travail.

Un autre exemple d'actualité est celui des mines anti-personnel encore enterrées dans le sol de plusieurs pays. Leur nombre est estimé à 110 millions et elles causent près de 30,000 mutilations ou blessures par an. Un robot marcheur est l'outil tout indiqué pour retirer du sol ses mines selon Hirose (2000). Ce dernier affirme que le robot marcheur offre d'importants avantages comparé aux autres véhicules utilisés jusqu'à présent en raison de son mode de déplacement, de sa grande capacité d'adaptation au terrain, de la surface de sol piétiné qui est minimale par rapport à une trace continue laissée par les véhicules à chenilles ou à roues et enfin, des possibilités de se positionner pour minimiser l'impact d'une détonation accidentelle.

Ce genre d'accident chez l'homme peut conduire à la perte d'usage de ses jambes. La perte de mobilité est un handicap sérieux. Gao (1990) s'est penché sur le problème et a étudié la possibilité de concevoir une chaise marchante capable de transporter un homme. D'autres ont carrément fabriqué des véhicules tout terrain sur pattes (Mosher 1968) et (Song 1988).

Si les véhicules sur roues permettent de plus grandes vitesses de déplacement et la portance des véhicules à chenilles est plus élevée, les marcheurs offrent eux aussi autant d'avantages de leur côté lorsqu'il s'agit de prendre d'assaut les terrains largement accidentés, les escaliers ou les sols fragiles ou mous. Selon Zhou (1999), Furosh (1995), Dunn (1996), Buehler (1998) et Lethimen (1994), les robots marcheurs surpassent tout autre type de véhicule en terrain naturel ou en forêt. En plus de profiter d'une plus grande mobilité, le

marcheur n'endommage pas le sol. Motivée par cet aspect, une compagnie finlandaise nommée PlusTech inc. a développé des machineries lourdes sur pattes pour l'exploitation forestière.

À mesure que les barrières technologiques tombent, les robots mobiles sont de plus en plus accessibles et viables économiquement. Si bien qu'ils feront partis de notre quotidien en nous rendant d'innombrables services d'ici quelques années. Ramasser les déchets, livrer le courrier, surveiller, entretenir les égouts, tondre le gazon, passer l'aspirateur sont autant de tâches quotidiennes qui pourraient prendre une toute autre allure dans les années à venir. Cette réalité, la compagnie Honda l'a déjà comprise et depuis plus de 10 ans, fabrique des androïdes (Hirai 1998). Sony s'est également lancé dans une aventure en créant un robot chien pour le divertissement, puis un petit humanoïde étonnamment agile. Cette compagnie visionnaire a même mis sur pied une compétition appelée RoboCup où les chiens AIBO, offerts par Sony en échange des programmes développés par les universités participantes, se disputent une joute de soccer, Veloso (1998).

Évidemment, l'homme ne s'arrête pas à ce qui est autour de lui. Le besoin d'explorer le pousse à se lancer dans l'espace et à explorer les planètes à sa portée. Il envoie donc des robots faire la reconnaissance comme le Pathfinder envoyé sur Mars par la NASA (Matthies 1995). De son côté, l'agence spatiale européenne a développé un programme d'exploration décrit par De La Fontaine (1996) où sont spécifiés les besoins pour un robot d'exploration de surface. Un marcheur, plus spécifiquement un sauteur, serait idéal pour le sol lunaire qui est recouvert de poussière.

Sommes toutes, les applications sont nombreuses et justifient pleinement la pertinence de développer les technologies associées aux robots marcheurs.

Surtout si l'on calcule que plus de la moitié de la surface de la Terre est inaccessible par des véhicules à roues ou à chenilles (Buehler 1998).

1.2.2 Définition d'un robot marcheur

Avant d'entreprendre la mission de contrôler un robot marcheur, il incombe de bien en comprendre sa conception et sa dynamique. D'un robot à l'autre, les mécanismes et le nombre de pattes varient énormément ainsi que le nombre de degrés de liberté (DDL) par patte. Ces éléments sont la clé de la composition du marcheur et ce n'est qu'une bonne combinaison de ceux-ci qui feront l'efficacité et l'adaptabilité au terrain du robot conçu. Un facteur à ne pas oublier est le poids total du robot. En effet, le poids et la vitesse de déplacement sont les deux paramètres les plus exigeants en terme d'énergie consommée pour un robot marcheur.

Le poids est le pire ennemi alors que la vitesse est un objectif à atteindre. D'une part, en augmentant le nombre de pattes et de DDL, un robot offre plus de configurations possibles, donc une meilleure stabilité et adaptabilité au terrain. D'autre part, le poids augmentera bien d'avantage et trop souvent le gain en flexibilité sera taxé sévèrement par l'augmentation de masse.

Zhou (1999) affirme qu'il faut un minimum de trois DDL par patte pour pouvoir s'adapter à tout type de terrain en gardant une posture convenable et un bon équilibre. De façon surprenante, Buehler (1998) a démontré qu'avec un seul DDL par patte, un quadrupède pouvait marcher en sautillant et même grimper une marche.

Certaines études ont démontré que les dimensions n'avaient pas une grande influence sur le rendement énergétique si ce n'est que la longueur des pattes a un lien direct avec la vitesse de déplacement. (Margaria 1976)

Une observation sur les animaux a permis de déterminer une règle où se produisait le changement d'allure ou de démarche. Cette règle est une relation basée sur l'observation de la vitesse v en fonction de la gravité g et de la longueur totale de la patte l:

$$v = \sqrt{gl} \tag{1.1}$$

où la vitesse de marche maximale est donnée par v. Au-dessus de cette vitesse, la course devient plus efficace énergiquement (Alexander 1990). Chez les animaux étudiés, à cette vitesse critique v, un changement de cadence a lieu, soit le passage de la marche à la course. Comme cette vitesse est propre à chaque animal, un chien peut avoir à courir pour suivre un cheval au trot par exemple. Une étude intéressante a été faite sur les animaux pour comprendre l'effet de la taille par McMahon (1975).

Pourquoi existe t'il une transition entre la marche et la course ? Ces deux allures ont des comportements très différents (Doerschuk 1996). La marche se présente comme un problème de pendule inversé et la course comme un problème de masse – ressorts, affirme Dickinson (2000).

Pour bien comprendre la dynamique de la patte, il faut décomposer le mouvement en analysant les énergies. Hirose (1984) a présenté une relation empirique qui permet d'évaluer le rendement énergétique d'un robot marcheur. La résistance spécifique ε est cet indice d'efficacité qui permet de comparer différentes configurations de robot marcheur de différentes tailles :

$$\varepsilon = \frac{E}{WL} \tag{1.2}$$

où ε est la résistance spécifique, une valeur sans unité, E est l'énergie consommée pour la période étudiée, W est le poids du robot et L est la distance parcourue durant la même période. Plus la résistance spécifique ε est faible, meilleur est le rendement. Cette formule a été utilisée par Lupien (2001) dans son étude sur les configurations de robots marcheurs et la planification des trajectoires de pattes.

La minimisation de l'énergie semble avoir été une considération importante lors de l'évolution animale remarque Alexander (1990). Le diagramme suivant permet de mieux comprendre comment l'énergie est dépensée pour produire un mouvement en fonction du mécanisme de patte choisi.

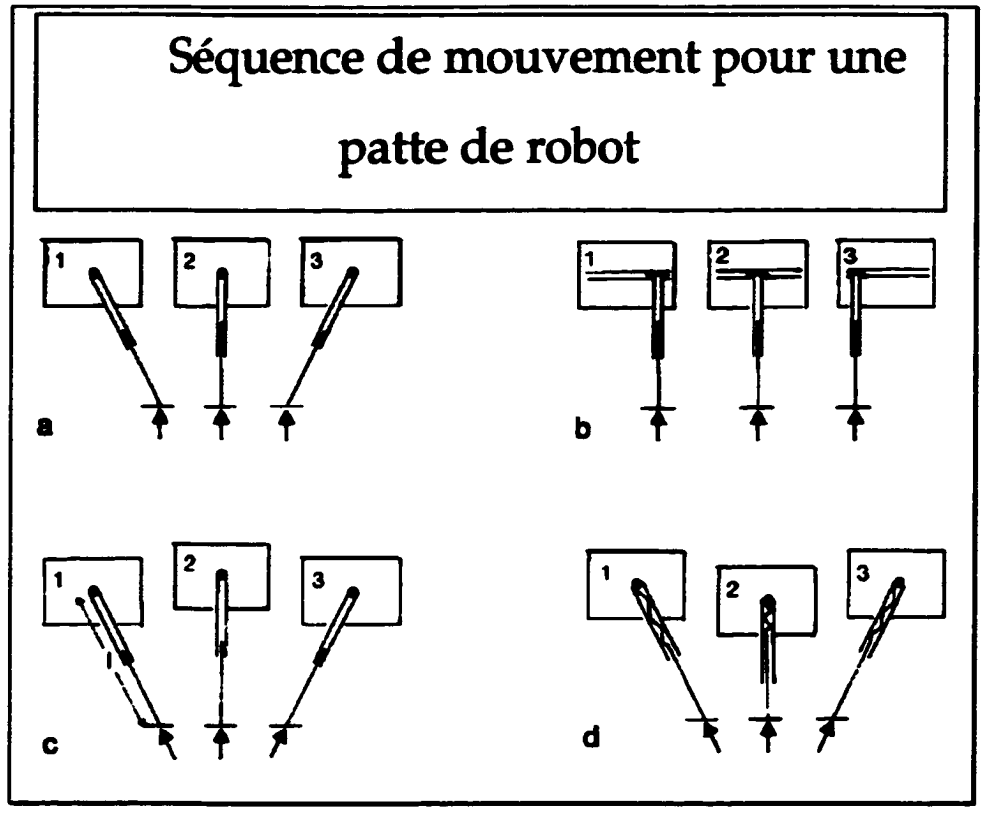


Figure 1.1 Mouvements de pattes avec le vecteur force associé.

Les flèches de la figure 1.1 indiquent la direction des forces sur l'extrémité des pattes. La figure 1.1 est tirée de Alexander (1990).

Le mécanisme présenté à l'illustration a) déplace le corps à une vitesse constante et une orientation constante. Les énergies cinétiques et potentielles sont donc préservées. Ce qui en premier lieu apparaît économique, puisqu'il n'y a aucune accélération. Cependant, un autre concept doit être introduit pour comprendre réellement comment l'énergie est dépensée. Entre l'état 1 et l'état 2, l'actionneur effectue un travail positif. C'est-à-dire que l'énergie dépensée est utilisée pour produire le mouvement. En terme de force, le couple appliqué par

l'actionneur sert à combattre les forces de gravité. Dans la phase suivante, de 2 à 3, le travail est négatif. L'actionneur doit freiner le mouvement naturel engendré par la gravité. Ce qui constitue une perte énergétique.

Le mécanisme présenté en b) permet d'effectuer le même mouvement de corps avec une dépense d'énergie comparable à un véhicule sur roues. Cela est possible grâce au découplage des forces verticales et horizontales. En effet, de la phase 1 à la phase 3, aucune énergie n'est dépensée pour combattre la gravité. Ce mécanisme peut être un mécanisme prismatique comme utilisé par les « beam walkers » que l'on appelle aussi « tables marchantes » ou encore un mécanisme à quatre membrures appelé pantographe. Celui-ci a été largement utilisé dans plusieurs marcheurs (Gao 1990), (Lin 1993), (Hirose 1984), (Hong 1999), et par le robot Hydraumas III de l'École de technologie supérieure. Bien qu'il soit le plus utilisé jusqu'à présent, ce mécanisme comporte cependant quelques inconvénients. Au delà d'une certaine cadence de mouvement, son rendement énergétique diminue considérablement (Hirose 1984).

Le mécanisme présenté en c) est une autre alternative de mouvement qui théoriquement ne requiert aucune énergie. En effet, aucun couple n'est nécessaire entre 1 et 3 si les pertes en friction sont négligeables. Le mouvement se fait autour de la hanche et la longueur de la patte demeure constante. Résultat : l'énergie cinétique varie inversement proportionnellement à l'énergie potentielle. Cependant, Ceci est vrai seulement à des vitesses très faibles. Dès que la vitesse dépasse le seuil critique v calculée en 1.1, le corps est projeté dans les airs et les pattes quittent le sol, ce qui rend alors la marche impossible. Un autre problème se présente entre l'état 3 et l'état 1 où un changement de vitesse doit se faire instantanément. Le vecteur de vitesse du corps change d'orientation instantanément !

Le mécanisme présenté en d) est beaucoup plus prometteur pour atteindre de grandes vitesses de déplacement. Un mécanisme équivalent à un ressort installé à l'intérieur de l'articulation en translation dans la patte sert à accumuler l'énergie cinétique durant la phase 1 à 2 pour la redonner au corps pendant la phase 2 à 3. Mises à part les pertes en friction, l'énergie est conservée. Le robot quadrupède de Raibert (1986) est conçu de cette façon. Ce principe se retrouve dans la nature chez les mammifères comme les kangourous, la plupart des quadrupèdes et les humains. Buehler a utilisé ce type de mécanisme dans le scout II (Buehler 2000).

Profitant des effets de récupération d'énergie à l'aide de mécanismes à ressorts, Buehler avec le Monopod II, prétend avoir réalisé la patte de robot la plus efficace au monde (Ahmadi 1999). Alexander (1990) soutient également qu'il est possible de faire des véhicules sur pattes aussi efficaces que les véhicules à roues. Avec l'approche qu'il propose, l'utilisation des ressorts est indispensable pour atteindre de grandes vitesses de déplacement.

Une autre notion connexe à la démarche est la posture. La posture décrit la configuration initiale des mécanismes de pattes ou la position moyenne que le robot prendra lors de la démarche. Les différentes postures retrouvées chez les quadrupèdes sont présentées par la figure 1.2.

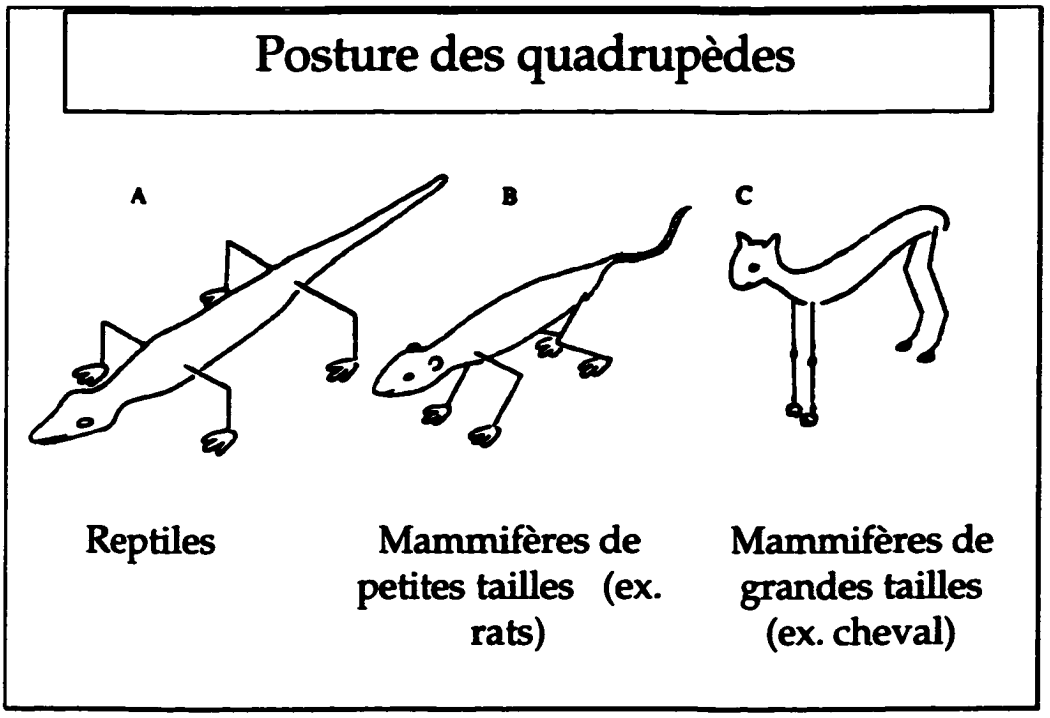


Figure 1.2 Postures des quadrupèdes.

Au moment de la marche, la posture telle que présentée par Hong (1999), joue un rôle très important au niveau de l'équilibre. La posture que le robot prendra lui permettra par exemple de garder le corps relativement horizontal même si le robot se retrouve dans une pente. La hauteur du corps par rapport au sol permet également de placer le centre de masse de façon plus avantageuse. Finalement, la posture doit également maximiser l'espace de travail de chacune des pattes dans le but de donner le maximum de liberté de mouvement.

Le quadrupède offre aussi plusieurs démarches. Cette diversité permet d'offrir une plus grande mobilité dans une vaste gamme de situations.

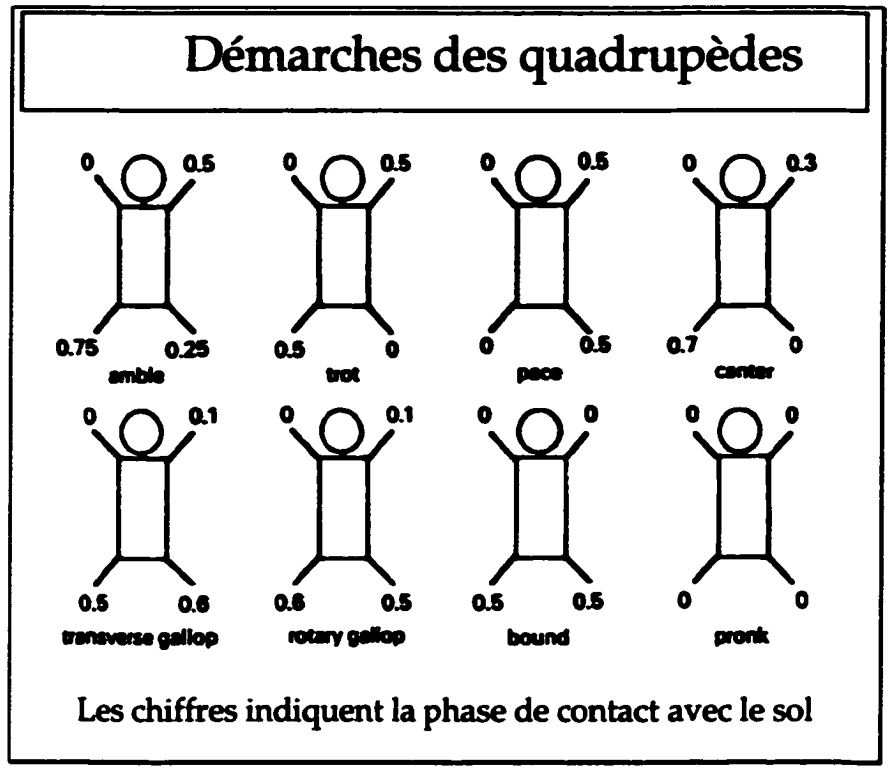


Figure 1.3 Démarches des quadrupèdes

Parmi ces démarches, seule la démarche « Amble » est stable statiquement. Pour qu'une démarche soit statiquement stable, il faut que la projection du centre de masse du robot demeure en tout temps à l'intérieur d'un polygone convexe formé par les pattes portantes. Un minimum de trois pattes au sol est donc nécessaire en tout temps.

Voici une autre façon de représenter les démarches des robots appelées « footfall patterns » (configuration pattes atterrissant au sol) :

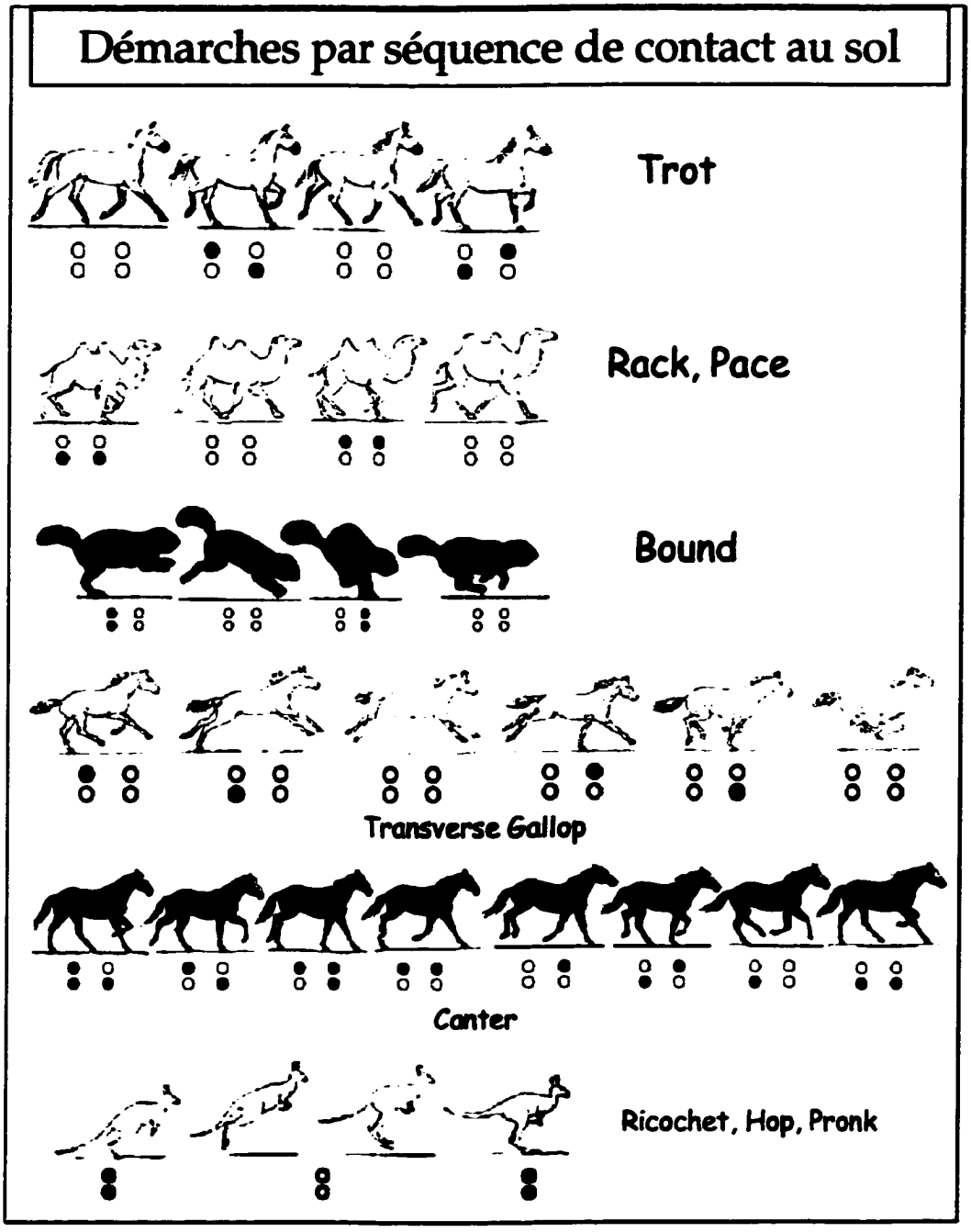


Figure 1.4 Empreintes des différentes démarches les plus connues

Il est bien établi que pour obtenir une bonne vitesse de déplacement et un rendement énergétique intéressant, la dynamique du robot doit être considérée dans la conception du contrôleur. (Inagaki 1993), Furosho (1995), (Yoneda 1996).

1.2.3 Réalisation d'un contrôleur

La réalisation du système de coordination des pattes d'un robot marcheur impose de jumeler deux consignes de mouvement à la fois : la direction du corps associée à une trajectoire assignée (position, vitesse, orientation) et l'équilibre (accélération angulaire du corps devant rester à l'intérieur d'une plage déterminée par la dynamique du robot).

Deux grandes approches offrent des solutions pour contrôler les robots marcheurs à l'heure actuelle. L'approche classique où le contrôleur est développé à partir d'un modèle mathématique et l'approche basée sur l'intelligence artificielle. Plusieurs soutiennent que l'approche classique ne permet pas de résoudre avec grâce le problème de démarche si complexe et si difficile à modéliser. (Brooks 1991), (Kimura 1998).

Pratt (1998) suggère que le contrôle conventionnel amène des équations de modélisations difficiles à développer et un contrôleur complexe. Il dit que cela n'exploite pas la dynamique naturelle et donc que le robot est moins efficace, car le suivi de trajectoire donne souvent une allure artificielle. Pourtant, son approche limite le robot à une vitesse dépendante des paramètres du robot et de certaines particularités mécaniques comme les forces de friction. Celle-ci n'est pas modélisée et ne doit donc pas être trop importante pour ne pas influencer la dynamique du robot. Aussi, pour ajuster les paramètres de marche, il faut un ajustement expérimental qui peut cependant être automatisé selon lui.

Juang (1998) et Kimura (1998) justifient leurs approches de la même façon. Ceux-ci, en se basant sur Brook (1991), ajoutent que les méthodes nécessitant de grandes séries de calculs comme le contrôle traditionnel avec planification deviennent inefficaces lors de déplacements à grandes vitesses. Portés vers la biologie, ces chercheurs s'inspirent des mécanismes biologiques de la nature pour reproduire la démarche de leurs créatures robotiques. Des études faites par les neurologues sur les animaux (Matsuoka 1987) ont conduit à l'élaboration de modèles de réseaux de neurones capables de reproduire les mécanismes de marche, Pearson(1991). Beer (1991) a reproduit une coquerelle dont les comportements et les mécanismes sont issus des études de l'insecte. Les résultats démontrent bien que les hypothèses de fonctionnement de cet insecte sont justes. Pourtant, malgré cette forte tendance aux réseaux de neurones (Snaith 1991), (Kimura 1998), (Berns 1999), (Berkemeier 1999), algorithmes génétiques (Lewis 1992) et logique floue (Juang 1998), bien peu ont réussi jusqu'à présent mentionne Kimura (2000) à faire marcher un robot avec ces approches. Kimura (2000) est un des rares à avoir réussi à ce jour à publier des résultats provenant d'expérimentations concrètes. Ses résultats sont pour les moins surprenants , surtout ses vidéos du robot Patrush, autant pour la version marcheur que sauteur. Néanmoins, pour obtenir de bons résultats, plusieurs paramètres doivent être ajustés expérimentalement pour chaque marche et chaque terrain. Les chercheurs oeuvrant à la réalisation de ces contrôleurs sont très optimistes, mais il reste encore beaucoup à faire pour atteindre un niveau d'applicabilité et de fiabilité suffisant pour une application hors laboratoire.

L'approche classique, quant à elle, a fait ses preuves par le passé (Takanishi 1985), (Raibert 1986), (Chevallereau 1997), (Hirai 1998) et est encore utilisée aujourd'hui. Cette approche nécessite la conception d'un modèle dynamique dont les paramètres sont parfois difficiles à établir. Certaines

techniques permettent d'arriver aux résultats plus aisément, mais les paramètres restent essentiels à moins d'utiliser un estimateur comme le suggère les contrôleurs adaptatifs. Les robots conçus et fabriqués au MIT possèdent un modèle mathématique Raibert (1986). Un avantage inhérent au modèle est qu'il permet de simuler le robot et de valider le concept avant la fabrication, évitant ainsi plusieurs essais empiriques coûteux selon Bruneau (1998). La simulation permet également de valider le contrôleur avant les essais expérimentaux sur le robot réel. Le modèle sert aussi à développer le contrôleur, calculant ainsi les couples nécessaires pour produire le mouvement désiré. Parfois certaines simplifications du modèle s'imposent pour réduire la complexité du contrôleur.

Malgré le succès de ces méthodes traditionnelles, le problème n'est pas entièrement résolu. Cherchant de nouvelles solutions, la tendance actuelle se porte vers des contrôleurs dits « intelligents » ou dotés de fonctions d'apprentissage (Berkemeier 1999). Pourtant ceux-ci ne sont pas requis à tous les niveaux. Un robot marcheur a besoin d'adaptabilité et de robustesse. Une bonne conception devrait comprendre une base conventionnelle bien établie et une couche supérieure évolutive ou dite « intelligente ». Ce qui permettrait de marier le meilleur des deux mondes. M'sirdi (1998) affirme que réaliser un contrôleur de robot marcheur où le robot et l'environnement sont modélisés est difficile pour une démarche rapide dans un environnement complexe. Dans ce sens, pour palier aux impondérables, réduire la complexité du contrôleur et augmenter l'adaptabilité au terrain, l'implantation de réflexes ou de comportements est une avenue offrant de grandes possibilités. D'ailleurs Brooks (1991) a développé une approche de contrôle complète basée sur les comportements qu'il applique à des robots insectoïdes. Ces comportements donnent une forme « d'intelligence », une sorte de réaction aux différentes situations.

La gestion de l'équilibre doit se faire en même temps que la marche, mais d'une façon différente, car si la marche se réalise par une trajectoire planifiée, la perte d'équilibre ne l'est pas. Une solution pour maintenir l'équilibre est d'utiliser les réflexes en parallèle avec les consignes de marche. Kimura (2000) explique comment il a implanté les mécanismes de réflexes qu'il utilise sur le robot Patrush. L'équilibre, en fait, est la capacité de garder le contrôle de la position et de l'orientation du corps en présence de forces externes. Mais comment savoir si le robot est en déséquilibre ? Au niveau statique, les équations sont simples, la projection du centre de masse doit rester à l'intérieur du polygone formé par les pattes portantes. Mais lorsque le robot est en mouvement, cette méthode n'est plus valide en raison de la présence des forces externes. Une méthode appelée « Zero Momented Point » (ZMP) a été développée pour tenir compte de ces effets et déterminer la stabilité des robots marcheurs. Le ZMP se définit comme le point où la somme des moments engendrés par les forces externes et la somme des moments engendrés par la masse du robot sont égales à zéro (Kang 1997). Cette méthode est largement utilisée depuis longtemps (Takanishi 1985), (Kang 1997), (Hirai 1998), (Juang 1998), Miyashita (1998).

1.3 Problématique

La problématique à résoudre pour ce qui concerne la partie asservissement qui est présentée dans cet ouvrage consiste à coordonner les efforts des pattes portantes pour une trajectoire assignée du corps et des pattes libres.

La dynamique d'un robot marcheur est très complexe. Certains chercheurs ont négligé la dynamique des pattes par le passé pour simplifier le

problème (Miura 1984, Takanishi 1985). Mais aujourd'hui, il apparaît évident que la dynamique des pattes doit être incluse dans le modèle du robot. Du moment que la masse des pattes prend de l'importance par rapport au corps et que la cadence des mouvements augmente, l'influence de celles-ci peut rendre le robot instable. Aussi, lorsqu'une patte est au sol et porte le robot, son comportement diffère largement avec la situation que l'on appelle patte libre.

Pour bien gérer cette complexité et faire en sorte qu'un contrôleur de haut niveau puisse coordonner les mouvements du robot, une couche de contrôle doit prendre en charge l'asservissement de la dynamique du corps et des pattes.

1.4 Objectif visé

L'objectif de ce projet est de réaliser un système de contrôle de démarche dynamique appliqué à un robot marcheur, permettant de suivre un parcours assigné ou de prendre une position précise sur ses pattes. Une attention particulière sera portée au rendement énergétique du déplacement du robot. L'implantation du système de démarche sur le robot Haedus permettra de valider l'approche proposée.

1.5 Objectifs spécifiques :

1. Étudier différentes approches proposées dans la littérature scientifique sur le contrôle de robot marcheur;
2. Étudier des lois de commandes qui vont permettre d'assurer un suivi de trajectoire. Les trajectoires devront être préalablement définies de sorte à coordonner la démarche du robot et d'en assurer sa stabilité;
3. Modéliser de façon adéquate un robot marcheur;
4. Pouvoir positionner le robot dans une configuration précise et stable;

5. Réaliser une démarche statiquement stable;
6. Valider le système pour un contrôle temps-réel sur un robot réel.

1.6 Contraintes

1. Les mouvements ne doivent jamais être discontinus, car ceux-ci risquent de déstabiliser le robot;
2. Les algorithmes du système de commande doivent s'exécuter à une vitesse suffisante pour être utilisés en temps réel ;
3. Les mouvements d'Haedus étant tous dans le même axe, le modèle sera contraint dans un plan.

1.7 Hypothèses de départ

1. Chaque patte est équipée d'un capteur pouvant détecter le contact avec le sol;
2. La position des articulations est connue en tout temps pour les huit articulations;
3. Le corps est manipulable dans la plage demandée par la trajectoire. Ce qui veut dire que pour une vitesse de corps donnée, une vitesse atteignable existe pour les articulations qui le supportent ;
4. Les forces de contacts entre les pattes et le sol demeurent à l'intérieur du cône de friction. Ce qui veut dire que les pattes ne glissent pas.

1.8 Méthodologie

Le chemin logique à prendre pour obtenir un contrôleur robuste, efficace et capable de s'adapter à différentes situations imprévues est de concevoir un modèle dynamique du robot et de bâtir un contrôleur sur celui-ci. Ce contrôleur prendra en charge l'asservissement des actionneurs, la cinématique et le contrôle de la dynamique du robot puisque celle-ci est connue et représentée par le modèle dynamique et les techniques pour obtenir ces relations sont bien établies. Par la suite, se grefferont une planification de trajectoire et des

mécanismes d'adaptation avec comportements et gestion d'équilibre avec réflexes. Ces derniers qui ne sont pas présentés dans cet ouvrage devraient donc permettre le lien avec l'environnement, complexe et difficile à modéliser avec certitude.

L'approche proposée dans cet ouvrage, est une approche de contrôle conventionnel basée sur la modélisation dynamique du robot. Le modèle servira à la simulation et à concevoir un contrôleur pour l'asservissement des axes du robot. Le contrôleur prendra également en charge le suivi de trajectoires qui s'effectue dans l'espace de travail par l'intermédiaire d'une transformation de coordonnées. Les trajectoires seront ainsi plus simples à résoudre et plus représentatives. De plus, une réduction de modèle permettra de résoudre les contraintes en temps réel sans avoir à changer de modèle pour chaque configuration de pattes au sol.

Évidemment, le modèle ne peut être parfait, en raison des phénomènes variables temporellement. Pour n'en citer qu'un exemple : la friction change en fonction de la température, de l'usure des pièces en contact ou de la charge transportée. Par contre, le modèle peu être assez fidèle pour permettre au contrôleur de compenser les écarts entre le modèle et la réalité. Les paramètres du modèle peuvent être obtenus par identifications et par le dessin du robot qui a été conçu à l'aide d'un logiciel de conception mécanique *Computer Aided Design* (CAD). Ainsi, il est facile d'obtenir avec précision les distances entre les axes, les centres de masse ainsi que les masses et les inerties de chacun des membres.

Pour ce qui est du développement du modèle, nous avons opté pour SYMOFROS qui est un logiciel de modélisation symbolique présenté à la section 2.9. Ce dernier permet de concevoir et de modifier rapidement des modèles de

robots complexes tel que Haedus ou même le bras de la navette spatiale canadienne. L'expérimentation du contrôleur en simulation ou en temps réel demande beaucoup de travail, lequel est allégé par RT-LAB, un environnement de travail temps-réel intégrant le modèle calculé par SYMOFROS, la simulation et le robot réel en même temps.

Enfin, Haedus a été développé pour l'expérimentation. Ce robot quadrupède comporte deux DDL (degrés de liberté) par patte, actionnés par des moteurs électriques. Ses mouvements sont contraints dans un plan vu sa simplicité mécanique. Malgré cette limitation, il sera en mesure de valider l'approche qui pourra éventuellement être implantée dans tous types de robots marcheurs respectant les hypothèses de départ cité en 1.7.

Cet ouvrage présente l'analyse, le développement et les résultats du système de démarche implanté dans le programme de contrôle utilisé avec Haedus, un robot marcheur quadrupède expérimental.

1.9 Conclusion

Ce chapitre présentait les différentes approches utilisées pour la conception de contrôleur appliquées aux robots marcheurs ainsi que les différents problèmes associés la démarche. Il a aussi été question des hypothèses de départ sur lesquelles le présent travail s'appuie. Les prochains chapitres démontrent comment le contrôle de la démarche peut être résolu à partir d'un modèle mathématique.

CHAPITRE 2

MODÉLISATION MATHÉMATIQUE

2.1 Introduction

La modélisation permet de représenter à l'aide d'outils mathématiques le comportement d'un système robotique basé sur ses paramètres physiques. Plus ces derniers sont précis, plus la modélisation est exacte et plus la simulation qui en découle est fidèle à la réalité. Le modèle du robot présenté dans cette section comporte trois parties : la cinématique, la cinématique différentielle et la dynamique du robot.

Un modèle cinématique permet d'établir une relation entre la position des extrémités du robot et la position de chacune de ses articulations. De façon similaire, la cinématique différentielle permet de faire la relation entre l'espace articulaire et l'espace cartésien pour les vitesses, les accélérations, les forces et les couples. Enfin, le modèle dynamique permet de calculer les couples nécessaires pour produire un mouvement. Une fois que le modèle dynamique est connu, il est possible de simuler le système et de le contrôler en utilisant les informations et les équations propres au modèle.

Modéliser un système dynamique est une tâche laborieuse et nécessite des outils de calcul informatiques. Vu le grand nombre d'opérations et la complexité des équations, il est facile d'introduire des erreurs dans le modèle en le développant. Pour ces deux raisons, le logiciel SYMOFROS a été choisi

comme outil de génération de modèle. D'une part, il permet d'obtenir et de modifier rapidement un modèle complet d'Haedus. D'autre part, les facteurs d'erreurs de modélisation sont réduits considérablement. Le logiciel SYMOFROS est décrit en fin du présent chapitre.

2.2 Description Géométrique du robot Haedus

Haedus est composé d'un corps rigide et de quatre pattes à deux DDL chacune. Ces dernières sont actionnées par huit moteurs, soit un par articulation. Voir la fiche technique en annexe I pour plus de détails.

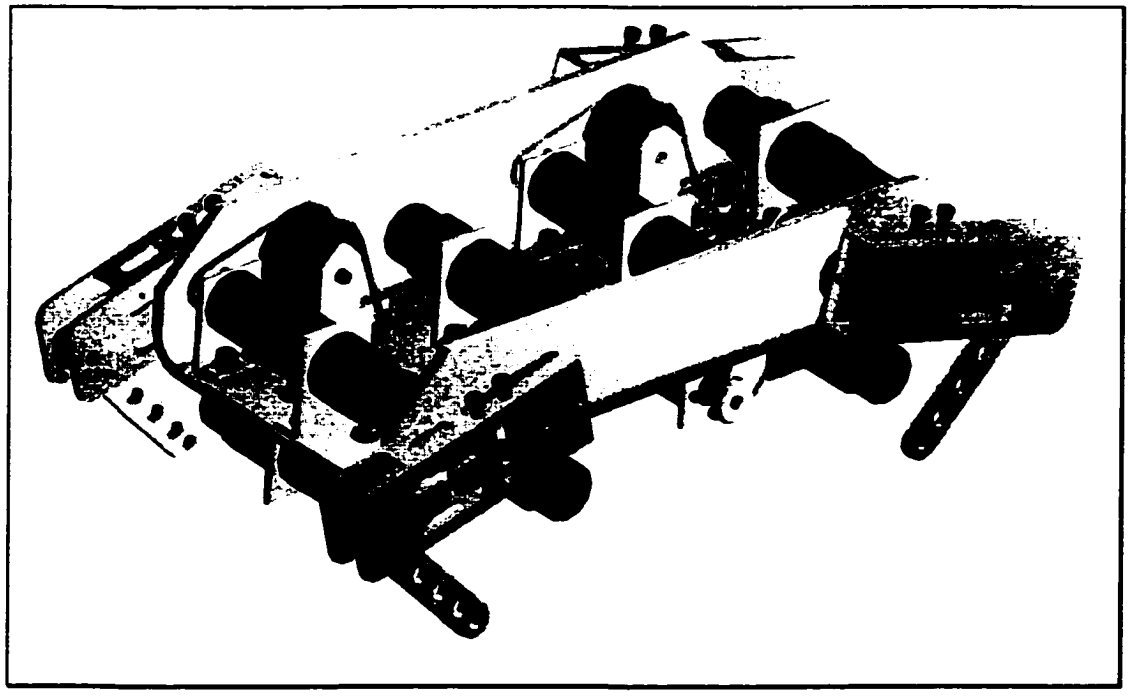


Figure 2.1 Dessin de conception d'Haedus

2.3 Cinématique

La première partie du modèle est décrite par la cinématique. La cinématique directe consiste en une transformation de l'espace articulaire (articulations du robot) à l'espace cartésien (espace de travail du robot). La position des articulations est mesurable à l'aide de capteurs présents aux articulations. Ainsi, dans l'application d'Haedus, la cinématique directe est utilisée pour déduire les trajectoires des pattes à partir des trajectoires articulaires.

Pour décrire la cinématique, une représentation matricielle peut être utilisée. Il suffit de décrire, par des matrices, tous les repères à partir de la base jusqu'à l'outil pour obtenir la cinématique directe. Ces matrices, telles que présentées par la relation (2.1), sont appelées matrices de transformation homogène. Elles permettent d'exprimer à la fois une rotation et une translation. (Craig 1989). Une matrice de transformation homogène a toujours la forme suivante :

$$ {}^{0}\mathbf{T}_{n}(\mathbf{q}) = \begin{bmatrix} {}^{0}_{n}\mathbf{R}(\mathbf{q}) & {}^{0}\mathbf{P}_{n}(\mathbf{q}) \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix} \qquad (2.1) $$

où ${}^{0}\mathbf{P}_{n}$ est le vecteur de positions [x,y,z] du repère {n} par rapport au repère origine {o} en fonction des positions des articulations $\mathbf{q}$ et ${}^{0}\mathbf{R}_{n}$ est la matrice de projection du repère {n} dans le repère {0}.

De façon générale, la cinématique peut s'exprimer sous la forme compacte suivante:

$$ \chi = \mathbf{K}(\mathbf{q}) \qquad (2.2) $$

où **K** est un vecteur de fonctions, **q** est le vecteur des positions articulaires du robot et χ est le vecteur de positions / orientation définit dans l'espace de travail représenté comme suit :

$$\chi = \begin{bmatrix} \mathbf{x} \\ \boldsymbol{\Theta} \end{bmatrix} \tag{2.3}$$

Les trois DDL exprimant la position sont représentés par le vecteur **x**, tandis que les DDL exprimant l'orientation sont représentés par $\boldsymbol{\theta}$. Le vecteur de positions et d'orientations peuvent exprimer jusqu'à trois DDL chacun selon la définition du système modélisé. Le vecteur de position **x** prend généralement la forme [x,y,z] tandis que le vecteur d'orientation peut prendre diverses formes dont voici quelques exemples: soit un vecteur $[\theta_x, \theta_y, \theta_z]$ que l'on nomme généralement roulis, tangage et lacet (convention RPY) qui est une des 24 configurations possibles utilisant les angles d'Euler, soit les paramètres d'Euler qui sont au nombre de quatre, soit les quaternions ou encore, la matrice de rotation.

Ces différentes formes comportent chacune leurs avantages. Pour une question de représentation, le modèle d'Haedus qui comporte une seule orientation (θ_z), est représenté avec la convention RPY.

2.3.1 Formalisme de Denavit-Hartenberg

Pour calculer la cinématique directe, la méthode généralement utilisée est basée sur les paramètres de Denavit-Hartenberg telle que présentée par Spong(1989) . La figure 2.1 illustre le modèle d'une patte d'Haedus qui peut être vu comme un manipulateur robotique 2 axes rotoïdes.

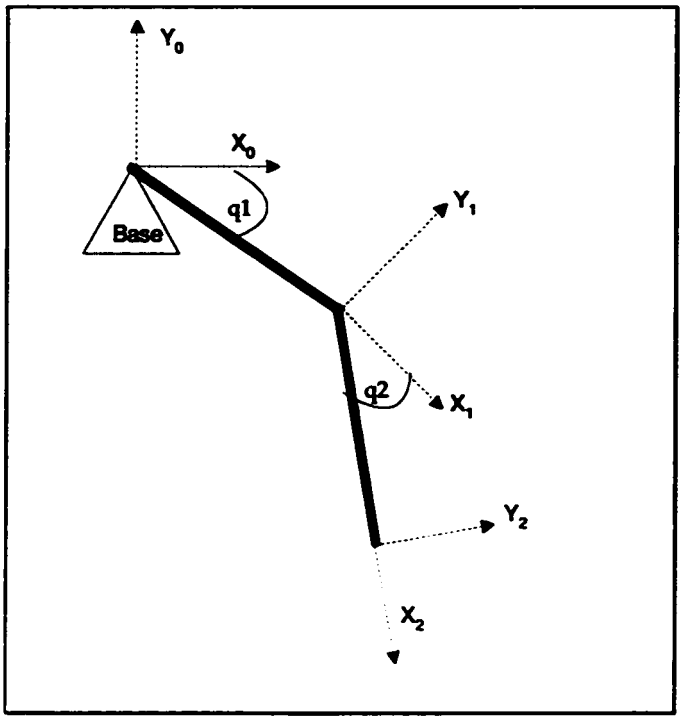


Figure 2.2 Représentation des systèmes d'axe d'un manipulateur 2 DDL

Pour modéliser un système selon la convention de Denavit-Hartenberg, il faut d'abord définir les référentiels sur chacune des articulations. L'étape suivante permet de caractériser chacun des segments et chacune des articulations avec ces quatre paramètres :

a_i = distance minimale entre z_i et z_{i+1} mesurée dans la direction x_i;
(cette mesure représente la longueur du segment)

α_i = angle entre z_i et z_{i+1} mesuré par rapport à x_i;
(courbure du segment i);

d_i = distance entre x_{i-1} et x_i mesurée dans la direction de z_i;
(ou variable de mouvement pour une articulation prismatique)

q_i = angle entre x_{i-1} et x_i mesuré par rapport à z_i;

(ou variable de mouvement pour une articulation rotoïde)

La valeur a_i ne peut être que positive puisqu'il s'agit d'une distance. Les autres peuvent prendre des valeurs positives et négatives. Les matrices de transformation homogène décrivant le passage d'un référentiel d'axe à un autre sont formées suivant la convention décrite par la matrice suivante :

$$ {}^{i-1}T_i = \begin{bmatrix} \cos(q_i) & -\sin(q_i)\cos(\alpha_i) & \sin(q_i)\sin(\alpha_i) & a_i\cos(\alpha_i) \\ \sin(q_i) & \cos(q_i)\cos(\alpha_i) & -\cos(q_i)\sin(\alpha_i) & a_i\sin(\alpha_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2.4) $$

où les paramètres a_i, α_i, d_i, et q_i sont définis plus haut.

2.3.2 Paramètres de Denavit-Hartenberg d'une patte de Haedus:

La description d'une patte d'Haedus illustrée par la figure 2.2 à partir de la hanche du robot jusqu'au sabot est décrite par le tableau ci-contre. Il n'y a que deux segments et deux articulations, lesquels sont identifiés comme deux DDL.

Tableau 2.1

Tableau des paramètres de Denavit-Hartenberg pour une patte d'Haedus

Lien	q_i	d_i	a_i	t_i
1	q_1	0	L_1	0
2	q_2	0	L_2	0

À partir de ces paramètres et de la matrice (2.4), les matrices de transformation homogène suivantes sont établies et respectent la structure présentée en (2.1) :

$$ {}^{1}T_{0} = \begin{bmatrix} \cos(q_1) & -\sin(q_1) & 0 & L_1\cos(q_1) \\ \sin(q_1) & \cos(q_1) & 0 & L_1\sin(q_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.5} $$

$$ {}^{2}T_{1} = \begin{bmatrix} \cos(q_2) & -\sin(q_2) & 0 & L_2\cos(q_2) \\ \sin(q_2) & \cos(q_2) & 0 & L_2\sin(q_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.6} $$

La multiplication de ces matrices donne la transformation de repère entre la base et l'extrémité de la patte en fonction des positions des articulations.

$$ {}^{2}T_{0} = \begin{bmatrix} \cos(q_1+q_2) & -\sin(q_1+q_2) & 0 & L_1\cos(q_1)+L_2\cos(q_1+q_2) \\ \sin(q_1+q_2) & \cos(q_1+q_2) & 0 & L_1\sin(q_1)+L_2\sin(q_1+q_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.7} $$

Ces matrices ont été générées à l'aide d'un programme développé sur Maple. Il est possible d'obtenir les mêmes résultats avec SYMOFROS. Voir la section 2.9.

2.3.3 Cinématique inverse

La cinématique inverse permet de connaître la position désirée de chacune des articulations à partir de la trajectoire à suivre. Il est à noter que les

calculateurs symboliques n'arrivent habituellement pas à isoler le vecteur de coordonnées généralisées articulaires du système d'équations décrivant la cinématique. Ces opérations sont généralement faites à la main ou à l'aide d'outils d'approximations numériques. Wang (1991) présente une de ces approches numériques.

Pour déterminer la transformation, il suffit d'exprimer la cinématique directe sous forme compacte telle que vue en 2.2 et de résoudre le système d'équations pour extraire les angles.

La cinématique inverse est utilisée pour calculer la position initiale du robot Haedus.

2.3.3.1 Cinématique inverse pour une patte d'Haedus

Le cas d'Haedus est assez simple, les angles q_1 et q_2 correspondants aux coordonnées généralisées articulaires doivent être isolés du vecteur de position **x** présenté en 2.3 ou directement du vecteur de position **P** de la matrice de transformation homogène 2.1.

Selon la relation 2.7, la forme compacte de la cinématique d'une patte d'Haedus est définie comme suit :

$$\chi = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} L_2\cos(q_1 + q_2) + L_1\cos(q_1) \\ L_2\sin(q_1 + q_2) + L_1\sin(q_1) \end{bmatrix} \tag{2.8}$$

où χ est le vecteur des coordonnées généralisées représentant la position de l'extrémité de la patte. La cinématique inverse est alors obtenue en solutionnant l'équation 2.8. Ainsi, en définissant

$$C2 = \frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1L_2} \tag{2.9}$$

$$S2 = \pm\sqrt{1 - C2^2} \tag{2.10}$$

la cinématiques inverse est donnée par :

$$q_2 = \arctan 2(S2, C2) \tag{2.11}$$

$$q_1 = \arctan 2(y, x) - \arctan 2(L_2S2, L_1 + L_2C2) + \pi/2 \tag{2.12}$$

2.4 Cinématique différentielle

Une relation appelée cinématique différentielle permet d'exprimer la vitesse et l'accélération des extrémités d'un système robotique en fonction de la vitesse des articulations. À partir de cette relation, la relation inverse peut être obtenue; c'est à dire la vitesse articulaire en fonction de la vitesse des extrémités. Ces relations peuvent être utilisées pour la planification de trajectoires. Il est également possible d'obtenir la force ou le couple aux articulations correspondants à une force ou à un couple appliqué aux extrémités du manipulateur. Cette même relation permettra aussi la transformation des accélérations. Pour Haedus, ces équations joueront un rôle particulièrement important pour transformer le modèle de façon à ce qu'il s'exprime en fonction des positions des extrémités des pattes.

Les notations pour décrire les vitesses linéaires et angulaires d'un corps sont tirées du livre de Craig (1989).

2.4.1 Vitesses linéaires

La relation 2.7 représente la cinématique d'un manipulateur par rapport à la base, exprimée sous forme matricielle. Sachant que **P(q)** est le vecteur de position de l'outil [x,y,z] exprimé en fonction des coordonnées généralisées **q**, la vitesse de l'outil par rapport à la vitesse du vecteur des coordonnées généralisées peut s'exprimer de la façon suivante :

$$ {}^0\mathbf{V}_{PT} = \mathbf{v}_T = {}^0\mathbf{J}_{VT}(\mathbf{q})\dot{\mathbf{q}} \tag{2.13} $$

où l'on définira **J**, la matrice Jacobienne de **P** par rapport au vecteur des positions articulaires **q**. Cette relation est exprimée par l'équation (2.14).

$$ {}^0\mathbf{J}_{VT}(\mathbf{q}) = \frac{\partial {}^0\mathbf{P}_T}{\partial \mathbf{q}} = \frac{\partial \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}}{\partial \begin{bmatrix} q_1 \\ q_2 \\ \cdots \\ q_n \end{bmatrix}} = \begin{bmatrix} \frac{\partial p_x}{\partial q_1} & \cdots & \frac{\partial p_x}{\partial q_n} \\ \frac{\partial p_y}{\partial q_1} & \cdots & \frac{\partial p_y}{\partial q_n} \\ \frac{\partial p_z}{\partial q_1} & \cdots & \frac{\partial p_z}{\partial q_n} \end{bmatrix} \tag{2.14} $$

2.4.2 Vitesses angulaires

La matrice jacobienne telle que définie par la relation (2.14) ne peut pas être utilisée pour calculer les vitesses angulaires des extrémités d'un manipulateur par rapport aux vitesses des articulations. La loi de propagation de vitesses démontre que les vitesses angulaires ne sont pas obtenues simplement en dérivant les orientations. La matrice Jacobienne ne peut alors être définie par la relation (2.14.) Il est cependant toujours possible d'exprimer les relations de vitesse sous la forme donnée par la relation suivante :

$$ {}^0\boldsymbol{\Omega}_T = {}^0\mathbf{J}_{\Omega T}\dot{\mathbf{q}} \tag{2.15} $$

Où ${}^0\Omega_T$ est la vitesse angulaire de l'outil par rapport au repère de la base et ${}^0J_{\Omega T}$ est une matrice appelée Jacobienne.

2.5 Tranformation de coordonnées

Même si le modèle d'un système robotique s'exprime souvent par rapport à ses articulations, une transformation peut toujours être appliquée pour le réexprimer par rapport à un autre ensemble de coordonnées. Ces coordonnées peuvent être choisies afin de simplifier le problème de planification de trajectoire et de commande du système.

Pour le système robotique Haedus, cette transformation de coordonnées est exprimée par la figure 2.3.

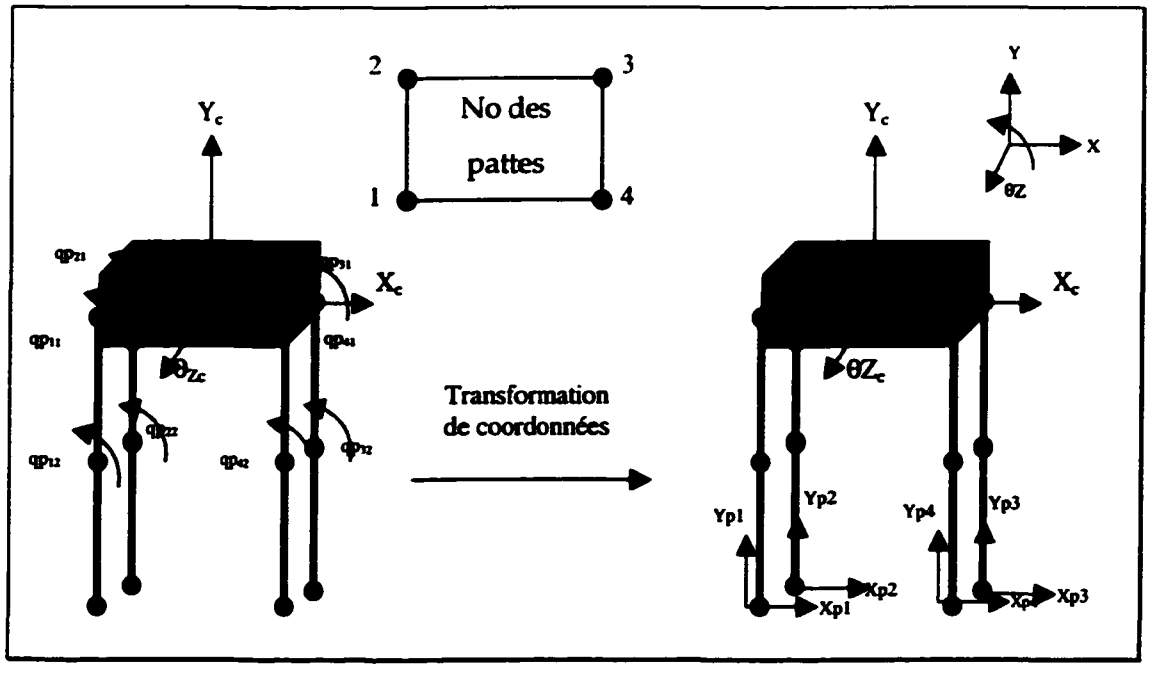


Figure 2.3 Représentation des deux systèmes de coordonnées généralisées

Cette transformation du modèle conduira à contrôler les extrémités des pattes et le corps dans l'espace cartésien plutôt que dans l'espace articulaire. Ceci procurera trois avantages : éviter d'avoir à calculer la cinématique inverse lors de la planification de trajectoire, contrôler le robot directement dans l'espace de travail et simplifier la prise en compte des équations de contraintes lors de la réduction du modèle.

Les coordonnées généralisées dans l'espace articulaire sont définies ainsi :

$$\mathbf{q} = [\mathbf{q}_c, \mathbf{q}_p]^T \tag{2.16}$$

Où $\mathbf{q}_c$ et $\mathbf{q}_p$ sont respectivement, les coordonnées généralisées du corps (c) et celles des pattes (p). Les coordonnées généralisées du corps sont absolues ou liées à un référentiel fixe. À l'opposé, les coordonnées généralisées des pattes décrivent la position des articulations des membres. Défini ainsi, le corps possède trois DDL et les pattes disposent de deux DDL chacune. Ces articulations sont identifiées comme θ_G pour le genou et θ_H pour la hanche et ce, pour les quatre pattes. Une représentation sous forme étendue donne ceci :

$$\mathbf{q}_c = [x, y, \Theta_z]^T \quad \text{et} \quad \mathbf{q}_p = [\Theta_{1H}, \Theta_{2G}, \ldots, \Theta_{4H}, \Theta_{4G}]^T \tag{2.17}$$

Les coordonnées généralisées du corps étant déjà dans l'espace cartésien, elles n'auront pas à être transformées. Ainsi $\mathbf{q}_c$ et χ_c sont identiques. Pour les pattes, deux nouvelles coordonnées sont définies : x_p et y_p, la position de l'extrémité de la patte. Les coordonnées généralisées dans l'espace cartésien sont donc données par :

$$\chi = [\chi_c, \chi_p]^T \tag{2.18}$$

où :

$$\chi_c = \begin{bmatrix} x \\ y \\ \Theta_z \end{bmatrix} \text{ et } \chi_p = \begin{bmatrix} \chi_{p1} \\ \chi_{p2} \\ \chi_{p3} \\ \chi_{p4} \end{bmatrix} = \begin{bmatrix} x_{p1} \\ y_{p1} \\ \cdots \\ x_{p4} \\ y_{p4} \end{bmatrix} \tag{2.19}$$

De la même façon, pour les vitesses généralisées obtenues à partir de la dérivée des coordonnées généralisées :

$$\dot{\chi}_c = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \omega_z \end{bmatrix} \text{ et } \dot{\chi}_p = \begin{bmatrix} \dot{\chi}_{p1} \\ \dot{\chi}_{p2} \\ \dot{\chi}_{p3} \\ \dot{\chi}_{p4} \end{bmatrix} = \begin{bmatrix} \dot{x}_{p1} \\ \dot{y}_{p1} \\ \cdots \\ \dot{x}_{p4} \\ \dot{y}_{p4} \end{bmatrix} \tag{2.20}$$

Pour calculer les vitesses généralisées, la relation (2.14) doit être utilisée, ce qui permet d'établir la relation suivante :

$$\dot{\chi}_p = \mathbf{J}\dot{\mathbf{q}} \tag{2.21}$$

Ou sous forme étendue :

$$\begin{bmatrix} \dot{x}_{p1} \\ \dot{y}_{p1} \\ \cdots \\ \dot{x}_{p4} \\ \dot{y}_{p4} \end{bmatrix} = \begin{bmatrix} \mathbf{J}_{c1} & \mathbf{J}_{p1} \\ \mathbf{J}_{c2} & \mathbf{J}_{p2} \\ \mathbf{J}_{c3} & \mathbf{J}_{p3} \\ \mathbf{J}_{c4} & \mathbf{J}_{p4} \end{bmatrix} \begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \omega_{Zc} \\ \dot{\theta}_{H1} \\ \dot{\theta}_{G1} \\ \cdots \\ \dot{\theta}_{H4} \\ \dot{\theta}_{G4} \end{bmatrix} \tag{2.22}$$

La transformation complète du système qui comprend les coordonnées généralisées du corps et celles des pattes peut finalement s'écrire ainsi :

$$\begin{bmatrix} \dot{\mathbf{X}}_c \\ \dot{\mathbf{X}}_p \end{bmatrix} = \mathbf{H} \begin{bmatrix} \dot{\mathbf{q}}_c \\ \dot{\mathbf{q}}_p \end{bmatrix} \tag{2.23}$$

où **H** est défini comme la matrice regroupant les Jacobiennes et la matrice identité.

$$\mathbf{H} = \begin{bmatrix} \mathbf{I} & 0 \\ \mathbf{J}_c & \mathbf{J}_p \end{bmatrix} \tag{2.24}$$

Ensuite, pour obtenir la transformation inverse, il suffit d'inverser la matrice **H**. L'inversion sous forme symbolique donne ceci :

$$\mathbf{H}^{-1} = \begin{bmatrix} \mathbf{I} & 0 \\ -\mathbf{J}_p^{-1}\mathbf{J}_c & \mathbf{J}_p^{-1} \end{bmatrix} \tag{2.25}$$

Ce qui permet de calculer les vitesses généralisées dans l'espace cartésien à partir des vitesses généralisées de l'espace articulaire :

$$\begin{bmatrix} \dot{\mathbf{q}}_c \\ \dot{\mathbf{q}}_p \end{bmatrix} = \mathbf{H}^{-1} \begin{bmatrix} \dot{\mathbf{X}}_c \\ \dot{\mathbf{X}}_p \end{bmatrix} \tag{2.26}$$

Enfin, pour obtenir l'accélération, il est possible de calculer la dérivée de l'équation (2.26). Ce qui donne :

$$\begin{bmatrix} \ddot{\mathbf{q}}_c \\ \ddot{\mathbf{q}}_p \end{bmatrix} = \frac{\partial \mathbf{H}^{-1}}{\partial t} \begin{bmatrix} \dot{\mathbf{X}}_c \\ \dot{\mathbf{X}}_p \end{bmatrix} + \mathbf{H}^{-1} \begin{bmatrix} \ddot{\mathbf{X}}_c \\ \ddot{\mathbf{X}}_p \end{bmatrix} \tag{2.27}$$

Finalement, cette nouvelle équation va substituer les accélérations généralisées de l'équation générale du modèle dynamique, ce qui complètera la transformation du modèle dynamique, de l'espace articulaire à l'espace cartésien. Cette transformation sera utilisée ultérieurement à la section 2.7.

2.6 Contraintes des pattes au sol

Contrairement aux robots industriels, les robots marcheurs ne sont pas constamment en contact avec le sol. Le robot peut aussi prendre plusieurs configurations : une, deux, trois ou quatre pattes au sol, voir même aucune, sur une courte période. Ce qui pose un problème de modélisation qui peut toutefois être résolu par l'emploi de plusieurs modèles ou par l'ajout de contraintes intermittentes. Ce problème peut également être simplifié grâce au changement de coordonnées introduit dans la section précédente. Les contraintes sont alors directement associées aux coordonnées généralisées des extrémités des pattes. C'est donc grâce à ce changement de coordonnées que les différentes configurations de pattes que le robot peut prendre seront prises en charge. En d'autres termes, le changement de coordonnées s'adapte à la configuration du robot. Cette astuce simplifiera la conception du contrôleur présenté au chapitre 3.

Il est bon de rappeler qu'une des hypothèses de départ assume qu'il n'y a aucun glissement des pattes en contact avec le sol. Cette hypothèse peut être, en fait, une définition de ce qu'est une patte au sol. Si un glissement est détecté, la patte ne doit plus être considérée comme étant au sol. Ce qui amène à distinguer les pattes libres des pattes au sol. Cette distinction est ainsi représentée :

$$\chi_p = \begin{bmatrix} \chi_{pl} \\ \chi_{ps} \end{bmatrix} \tag{2.28}$$

Où sont notées les pattes libres (pl) et les pattes au sol (ps). La contrainte s'exprime donc ainsi, si la patte i est au sol :

$$\chi_{pi} = \begin{bmatrix} \mathbf{L}_i \\ \mathbf{D}_i \end{bmatrix} \tag{2.29}$$

où $\mathbf{L}_i$ et $\mathbf{D}_i$ représentent la distance entre la patte i et le repère inertiel.

Cette contrainte peut également s'exprimer sous la forme différentielle suivante :

$$\dot{\chi}_{pi} = 0 \tag{2.30}$$

En fait, les pattes au sol serviront à contrôler le corps et les pattes libres seront contrôlées pour suivre une trajectoire propre à elles. Un nombre minimal de deux pattes au sol (quatre actionneurs) sera nécessaire pour pouvoir contrôler le corps dans ces trois DDL. En effet, pour chaque DDL, un actionneur doit être assigné pour le contrôler. Donc, si le corps doit être contrôlé selon trois DDL, il faut trois actionneurs. De plus, il faut mentionner qu'une combinaison d'au moins une patte avant et d'une patte arrière diagonalement opposées est nécessaire pour supporter le corps, sans quoi le robot risque de perdre l'équilibre et le contrôle devient alors impossible.

2.7 Modèle dynamique

Ajouté au modèle cinématique qui fait le lien entre les positions, les vitesses et les forces au niveau statique, le modèle dynamique calcule les forces nécessaires pour engendrer le mouvement. L'utilisation d'un modèle dynamique permet la conception du contrôleur et la simulation du robot.

Pour concevoir un contrôleur qui donnera des performances adéquates et pour bien définir les stratégies de contrôle, le modèle dynamique s'avère un outil précieux. Le terme performances fait référence à la précision, le temps de réponse et la robustesse du contrôleur. En fait, les besoins pour Haedus sont de grandes vitesses de déplacements sur de courtes distances et de grandes accélérations de mouvements. Voilà la nécessité de baser le contrôleur sur un modèle dynamique précis. Pour accélérer et arrêter une lourde charge rapidement avec précision, il faut connaître le modèle, la dynamique du robot. Un modèle dynamique complet comprend les forces liées aux inerties, les forces de friction, de gravité, de Coriolis et centrifuges.

Certains auteurs négligent la dynamique des pattes (Takanishi 1985) justifiant cette simplification en disant que le corps a une dynamique beaucoup plus importante. Cependant, lorsqu'il s'agit de faire courir un robot et que les accélérations des pattes sont importantes, la dynamique de celles-ci ne peut être négligée sans quoi le comportement du robot serait largement perturbé (Hong 1999, Yoneda 1996).

La simulation dynamique du système robotique procure également plusieurs avantages. Elle est un bon outil d'aide à la conception mécanique pour le dimensionnement puisqu'il est possible d'obtenir les couples nécessaires à chaque articulation ainsi que les forces agissantes sur les membrures. Elle permet également d'explorer plusieurs stratégies de contrôleurs et de les valider avant de les implanter sur le robot réel.

Il existe deux méthodes principales pour obtenir un modèle dynamique en robotique : La méthode de Lagrange, présentée par Spong (1989), basée sur les énergies, et la méthode de Newton-Euler telle que vu par Craig (1989). Il

existe aussi des méthodes plus élaborées pour obtenir des modèle dynamiques en chaîne fermée. Xu (1998) a présenté une approche dérivée de Kane et de Newton-Euler pour concevoir un modèle dynamique de robots en chaînes fermées qui a pour avantage de réduire le nombre de calculs. Un tableau comparatif des différentes méthodes accompagne son étude. Le logiciel SYMOFROS utilisé pour la modélisation d'Haedus et présenté par Piedboeuf (1998) utilise la méthode de Jourdain, basée sur le principe de puissance virtuelle, une généralisation du principe d'Alembert.

Ces méthodes conduisent toutes deux au modèle dynamique exprimé sous la forme suivante :

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{N}(\dot{\mathbf{q}},\mathbf{q}) = \boldsymbol{\tau} \tag{2.31}$$

Où $\mathbf{q}$ est le vecteur de coordonnées généralisées, $\boldsymbol{\tau}$ est le vecteur des forces généralisées et $\mathbf{M}$ désigne la matrice de masses et $\mathbf{N}$, le vecteur non-linéaire des forces de gravité, de friction, de Coriolis, et centrifuges.

Pour Haedus, le modèle s'exprime en séparant en deux portions les coordonnées généralisées telles que présentées précédemment en 2.17. Ceci permet la séparation du corps et des pattes et conduit à la représentation suivante du modèle :

$$\begin{bmatrix} \mathbf{M}_c & \mathbf{M}_{cp} \\ \mathbf{M}_{pc} & \mathbf{M}_p \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}}_c \\ \ddot{\mathbf{q}}_p \end{bmatrix} + \begin{bmatrix} \mathbf{N}_c \\ \mathbf{N}_p \end{bmatrix} = \begin{bmatrix} \mathbf{B}_c \\ \mathbf{B}_p \end{bmatrix} \boldsymbol{\tau} = \mathbf{B}\boldsymbol{\tau} = \boldsymbol{\Gamma} \tag{2.32}$$

où les forces généralisées symbolisées par $\boldsymbol{\Gamma}$, ne correspondent pas directement aux couples des actionneurs. En effet, la matrice de couplage $\mathbf{B}$ assure la transformation entre les efforts des actionneurs $\boldsymbol{\tau}$ et les forces généralisées $\boldsymbol{\Gamma}$. À remarquer que la matrice de couplage pour le corps $\mathbf{B}_c$ est une matrice nulle, car

aucun actionneur n'a de lien direct sur la position du corps. La matrice $\mathbf{B_p}$ est une matrice identité, car les coordonnées généralisées sont décrites dans le même ordre que les forces généralisées dans le modèle. De plus, il y a une correspondance directe entre ces deux vecteurs.

Pour faire suite à cette séparation, citons la transformation de coordonnées expliquée à la section 2.5 qui permet de simplifier la conception du contrôleur et la réduction de modèle. En appliquant la transformation de coordonnées de l'équation 2.27 au modèle décrit par l'équation 2.32, l'équation suivante est obtenue :

$$\begin{bmatrix} \mathbf{M_c} & \mathbf{M_{cp}} \\ \mathbf{M_{pc}} & \mathbf{M_p} \end{bmatrix} \frac{\partial \mathbf{H}^{-1}}{\partial t} \begin{bmatrix} \dot{\chi}_c \\ \dot{\chi}_p \end{bmatrix} + \begin{bmatrix} \mathbf{M_c} & \mathbf{M_{cp}} \\ \mathbf{M_{pc}} & \mathbf{M_p} \end{bmatrix} \mathbf{H}^{-1} \begin{bmatrix} \ddot{\chi}_c \\ \ddot{\chi}_p \end{bmatrix} + \begin{bmatrix} \mathbf{N_c} \\ \mathbf{N_p} \end{bmatrix} = \begin{bmatrix} \mathbf{B_c} \\ \mathbf{B_p} \end{bmatrix} \boldsymbol{\tau} \qquad (2.33)$$

Qui peut être écrite sous forme plus compacte :

$$\mathbf{M}\mathbf{H}^{-1}\ddot{\chi} + \mathbf{M}\frac{\partial \mathbf{H}^{-1}}{\partial t}\dot{\chi} + \mathbf{N} = \mathbf{B}\boldsymbol{\tau} \qquad (2.34)$$

Ensuite, pour compléter la transformation et retrouver les propriétés de symétrie et de positivité de la matrice de masses, les termes de l'équation peuvent être pré-multipliés par la transposée inverse de **H**, soit $\mathbf{H}^{-T}$:

$$\mathbf{H}^{-T}\mathbf{M}\mathbf{H}^{-1}\ddot{\chi} + \mathbf{H}^{-T}\mathbf{M}\frac{\partial \mathbf{H}^{-1}}{\partial t}\dot{\chi} + \mathbf{H}^{-T}\mathbf{N} = \mathbf{H}^{-T}\mathbf{B}\boldsymbol{\tau} \qquad (2.35)$$

La nouvelle matrice de masses H^{-T} M H^{-1} étant symétrique, définie positive, est nécessairement inversible. Tel qu'il sera vu au prochain chapitre sur le contrôle, cette propriété est essentielle pour la conception des lois de commande.

Enfin, pour simplifier l'écriture, les matrices **M**, **N** et **B** sont renommées, considérant la nouvelle forme du modèle dynamique après cette transformation vers le nouveau système de coordonnées :

$$\overline{\mathbf{M}}\ddot{\chi} + \overline{\mathbf{N}} = \overline{\mathbf{B}}\tau \qquad (2.36)$$

où $\overline{\mathbf{M}}$ est la nouvelle matrice de masses exprimée dans le système de coordonnées cartésiennes, $\overline{\mathbf{N}}$ est le nouveau vecteur non-linéaire et $\overline{\mathbf{B}}$, la matrice de couplage transformée.

2.8 Réduction du modèle suivant les contraintes

La dernière partie du développement du modèle dynamique est la réduction de celui-ci selon les pattes considérées comme étant « au sol ». Cette opération a tout simplement pour but d'intégrer les contraintes dans le modèle en réduisant le nombre de DDL. De façon plus intuitive, les pattes portantes, dites « au sol », sont utilisées pour manipuler le corps. Cette approche est similaire à celle présentée par Murray (1994) pour le contrôle d'un objet manipulé par les doigts d'une main robotique. Les pattes portantes servent donc à contrôler les trois DDL du corps, à la manière dont les doigts manipulent un objet. C'est pourquoi leurs coordonnées généralisées (la position de leurs extrémités) doivent être retirées du modèle sous cette condition. Les pattes libres, quant à elles, peuvent être contrôlées dans l'espace cartésien comme le corps. Cette stratégie est particulièrement puissante dans le cas d'un robot marcheur puisqu'elle permet de définir un seul modèle dynamique pour toutes les configurations possibles. Pour Haedus par exemple, il y a 16 configurations possibles de pattes au sol ou libres. En fait, il existe 2^n configurations où n représente le nombre de pattes.

Avec la distinction proposée en 2.28 qui nous a permis de séparer les pattes libres et les pattes au sol, et la définition du modèle après transformation en 2.36, le modèle avant réduction apparaît ainsi :

$$\begin{bmatrix} \overline{\mathbf{M}}_c & \overline{\mathbf{M}}_{cpl} & \overline{\mathbf{M}}_{cps} \\ \overline{\mathbf{M}}_{plc} & \overline{\mathbf{M}}_{pl} & \overline{\mathbf{M}}_{plps} \\ \overline{\mathbf{M}}_{psc} & \overline{\mathbf{M}}_{pspl} & \overline{\mathbf{M}}_{ps} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{X}}_c \\ \ddot{\mathbf{X}}_{pl} \\ \ddot{\mathbf{X}}_{ps} \end{bmatrix} + \begin{bmatrix} \overline{\mathbf{N}}_c \\ \overline{\mathbf{N}}_{pl} \\ \overline{\mathbf{N}}_{ps} \end{bmatrix} = \begin{bmatrix} \overline{\mathbf{B}}_c \\ \overline{\mathbf{B}}_{pl} \\ \overline{\mathbf{B}}_{ps} \end{bmatrix} \tau \qquad (2.37)$$

Sous cette forme, le corps, les pattes au sol et les pattes libres sont distingués. Mais en réalité, à cette étape, seules les pattes au sol seront traitées différemment. Grâce à cette distinction, les coordonnées des pattes au sol peuvent être éliminées par une simple multiplication matricielle. La réduction du modèle est donc définie par :

$$\begin{bmatrix} \dot{\mathbf{X}}_c \\ \dot{\mathbf{X}}_{pl} \\ \dot{\mathbf{X}}_{ps} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{X}}_c \\ \dot{\mathbf{X}}_{pl} \end{bmatrix} \qquad (2.38)$$

où la matrice de réduction est composée d'une matrice identité dans le modèle avec les nouvelles coordonnées généralisées pour le corps et les pattes libres, et d'une matrice nulle pour les pattes au sol en raison de l'équation 2.30 qui exprime la vitesse des extrémités des pattes au sol égale à 0.

Finalement, le modèle réduit conduit à cette équation :

$$\begin{bmatrix} \overline{\mathbf{M}}_c & \overline{\mathbf{M}}_{cpl} \\ \overline{\mathbf{M}}_{plc} & \overline{\mathbf{M}}_{pl} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{X}}_c \\ \ddot{\mathbf{X}}_{pl} \end{bmatrix} + \begin{bmatrix} \overline{\mathbf{N}}_c \\ \overline{\mathbf{N}}_{pl} \end{bmatrix} = \begin{bmatrix} \overline{\mathbf{B}}_c \\ \overline{\mathbf{B}}_{pl} \end{bmatrix} \tau \qquad (2.39)$$

Qui peut être représentée sous forme compacte ainsi :

$$\overline{\overline{\mathbf{M}}}\ddot{\chi} + \overline{\overline{\mathbf{N}}} = \overline{\overline{\mathbf{B}}}\tau = \mathbf{F} \qquad (2.40)$$

Ce modèle sera utilisé pour concevoir le système de commande présenté au chapitre trois.

2.9 Logiciel de modélisation SYMOFROS

Symbolic Modeling of Flexible Robots and Simulation (SYMOFROS) est un logiciel de modélisation basé sur une description topologique d'un système robotique. SYMOFROS permet de modéliser et de simuler des systèmes robotiques complexes en chaînes ouvertes ou fermées, avec des membres et des articulations flexibles ou rigides. Il offre une banque d'outils de 55 fonctions incluant les fonctions de cinématique et de dynamique, ainsi que des fonctions de conversion et des outils de manipulation matricielle. Il permet aussi d'obtenir les détails du modèle en n'importe quel point du système et les matrices qui le composent. SYMOFROS a été développé par l'Agence spatiale canadienne dans le but d'apporter un support à la conception et à l'expérimentation de contrôleurs. Il offre aussi quelques fonctions de contrôle linéaire et non-linéaire. Voir la liste des fonctions au tableau 2.2.

Tableau 2.2

Bibliothèque de fonctions de SYMOFROS

Section	Description
Basic Functions	Initialisation de l'environnement Simulink et des blocs SYMOFROS, fonctions de base de SYMOFROS
Controller Toolbox	Contrôleurs PD, PI, PID, cartésiens, contrôleurs de dynamique directe et inverse

Differential Eq. Solvers	Fonctions de calcul de dynamique directe avec ou sans friction
Friction Toolbox	Friction au niveau des articulations et couples de friction, vecteur non-linéaire d'ajustement en fonction de la friction aux articulations
Extremity Operators	Matrices Jacobiennes, positions, vitesses et accélérations des extrémités, cinématique directe et inverse
Signal Mux/Demux	Extrait les coordonnées, vitesses généralisées, accélérations et couples généralisés
Contact Dynamics	Modèles de contacts
Blockset Extras	Utilités diverses
IPC blockset	Communication inter-processus comme TCP-IP et UDP-IP et protocole de partage de mémoire
Transformations	Transformations des orientations (quaternions, Euler, matrice de rotation)

Pour la modélisation d'Haedus, SYMOFROS a été choisi considérant sa rapidité à concevoir et à modifier le modèle d'un robot. Son choix repose également sur les outils de simulation dont il dispose et l'accessibilité de l'information au niveau du modèle. Grâce à l'approche présentée dans ce mémoire dont SYMOFROS fait parti, il est facile par exemple, d'ajouter un bras manipulateur au robot ou un axe de mouvement supplémentaire à une patte.

Pour la définition du modèle, SYMOFROS utilise l'interface Simulink de Matlab. Le modèle est ensuite généré par Maple qui fait tout le calcul symbolique et transcrit en language de programmation C les matrices et les fonctions liées au modèle. Enfin, Matlab compile le code C pour en faire une bibliothèque de fonctions à liens dynamiques (FICHIER.DLL). Le même code peut aussi être compilé sur la plate-forme d'opération temps-réel QNX où le modèle peut être interrogé en simulation ou en opération réelle avec un robot.

Le code a spécialement été concu pour être exécuté en temps réel et utilise les fonctions DSP Blockset de Matlab. L'exécution d'un programme de simulation et/ou de contrôle d'un robot physique se fait sur Simulink et peut être exécuté en temps réel sur une ou plusieurs plate-formes appelées nœuds de calcul, grâce à l'environnement de travail RT-Lab. Cet environnement sera décrit plus en détail au chapitre 4 et un bref exposé sur l'ordre d'exécution des logiciel pour générer un programme de contrôle temps réel est présenté à la section 4.3.1.

Ref : http://www.opal-rt.com/symofros.htm

2.9.1 Définition du modèle d'Haedus sur SYMOFROS

Concevoir un modèle consiste à déterminer les axes de mouvements et les liens qui les relient. Le modèle complet tel qu'utilisé pour définir Haedus comporte les liens cinématiques exprimant le mouvement entre chacun des repères et les liens dynamiques reliant les forces aux mouvements. Chaque paramètre doit être entré dans les blocs SYMOFROS pour bien définir les liens. Un bloc représente un segment du robot et peut comporter jusqu'à trois DDL en translation et un DDL en rotation. Les blocs servent également à décrire les actionneurs, ainsi que tout type de force relié au segment représenté (friction articulaire, force externe, force élastique, etc.) Les paramètres peuvent être entrés sous forme de constantes ou sous forme de variables de façon à être modifiés lors de l'initialisation du modèle.

Une fois les axes et les repères identifiés, comme sur la figure suivante, il suffit d'utiliser l'interface de SYMOFROS présentée ci-contre pour entrer les données.

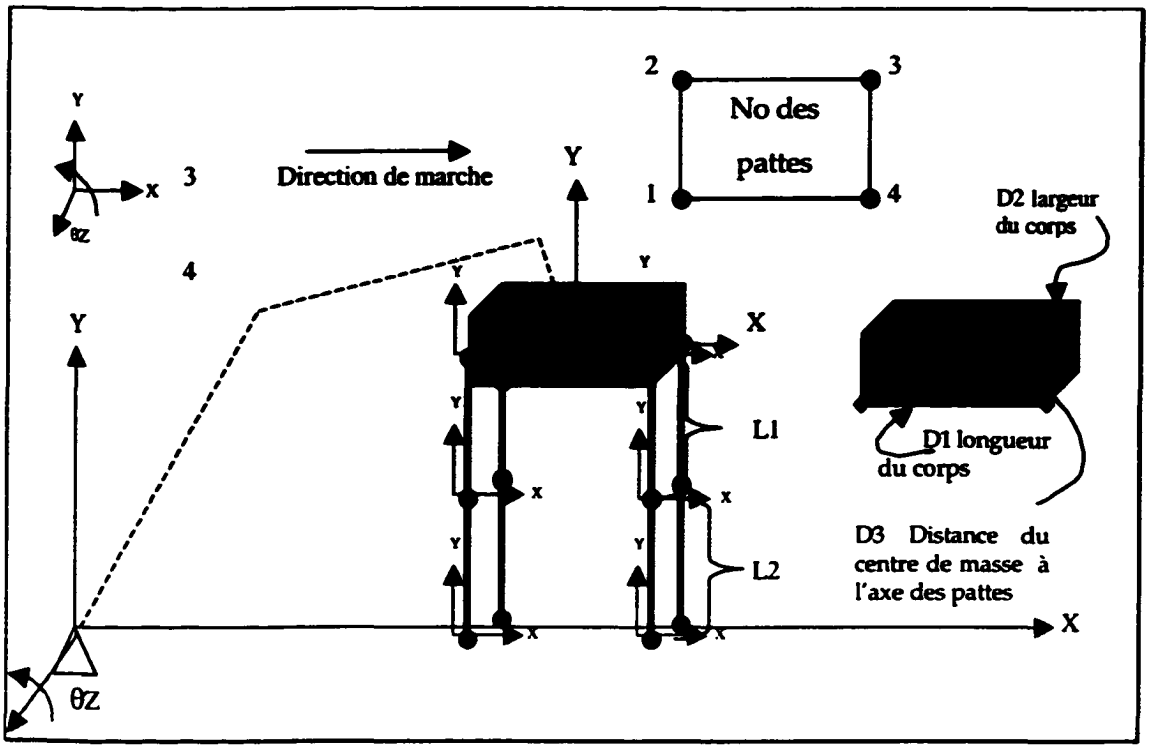


Figure 2.4 Définition des systèmes d'axe pour le robot Haedus

Cette définition ne suit pas la convention de Denavit-Hartenberg présentée en section 2.3. Avec SYMOFROS, il n'y a pas de convention spécifique à respecter, il faut simplement être intègre lors de la saisie des données. Sur la figure 2.4, la base est symbolisée par un triangle. C'est à partir de ce référentiel que le système robotique sera défini. Comme aucun membre du robot n'est rattaché à ce référentiel, il a été nécessaire d'en définir un.

2.9.1.1 Création d'un lien virtuel

Le corps de Haedus possède trois DDL qui lui sont conférés par un lien virtuel qui relie la « base » à ce dernier. Cet artifice se dessine comme un bras manipulateur sans masse et sans force perturbatrice, d'une longueur variable. La base est un référentiel fixe auquel s'accroche tous les segments du robot. Outre cette différence, Haedus se modélise comme un manipulateur robotique habituel.

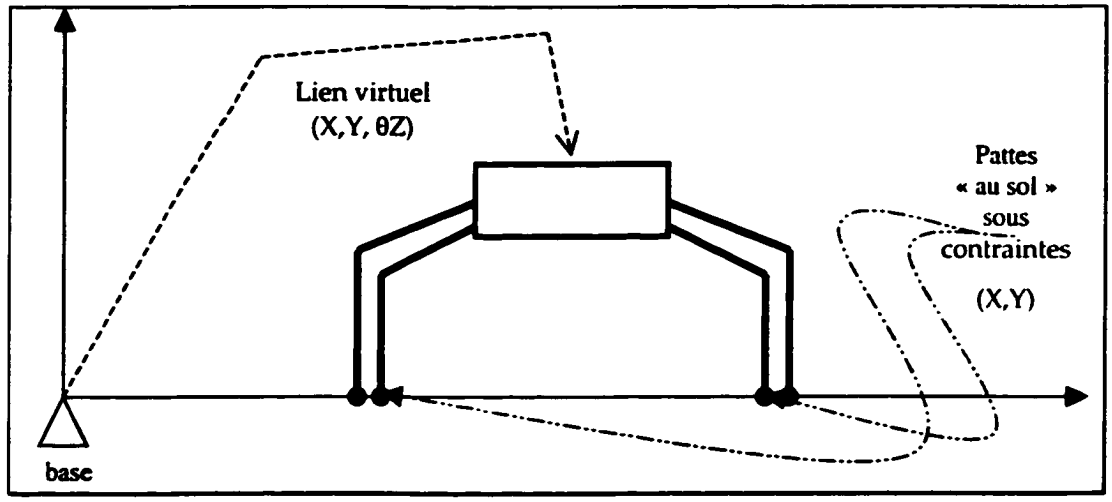


Figure 2.5 Concept du bras virtuel

Ce lien est représenté par le bloc « lien virtuel » qui se rattache à la base sur la figure 2.7. Ce bloc possède uniquement la définition des trois DDL du corps. Pour entrer les données, l'interface de la figure 2.6 fut utilisée.

2.9.1.2 Interface de définition du modèle

Le modèle est défini simplement en déterminant les segments qui composent le robot et les liaisons qui les unissent. En partant de la base, référentiel global, le lien virtuel définit les 3 premiers DDL auxquels le corps se rattache. À partir du centre de masse du corps, 4 référentiels sont définis où vont s'accrocher les pattes. Chaque patte est composée du segment appelé « cuisse » qui inclut le moteur « hanche » et d'un deuxième segment appelé « mollet » qui inclut le moteur « genou ». La patte se termine par un dernier bloc appelé « sabot ». Voir la figure 2.7.

2.9.1.3 Du dessin à l'interface

Ainsi, à partir de la figure 2.4, le modèle de la figure 2.7 a été défini à l'aide de l'interface de la figure 2.6. Le bloc « base » est le repère inertiel, soit la référence absolue du modèle. Sur le dessin de la figure 2.4, il est symbolisé par un triangle. Chaque bloc qui succède le bloc « base » dans le modèle de la figure 2.7 est un bloc « rigide », ce qui signifie, un lien non-flexible comportant une articulation, un segment ou les deux selon le cas. Voir la figure 2.8.

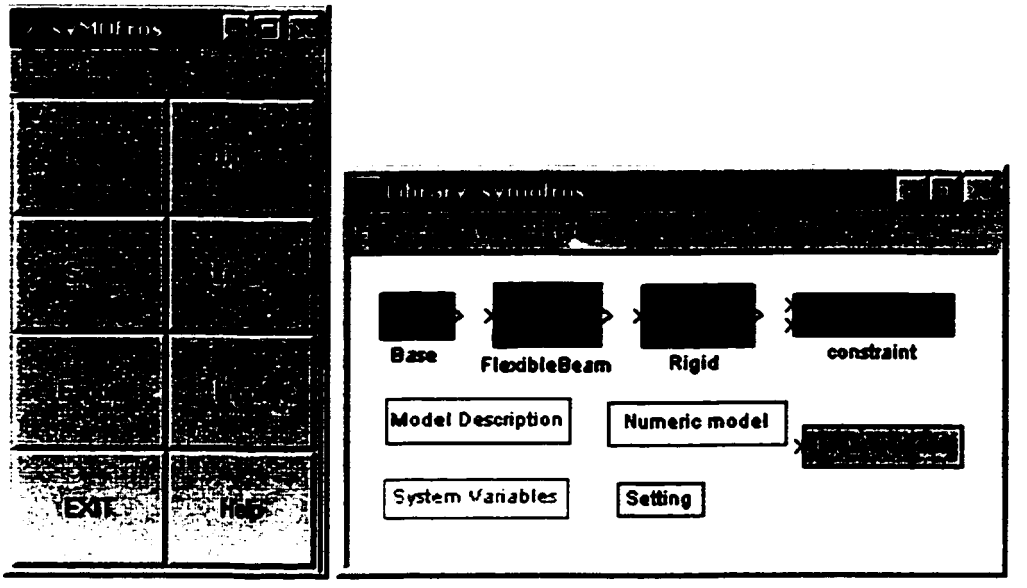


Figure 2.6 Interface de création de modèle SYMOFROS

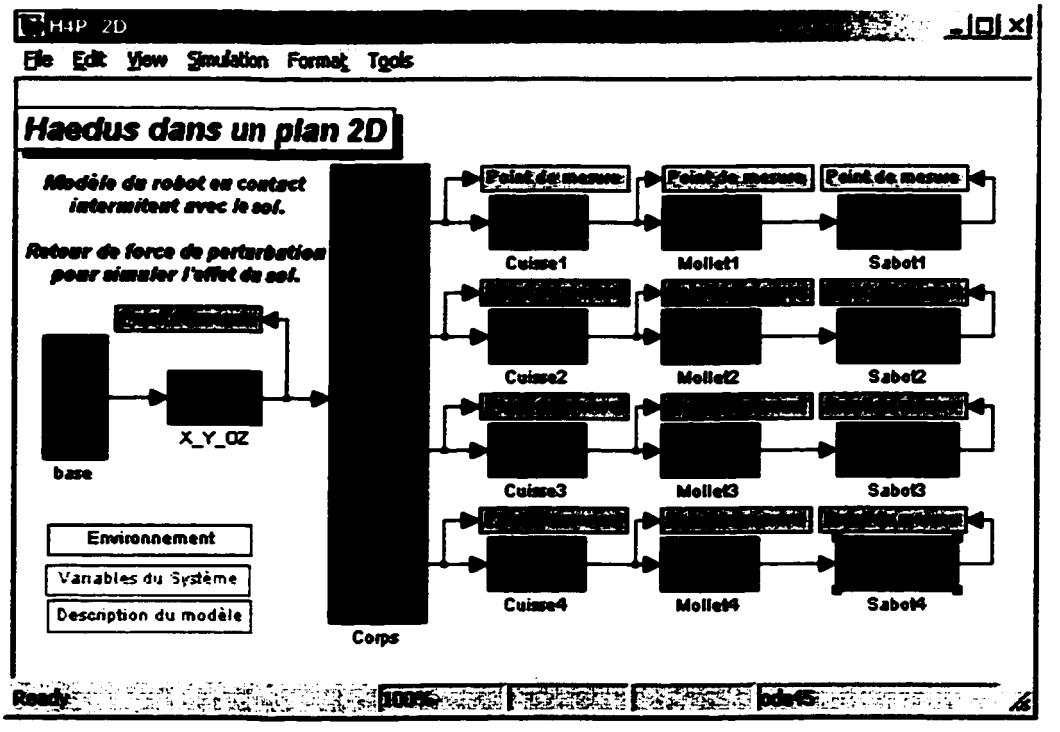


Figure 2.7 Modélisation de Haedus sur SYMOFROS

Les blocs « points de mesure » permettent d'accéder aux informations de chacun des blocs comme la position cartésienne, les vitesses, etc. Sans préciser les points de mesure, l'information n'est accessible qu'aux extrémités au moment de la simulation.

Chaque segment rigide est composé des éléments apparaissant sur la figure suivante :

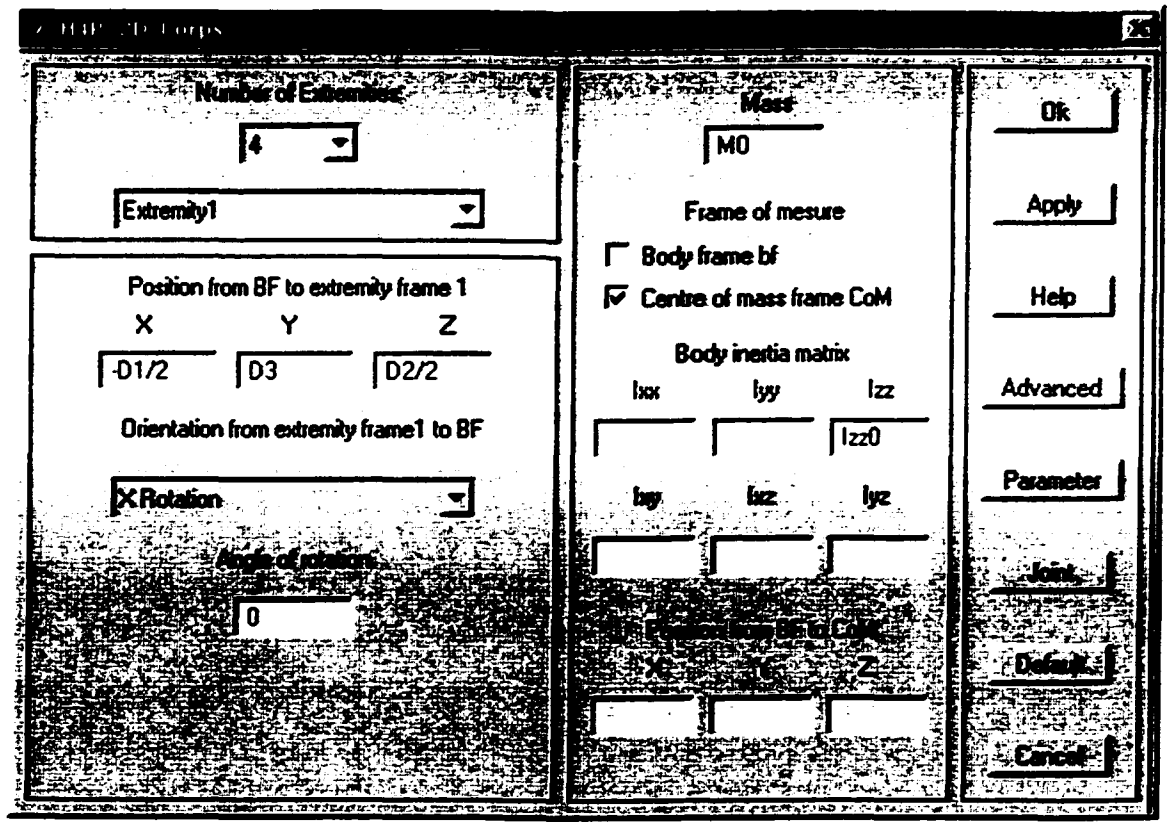


Figure 2.8 Définition d'un bloc « Rigid »

Le corps comporte 4 extrémités et la première est déterminée par les paramètres X,Y, et Z qui sont les distances entre le « body frame » (BF sur la

figure 2.8) ou le référentiel du corps et le prochain référentiel. Chaque paramètre d'un bloc est toujours en relation avec le bloc précédent si cela est applicable.

Ensuite, un angle peut être défini selon n'importe quel axe de rotation pour faire aligner le prochain référentiel au précédent. Il faut préciser que le robot est considéré dans une configuration initiale, c'est à dire que tous ses axes sont à 0. Dans le cas d'Haedus, tous les référentiels possèdent la même orientation, alors un angle de 0 est toujours utilisé dans le modèle. (Aucun changement d'orientation entre deux repères d'un même corps)

Enfin, viennent les paramètres de masse, d'inertie et de centre de masse. Le centre de masse peut être appliqué au « body frame », soit le point d'attache avec le repère précédent, ou peut être appliqué à un point spécifié par une translation exprimée dans les cases [X, Y et Z] sous « Position from BF to CoM ». La masse M0 est une variable, de même pour Izz0, l'inertie autour de Z. Les deux sont appliquées au centre de masse qui coïncide avec le référentiel, puisqu'il n'y a aucune translation de spécifiée entre le référentiel et le centre de masse (CoM).

Les variables de chacun des blocs doivent être définies en sélectionnant le bouton « Parameters ». Elles peuvent être définies comme des constantes ou simplement laissées comme variables à initialiser lors de la mise en marche du programme de simulation.

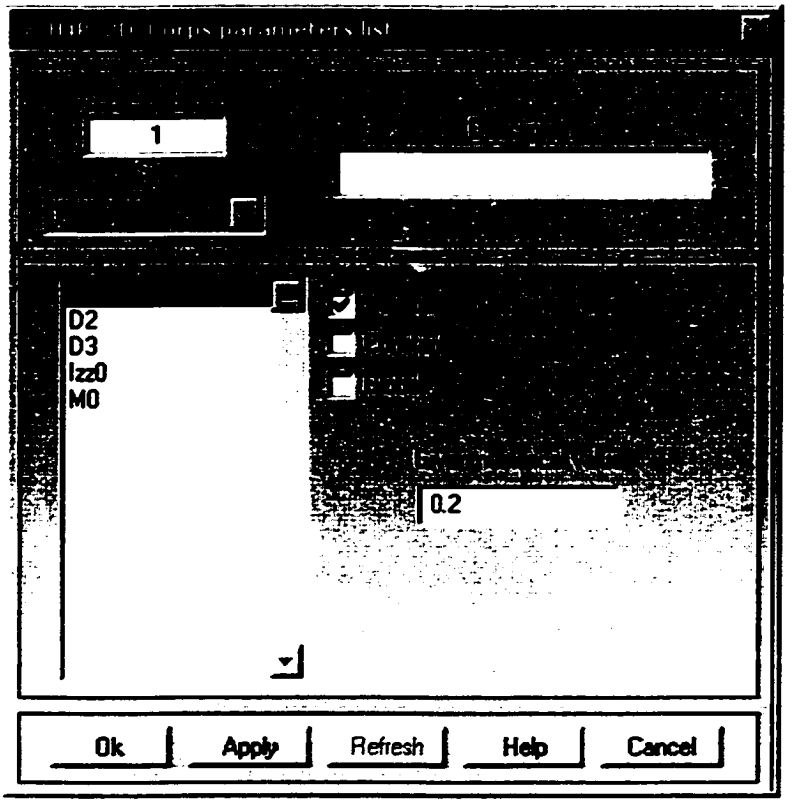


Figure 2.9 Définition des paramètres

Pour la définition des actionneurs, il faut utiliser le bouton « Joint » de la fenêtre principale du bloc. Voir la figure 2.8. Voici un exemple avec la hanche1 :

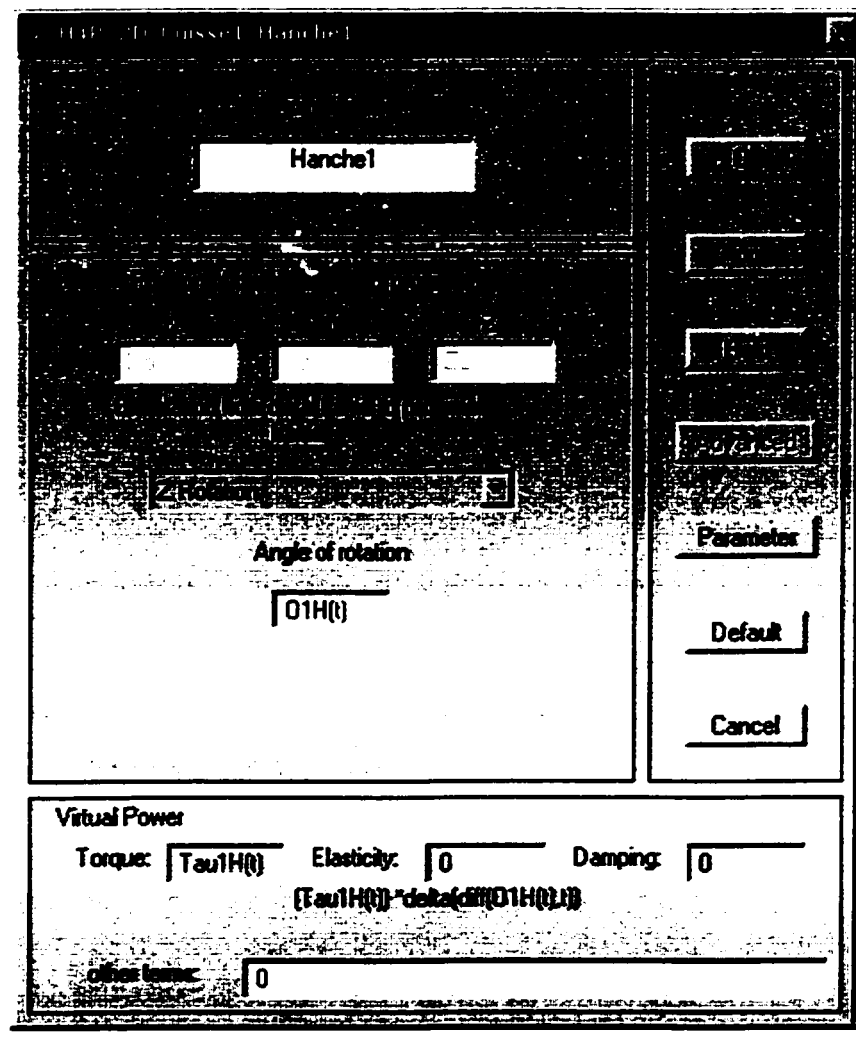


Figure 2.10 Définition d'une articulation

Les DDL sont définis dans les cases X, Y, Z et « Angle of rotation ». C'est à cette étape que seront déclarées les coordonnées généralisées. Pour ce faire, le nom de la variable doit être suivi de (t), pour dire qu'elle varie dans le temps.

Le même concept s'applique aux forces généralisées. Il ne suffit pas de définir le mouvement, il faut aussi définir l'actionneur qui le contrôle. Tau1H(t)

est l'actionneur qui fait tourner la hanche1. L'articulation peut être élastique et peut aussi comporter de l'amortissement.

2.9.2.3 Forces externes

Les forces externes sont modélisées comme des perturbations appliquées aux extrémités des pattes. Cette particularité est utilisée en simulation seulement où le sol est virtuel. Pour simuler les forces externes que le sol exerce sur les pattes du robot, ces dernières sont calculées et soumises comme perturbation de sorte à assurer que la contrainte 2.30 relative au mouvement des pattes au sol soit respectée. Un détail de ces calculs est présenté à la section sur la simulation en 3.4.

Le bouton « Advanced » donne droit aux autres options suivantes, comme pour le sabot :

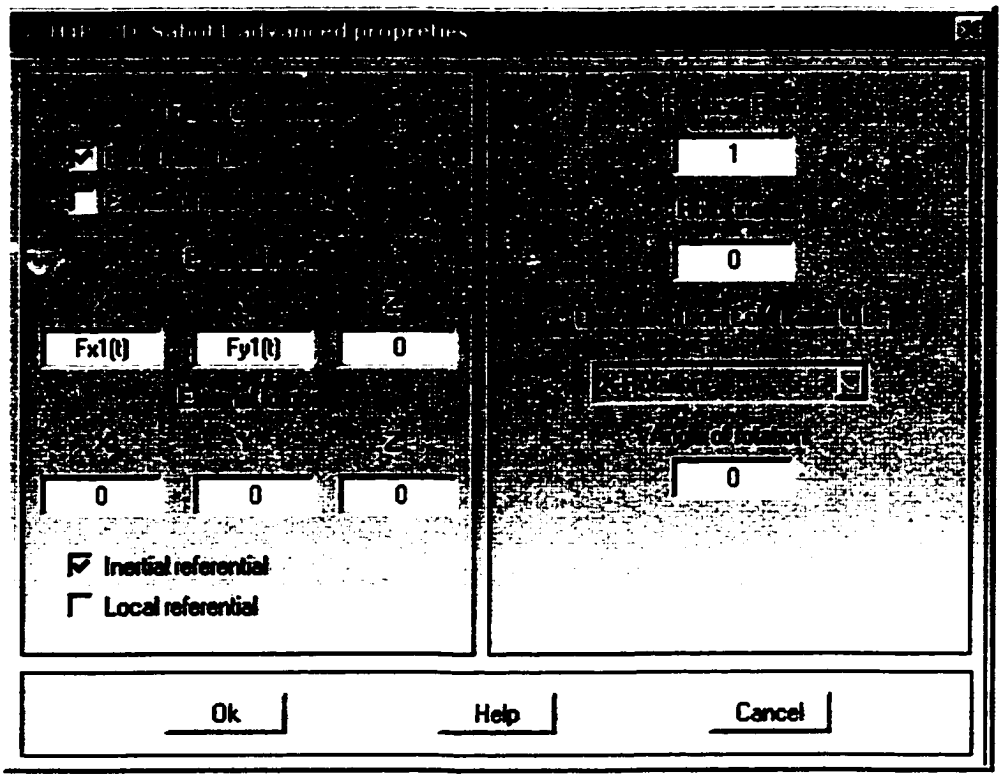


Figure 2.11 Propriétés avancées : forces externes

Des variables de forces en x et en y ont été ajoutées pour simuler le contact avec le sol.

Enfin, il faut préciser la nature de chacune des variables avec l'icône des variables du système, « System Variables ». Cette icône est présentée dans la banque d'outils et apparaît avec chaque nouveau modèle. Voir figure 2.6 et 2.7.

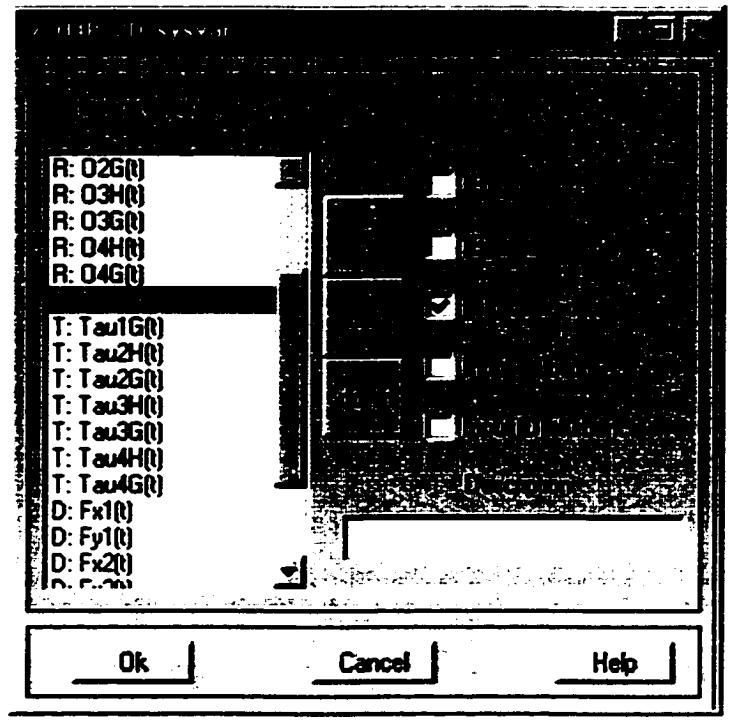


Figure 2.12 Variables du système

Les coordonnées généralisées sont placées dans l'ordre apparaissant dans cette liste et sont symbolisées par un R pour « rigid ». Les couples ou forces généralisés sont symbolisés par un T pour « torque ». Et enfin, les perturbations ou forces externes sont symbolisées par un D pour « disturbance ». Voir figure 2.12.

Une fois le modèle complété et les variables spécifiées, le modèle peut être sauvé et exporté. La commande à entrer pour générer le code avec Maple est : symogenmodel('H4P_2D'). Ensuite, une fois que Maple a terminé de générer le code C, il faut entrer : symomakemodel('H4P_2D.c') pour créer la bibliothèque H4P_2D.DLL qui sera utilisée en simulation. L'aide sur ces

commandes est disponible en entrant, toujours dans l'environnement de commande de Matlab ; symohelp

Pour créer des simulations avec Simulink, il suffit d'utiliser les fonctions de la banque d'outils SYMOFROS pour Simulink. Toutes les fonctions de SYMOFROS sont aussi accessibles par Matlab.

2.10 Conclusion

Ce chapitre présentait un survol de la modélisation des systèmes robotiques. Il décrivait en particulier le modèle d'Haedus et le logiciel de modélisation SYMOFROS, lequel a été choisi pour la réalisation du modèle d'Haedus afin de générer automatiquement le modèle. Les éléments essentiels de l'approche de définition du modèle développée sont la transformation de coordonnées et la réduction du modèle suivant les contraintes présentes lorsque les pattes sont au sol. Ces opérations simplifient de façon magistrale la conception du contrôleur et la prise en charge des contraintes dans le modèle.

CHAPITRE 3

CONCEPTION DU CONTROLEUR ET SIMULATION

3.1 Introduction

La commande assurant la démarche d'un robot marcheur doit tenir compte de plusieurs facteurs. D'abord, les pattes portantes, celles qui entraînent le mouvement du corps, doivent travailler dans le même sens pour éviter de générer des efforts internes qui pourraient endommager la mécanique et dépenser l'énergie inutilement. Ensuite, la synchronisation des mouvements est importante pour assurer une démarche fluide et garantir la stabilité du robot. Les objectifs de performance du contrôleur de démarche sont la robustesse, la flexibilité, la stabilité et la réponse dynamique. Ce chapitre présente comment la conception du contrôleur d'Haedus permettra d'atteindre ces objectifs. Un schéma de conception illustre la structure du contrôleur et les équations qui l'accompagnent expliquent comment le réaliser. Le contrôleur a été validé par les simulations présentées dans ce chapitre.

3.2 Schéma de contrôle

Dans le but d'atteindre les performances désirées, le contrôleur d'Haedus a été décomposé en deux étages. Le premier étage sert à la planification des trajectoires du corps et des pattes libres dans le plan cartésien. C'est à ce niveau que les mouvements libres seront synchronisés. Le deuxième étage est utilisé

pour effectuer le suivi de trajectoire et la distribution des efforts au niveau des pattes au sol. Cette répartition est obtenue à l'aide d'une transformation des efforts dans le plan cartésien en couples appliqués aux articulations. Pour effectuer cette transformation, l'utilisation d'une pseudo-inverse telle que présentée à la section 3.3.6.1 assure de minimiser les efforts internes. Cette méthode est une des plus communément utilisée (Gao 1990), (Lehtinen 1994), (Alexander 1997) et (Zhou 1999). De plus, l'énergie dépensée pour effectuer le mouvement est ainsi réduite au minimum puisque les efforts sont répartis sur toutes les pattes portantes.

La partie planification de trajectoire comme celle effectuée par Bai (1999) pour le corps dans son environnement n'est pas détaillée dans cet ouvrage, puisqu'elle relève d'un autre système doté de mécanismes de décisions. Le schéma suivant illustre les différentes parties du contrôleur.

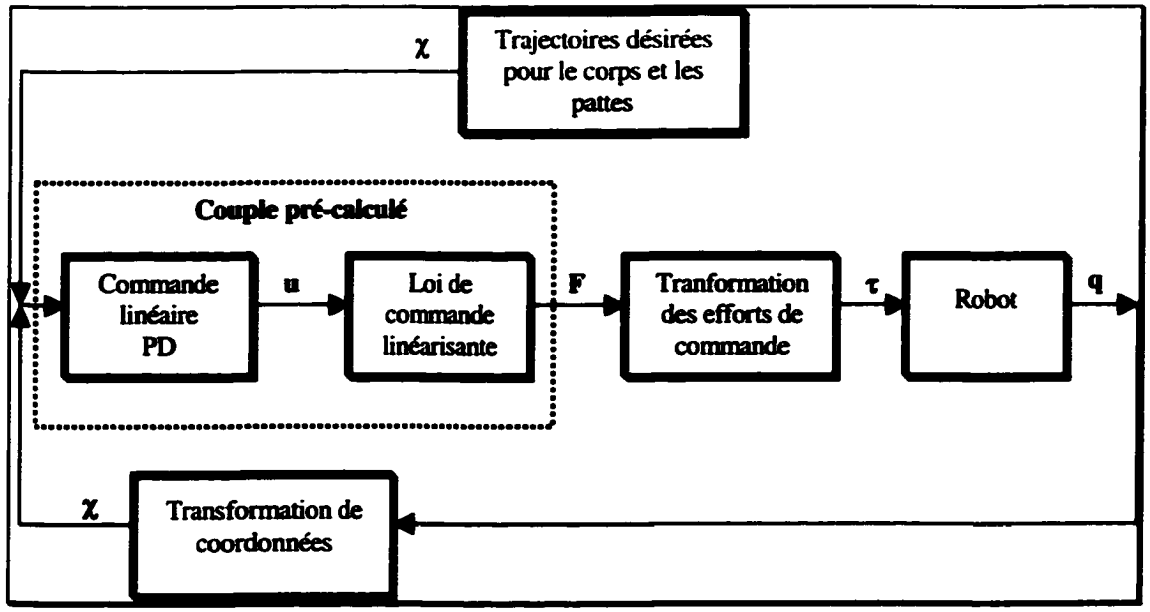


Figure 3.1 Schéma de conception du contrôleur de démarche de Haedus

3.3 Description du contrôleur

Le contrôleur de démarche d'Haedus est basé sur le modèle dynamique présenté au chapitre précédent. Il utilise un contrôleur de couple pré-calculé pour le suivi de trajectoire et une transformation des efforts par un calcul matriciel, appelée pseudo-inverse, pour générer les couples correspondants aux forces généralisées, à partir des efforts déterminés dans le plan cartésien.

Les forces généralisées, désignées par τ, représentent les huit couples appliqués aux articulations du robot :

$$\tau = [tau1H(t), tau1G(t), ..., tau4H(t), tau4G(t)] \qquad (3.1)$$

Ces huit couples articulaires, destinés aux quatre pattes, sont, identifiés par paires avec l'identification : *H* pour la hanche et *G* pour le genou. Ils doivent être combinés de façon à positionner le corps (3 DDL) et les pattes (8 DDL), soit 11 DDL en tout pour 8 actionneurs. Tel qu'expliqué à la fin de la section 2.6, il faut minimalement un effort fournit par un actionneur pour contrôler chaque degré de liberté. Cependant, le nombre de DDL peut être réduit par les contraintes. Par exemple, en posant deux pattes au sol, quatre contraintes sont ajoutées, ce qui réduit à 7 le nombre de DDL du système. Ces derniers peuvent donc être contrôlés par les 8 actionneurs présents.

3.3.1 Trajectoire désirée pour le corps

Les instructions relatives au mouvement sont prévues pour être données soit par l'usager, soit par un contrôleur de plus haut niveau. Comme Haedus est contraint dans un plan, cette commande se limite à avancer, reculer, incliner le

corps ou en modifier la hauteur. Les trois DDL du corps se résument donc à (x, y, θ_z). La définition de la trajectoire du corps est donnée à la section 3.3.3.

Pour la simulation et l'expérimentation, la démarche choisie est une démarche lente, statiquement stable, appelée « Amble » ou « Crawl » tel qu'indiqué sur les figures 1.3 et 1.4. Cette démarche est statiquement stable tant et aussi longtemps que la projection du centre de masse du corps reste à l'intérieur d'un triangle formé par les trois pattes qui le portent. Une seule patte à la fois est libre et se déplace librement. Pour synchroniser le mouvement des quatre pattes, elle doit donc se déplacer à une vitesse au moins trois fois supérieure à celle des pattes au sol. Selon la définition de la numérotation des pattes présentée à la figure 2.3 ou 2.4, la séquence de démarche sera ainsi (1,3,2,4) où le chiffre correspond à la patte libre. Ce type de démarche possède un paramètre appelé « coefficient de recouvrement » qui représente la période de temps où les quatre pattes sont au sol par rapport au temps où une patte est dans les airs. Ce chevauchement assure un meilleur équilibre lors du changement de patte.

3.3.2 Trajectoire désirée pour les pattes

À partir de la trajectoire du corps, une trajectoire est générée pour chaque patte libre. Comme le corps, les pattes évoluent dans un plan x,y. La trajectoire des pattes est donc définie en (x,y). Les trajectoires des pattes, comme celle du corps, sont exprimées par un polynôme de degré cinq. Cela assure une continuité au niveau de la position, de la vitesse et de l'accélération en tout point sur la trajectoire. Cette continuité est importante. Sans elle, la démarche

pourrait être sérieusement perturbée et le robot exécuterait des mouvements potentiellement dangereux pour son entourage et sa mécanique.

3.3.3 Représentation des trajectoires par un polynôme

Le polynôme de cinquième ordre, tel qu'exprimé par Craig (1989), permet de définir une trajectoire continue en position, vitesse et accélération entre deux points. L'équation décrivant ce polynôme est la suivante :

$$\chi(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \tag{3.2}$$

Les 6 coefficients du polynôme tel que présentés par Craig(1989) peuvent être calculés par les expressions suivantes :

$$\begin{aligned}
a_0 &= \chi_i, \\
a_1 &= \dot{\chi}_i, \\
a_2 &= \frac{\ddot{\chi}_0}{2}, \\
a_3 &= \frac{20\chi_f - 20\chi_i - (8\dot{\chi}_f + 12\dot{\chi}_o)t_f - (3\ddot{\chi}_o - \ddot{\chi}_f)t_f^2}{2t_f^3}, \\
a_4 &= \frac{30\chi_o - 30\chi_f + (14\dot{\chi}_f + 16\dot{\chi}_o)t_f - (3\ddot{\chi}_o - 2\ddot{\chi}_f)t_f^2}{2t_f^4}, \\
a_5 &= \frac{12\chi_f - 12\chi_o - (6\dot{\chi}_f + 6\dot{\chi}_o)t_f - (\ddot{\chi}_o - \ddot{\chi}_f)t_f^2}{2t_f^5}.
\end{aligned} \tag{3.3}$$

Ces équations découlent de la résolution du polynôme pour les 6 données exprimant la trajectoire à suivre, soient position χ, vitesse $\dot{\chi}$ et accélération $\ddot{\chi}$, initiales et finales (entre deux points de trajectoire). Une série de points ainsi définie exprime une trajectoire complète à suivre qui est utilisée pour les pattes libres et pour le corps.

3.3.4 Loi de commande linéarisante

Le modèle dynamique est la meilleure information disponible pour calculer les couples nécessaires pour produire un mouvement. En utilisant l'équation générale du modèle, il est possible de linéariser le système à l'aide d'une loi de commande, ce qui simplifie énormément la conception du système de contrôle. En considérant le modèle dynamique d'Haedus donné par l'équation 2.40, la loi de commande linéarisante est donnée par la relation suivante :

$$\mathbf{F} = \overline{\overline{\mathbf{M}}}\mathbf{u} + \overline{\overline{\mathbf{N}}} \tag{3.4}$$

où **F** est le vecteur des forces généralisées correspondant au vecteur des coordonnées généralisées χ. Comme l'indique l'équation 2.38, le vecteur χ se divise en deux parties : la première partie est formée des coordonnées généralisées décrivant la position et l'orientation du corps et la seconde partie est formée des coordonnées généralisées décrivant la position de l'extrémité des pattes libres. Le vecteur des forces généralisées **F** ne correspond pas directement aux efforts articulaires. En effet, les forces sont liées aux efforts articulaires par l'entremise de la relation suivante :

$$\mathbf{F} = \overline{\overline{\mathbf{B}}}\boldsymbol{\tau} \tag{3.5}$$

Comme il sera présenté à la section 3.3.5, cette relation peut être inversée de façon à obtenir les efforts articulaires en fonction des forces généralisées **F** obtenues par la loi de commande 3.5.

En appliquant la loi de commande 3.4 au système décrit par la relation 2.40, le système linéarisé suivant est obtenu :

$$\ddot{\chi} = \mathbf{u} \tag{3.6}$$

où **u** est la nouvelle variable de commande.

Le mouvement est alors entièrement caractérisé par **u**. Par exemple, si la contribution de **u** devait être fixée à 0, le robot devrait maintenir une position neutre sans bouger, ni même s'affaisser. Ceci est vrai à la condition qu'initialement les vitesses soient nulles, car c'est la contribution du terme $\overline{\mathbf{N}}$ qui fournit les efforts nécessaires pour lutter contre la gravité.

3.3.5 Commande linéaire

Le système étant linéarisé par la loi de commande 3.4, la commande de type Proportionnelle – Dérivée (PD) est utilisée pour le suivi de trajectoire. La boucle de rétroaction compare la trajectoire désirée avec la position réelle du corps et des pattes libres du robot. La commande est donc appliquée directement dans le plan cartésien, grâce à la transformation vue à la section 2.5.

La loi de commande PD est exprimée par l'équation suivante :

$$\mathbf{u} = \ddot{\chi}_d + \mathbf{K}_p(\chi_d - \chi) + + \mathbf{K}_d(\dot{\chi}_d - \dot{\chi}) \tag{3.7}$$

où $\ddot{\chi}_d$, $\dot{\chi}_d$ et χ_d sont les positions, vitesses et accélérations de la trajectoire désirée, $\mathbf{K}_p$ et $\mathbf{K}_d$ sont les matrices de gains proportionnel et dérivé et **u** est la commande linéaire à appliquer au modèle linéarisé. Le paramètre d'accélération $\ddot{\chi}_d$ est utilisé comme terme d'anticipation.

En appliquant la loi de commande 3.7 sur la dynamique linéarisée 3.6, la réponse dynamique de l'erreur de suivi devient :

$$\ddot{\tilde{\chi}} + \mathbf{K}_d \dot{\tilde{\chi}} + \mathbf{K}_p \tilde{\chi} = 0 \qquad (3.8)$$

où l'erreur de position et l'erreur de vitesse sont respectivement représentées par :

$$\tilde{\chi} = (\chi_d - \chi) \text{ et } \dot{\tilde{\chi}} = (\dot{\chi}_d - \dot{\chi}) \qquad (3.9)$$

En somme, la commande linéaire présentée par la relation 3.7 et la loi de commande linéarisante présentée par la relation 3.6 forment ce que l'on appelle une commande de couple précalculé telle qu'utilisée par Chevallereau (1997) dans son approche de commande utilisée sur un robot quadrupède.

3.3.5.1 Calcul des gains $\mathbf{K}_p$ et $\mathbf{K}_d$

En considérant les matrices de gains $\mathbf{K}_p$ et $\mathbf{K}_d$ diagonales, un ensemble de gains ($\mathbf{K}_p$ et $\mathbf{K}_d$) correspond à chaque terme du vecteur des erreurs de suivi. Ces gains sont calculés pour atteindre certaines performances qui peuvent être différentes pour chaque coordonnée généralisée. Par exemple, il serait logique d'imposer un temps de réponse plus court aux erreurs de suivi des extrémités des pattes qu'à celles des coordonnées généralisées du corps, qui est plus lourd et qui possède une dynamique plus lente. Selon le même raisonnement appliqué aux pattes, l'axe horizontal pourrait aussi être plus nerveux que l'axe vertical qui doit porter le poids du robot.

Les relations suivantes permettent d'établir les gains suivant le temps de réponse, avec un dépassement nul :

$$\mathbf{k}_p = -\lambda^2 \qquad \mathbf{k}_d = -2\lambda \qquad (3.10)$$

où λ, une valeur propre de multiplicité deux est donnée par la relation suivante :

$$\lambda = -\frac{4.73}{Tr} \tag{3.11}$$

où *Tr* est le temps de réponse désiré à 5%. En sommes, il suffit de poser le temps de réponse *Tr* désiré, calculer λ et enfin les gains Kp et Kd. La stabilité est assurée par le placement des deux pôles superposés et négatifs.

3.3.6 Transformation des efforts de commande

L'équation 3.5 précise que les couples ne sont pas directement fournis par le contrôleur de couple précalculé. Ce dernier calcule en effet les efforts pour les mouvements permettant de suivre la trajectoire, selon les coordonnées généralisées du corps et des pattes libres. Il faut donc transformer ces efforts en couple au niveau des articulations et les faire correspondre au vecteur de forces généralisées de l'équation 3.1. Un calcul matriciel appelé pseudo-inverse va permettre de faire cette transformation tout en minimisant la somme des couples au carré, donc indirectement en minimisant l'énergie. Cette méthode commune permet de calculer une solution pour un vecteur d'équations de type **y** = **A x**, où le nombre d'éléments en **x** est supérieur au nombre d'éléments dans **y** et où **A** est considéré comme étant de plein rang. La solution particulière utilisée ici correspond aux moindres carrés, puisque le calcul minimise la norme $||\mathbf{y}-\mathbf{A}\mathbf{x}||^2$. Ce calcul se traduit comme suit :

$$\boldsymbol{\tau} = \left[\overline{\overline{\mathbf{B}}}\right]^{+} \mathbf{F} \tag{3.13}$$

où la pseudo-inverse est symbolisée par $[\ ]^+$ tel que présenté par Khalil (1999) :

$$\left[\overline{\overline{\mathbf{B}}}\right]^{+} = \overline{\overline{\mathbf{B}}}^{T}(\overline{\overline{\mathbf{B}}\overline{\mathbf{B}}}^{T})^{-1} \tag{3.14}$$

3.3.6.1 Inverse généralisée

Ce calcul n'utilise pas tout l'espace des solutions, ce qui veut dire qu'une partie des actionneurs pourrait être utilisée pour accomplir d'autres objectifs, comme par exemple, limiter les possibilités de glisser au sol. En limitant les efforts contre la friction au niveau du sol, les chances de déraper seraient donc réduites.

La méthode de l'inverse généralisé permet de décrire tout l'espace des solutions de la transformation des forces généralisées en couples aux actionneurs. L'inverse de la relation de transformation des efforts de la relation 3.5 est alors donnée par l'équation suivante :

$$\boldsymbol{\tau} = \left[\overline{\overline{\mathbf{B}}}\right]^{+}\mathbf{F} + \left(\mathbf{I} - \left[\overline{\overline{\mathbf{B}}}\right]^{+}\overline{\overline{\mathbf{B}}}\right)z \tag{3.15}$$

où z est un vecteur quelconque qui caractérise entièrement l'espace nul de la matrice $\overline{\overline{\mathbf{B}}}$.

Simplement pour vérifier que l'équation 3.15 solutionne bien la relation 3.5, l'équation 3.15 a été remplacée dans 3.5, ce qui donne :

$$\mathbf{F} = \overline{\overline{\mathbf{B}}}\boldsymbol{\tau} = \overline{\overline{\mathbf{B}}}\left[\overline{\overline{\mathbf{B}}}\right]^{+}\mathbf{F} + (\overline{\overline{\mathbf{B}}} - \overline{\overline{\mathbf{B}}}\left[\overline{\overline{\mathbf{B}}}\right]^{+}\overline{\overline{\mathbf{B}}})z \tag{3.16}$$

où selon la relation 3.14 :

$$\overline{\overline{\mathbf{B}}}\left[\overline{\overline{\mathbf{B}}}\right]^{+} = \overline{\overline{\mathbf{B}}\overline{\mathbf{B}}}^{T}\left(\overline{\overline{\mathbf{B}}\overline{\mathbf{B}}}^{T}\right)^{-1} = \mathbf{I} \tag{3.17}$$

et

$$(\overline{\overline{\mathbf{B}}} - \overline{\overline{\mathbf{B}}}\left[\overline{\overline{\mathbf{B}}}\right]^{+}\overline{\overline{\mathbf{B}}}) = \overline{\overline{\mathbf{B}}} - \overline{\overline{\mathbf{B}}}\overline{\overline{\mathbf{B}}}^{T}\left(\overline{\overline{\mathbf{B}}}\overline{\overline{\mathbf{B}}}^{T}\right)^{-1}\overline{\overline{\mathbf{B}}} = 0 \qquad (3.18)$$

La relation 3.15 est la solution générale de l'équation 3.5 pour n'importe quelle valeur de z. Le vecteur z caractérise les efforts internes du système; il n'a donc aucun effet sur les forces généralisées puisque $(\mathbf{I} - \left[\overline{\overline{\mathbf{B}}}\right]^{+}\overline{\overline{\mathbf{B}}})z$ représente une projection du vecteur z dans l'espace nul de la matrice $\overline{\overline{\mathbf{B}}}$, ce qui n'affecte pas la solution de l'équation 3.13. Le vecteur z peut donc être utilisé pour accomplir une tâche auxiliaire.

En choisissant z=0, tel qu'utilisé en 3.13, la fonction de coût suivante est minimisée :

$$\varphi(\boldsymbol{\tau}) = \boldsymbol{\tau}^{T}\boldsymbol{\tau} \qquad (3.19)$$

ce qui correspond à minimiser les carrés des couples au niveau des articulations du système. En effet, en remplaçant la relation 3.15 dans la relation 3.19, l'équation résultante est :

$$\boldsymbol{\tau}^{T}\boldsymbol{\tau} = \mathbf{F}^{T}\left[\overline{\overline{\mathbf{B}}}\right]^{+T}\left[\overline{\overline{\mathbf{B}}}\right]^{+}\mathbf{F} + \mathbf{z}^{T}(\mathbf{I} - \left[\overline{\overline{\mathbf{B}}}\right]^{+}\overline{\overline{\mathbf{B}}})^{T}(\mathbf{I} - \left[\overline{\overline{\mathbf{B}}}\right]^{+}\overline{\overline{\mathbf{B}}})\mathbf{z} + 2\mathbf{F}^{T}\left[\overline{\overline{\mathbf{B}}}\right]^{+T}(\mathbf{I} - \left[\overline{\overline{\mathbf{B}}}\right]^{+}\overline{\overline{\mathbf{B}}})\mathbf{z} \qquad (3.20)$$

Puis, de la relation 3.14, il est possible de déduire que :

$$\left[\overline{\overline{\mathbf{B}}}\right]^{+T}(\mathbf{I} - \left[\overline{\overline{\mathbf{B}}}\right]^{+}\overline{\overline{\mathbf{B}}}) = (\overline{\overline{\mathbf{B}}}\overline{\overline{\mathbf{B}}}^{T})^{-1}\overline{\overline{\mathbf{B}}}(\mathbf{I} - \overline{\overline{\mathbf{B}}}^{T}(\overline{\overline{\mathbf{B}}}\overline{\overline{\mathbf{B}}}^{T})^{-1}\overline{\overline{\mathbf{B}}}) = 0 \qquad (3.21)$$

d'où :

$$\boldsymbol{\tau}^{T}\boldsymbol{\tau} = \mathbf{F}^{T}\left[\overline{\overline{\mathbf{B}}}\right]^{+T}\left[\overline{\overline{\mathbf{B}}}\right]^{+}\mathbf{F} + \mathbf{z}^{T}(\mathbf{I} - \left[\overline{\overline{\mathbf{B}}}\right]^{+}\overline{\overline{\mathbf{B}}})^{T}(\mathbf{I} - \left[\overline{\overline{\mathbf{B}}}\right]^{+}\overline{\overline{\mathbf{B}}})\mathbf{z} \qquad (3.22)$$

Il est alors clair que z=0 minimise la fonction de coût donnée par la relation 3.19.

Dans son étude sur le contrôle de forces appliquées aux pattes d'un robot marcheur forestier, Lehtinen (1996) conclue que la transformation des efforts par l'emploi d'une pseudo-inverse est une méthode pratique et efficace pour la répartition en temps-réel des forces au niveau des pattes. Il ajoute également que cette solution permet de réduire l'énergie consommée pour effectuer le travail.

Par ailleurs, Zhou (1999) propose une approche qu'il a baptisée « Friction constraint method » (FriCoM). Celle-ci, plus pratique selon lui, devrait remplacer la pseudo-inverse. Cette méthode assure une continuité des efforts générés par les pattes au niveau du sol. FriCoM vise à distribuer les forces de chacune des pattes considérant le cône de friction. La répartition des forces tient compte des forces horizontales et verticales. Cette approche est aussi développée pour être utilisée en temps réel, donc elle se calcule assez rapidement. Dans cet article, la méthode n'avait pas été expérimentée et l'on ne dit pas comment estimer le coefficient de friction.

3.4 Simulation

La simulation est programmée avec MatLab/Simulink. La simulation H4P_Fin1 utilise le modèle dynamique complet d'Haedus nommé H4P_2D. Les annexes E et D présentent respectivement, le programme et le modèle utilisé pour la simulation finale.

Cette simulation a permis de développer le contrôleur avant même d'effectuer les premiers essais sur le robot réel; la partie mécanique et le

contrôleur ont ainsi été développés en parallèle. Un autre avantage important est que la simulation permet d'effectuer les premiers essais qui sont souvent les plus critiques.

Pour en simplifier sa compréhension, le programme de simulation peut être divisé en trois parties : la génération de trajectoire, le suivi de trajectoire et la simulation du robot réel. La trajectoire est définie telle qu'exposée à la section 3.3.3. Le suivi de trajectoire est assuré par le contrôleur énoncé dans le présent chapitre et la simulation du robot est possible grâce à la modélisation exposée au deuxième chapitre.

La simulation utilise les trajectoires dans l'espace cartésien pour le corps et les pattes. Les couples nécessaires pour produire le mouvement sont calculés à l'aide des lois de commande (3.4) et (3.13). Un contrôleur PD est utilisé sur la trajectoire qui est définie en position, vitesse et accélération. Les couples précalculés sont ensuite envoyés à la fonction *Forward Dynamic* qui simule le robot. (Voir le programme de simulation à l'annexe E) Cette fonction retourne les états du robot, soit les positions et les vitesses généralisées.

La condition de changement de pattes est gérée par la trajectoire désirée pour simplifier la réalisation. Pour simuler l'effet d'un sol parfaitement rigide, la force appliquée par une patte contre celui-ci est calculée et injectée à l'inverse comme perturbation à l'extrémité de cette dernière. Cet astuce permet de satisfaire la condition 2.30 relativement au mouvement des pattes au sol.

3.4.1 Développement de la simulation

La dernière simulation est celle effectuée avant d'expérimenter la démarche sur le robot. Pour les premiers pas, la démarche "Amble" a été choisie en raison de sa stabilité naturelle. Cette dernière est la démarche la plus simple pour un quadrupède. Elle consiste à déplacer le corps sur trois pattes pendant que la quatrième va se repositionner pour le prochain pas. Le robot peut donc conserver un équilibre statique tout au long du parcours.

La simulation doit être initialisée avant son exécution. Le fichier initialize.m se trouve à l'annexe G et les principaux aspects en sont expliqués à l'annexe F. Il permet d'initialiser les variables du modèle SYMOFROS, la position initiale du robot et la définition des trajectoires. La simulation utilise un pas fixe de une milliseconde, en vue d'être implanté par la suite dans un système en temps réel. La méthode d'intégration utilisée est Ode4 (Runge-Kutta). Il n'y a pas eut d'étude effectuée sur les méthodes d'intégration, Ode4 et Ode5 donnaient les mêmes résultats à l'exception que Ode4 est plus rapide. Le pas de calcul à cette étape-ci a été fixé par essais et erreurs.

3.4.1.1 Interface de visualisation graphique

Pour visualiser le robot et le triangle de support, deux outils graphiques ont été développés : une interface de visualisation en deux dimensions donnant une vue de côté du robot et une interface de visualisation du triangle de support avec la projection de son centre de masse à l'intérieur.

Les coordonnées cartésiennes du corps, des hanches, des genoux et des sabots du robot sont envoyées à l'interface qui exprime la position du robot dans le plan X,Y. Le centre de masse est exprimé par le point au centre du corps.

Dans l'interface de droite présentée à la figure 3.2, le triangle formé sur le plan X,Z, représente la zone de stabilité dans laquelle le centre de masse devrait se situer. Le point le plus à droite en dehors du triangle indique la position de la patte libre. Le fait que le centre de masse se trouve à la limite de la zone stable au moment du changement de patte amène un débalancement ponctuel.

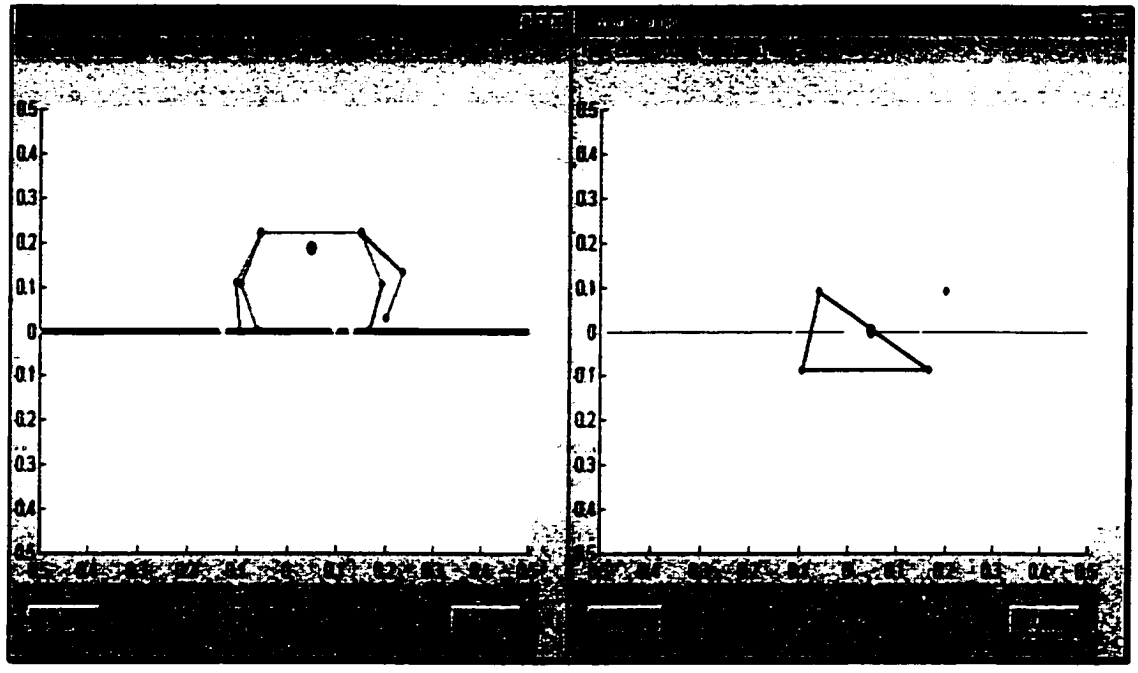


Figure 3.2 Interfaces graphiques de visualisation et de son triangle de support

3.4.2 Description de la simulation

La simulation réalisée pour valider l'approche de contrôle exécute une démarche appelée "Amble". La démarche débute à une position initiale préétablie et accélère lentement jusqu'à une vitesse de 0.025 m/sec. La séquence

de marche effectuée par le robot est la suivante [4,2,3,1]. La patte 4 est la première à se lever, puis une fois qu'elle est déposée, c'est au tour de la patte 2 se lève et ainsi de suite. Les trajectoires ont été obtenues à l'aide de polynômes générés selon l'approche 3.3.3 à partir d'un programme mis en référence à l'annexe B nommé trajssm.c et réalisé par Capra. Les paramètres qui définissent la démarche utilisée sont présentés dans le tableau 3.1.

Tableau 3.1

Paramètres utilisés pour la démarche "Amble"

Paramètre	Dimension	% de la longueur totale d'une patte
Hauteur du centre de masse du corps par rapport au sol	0.1865 m	82%
Longueur des pas	0.0687 m	30%
Hauteur du pas	0.04 m	17%
Vitesse de marche	0.025 m/s	-
Coefficient de recouvrement	25%	-

Le pourcentage de la longueur de la patte ne donne pas d'information supplémentaire mais permet de comparer un robot à un autre en considérant les paramètres de marche utilisés éliminant ainsi le facteur de dimension.

3.4.3 Résultats

Les positions cartésiennes sur l'axe des X et l'axe des Y sont présentées sur la figure 3.3. En X, l'axe horizontal, la position de l'extrémité de chaque patte demeure constante jusqu'à ce qu'elle quitte le sol pour effectuer son pas. La position du corps progresse selon une accélération de 0.005 m/ses. Le corps atteint la vitesse de 0.025 m/sec au bout de la cinquième seconde. En Y, il est possible de voir chacune des pattes se soulever tour à tour, selon l'ordre prescrit : [4, 2, 3, 1]. Le pas, plus lent au début, atteint une largeur constante après 5 secondes.

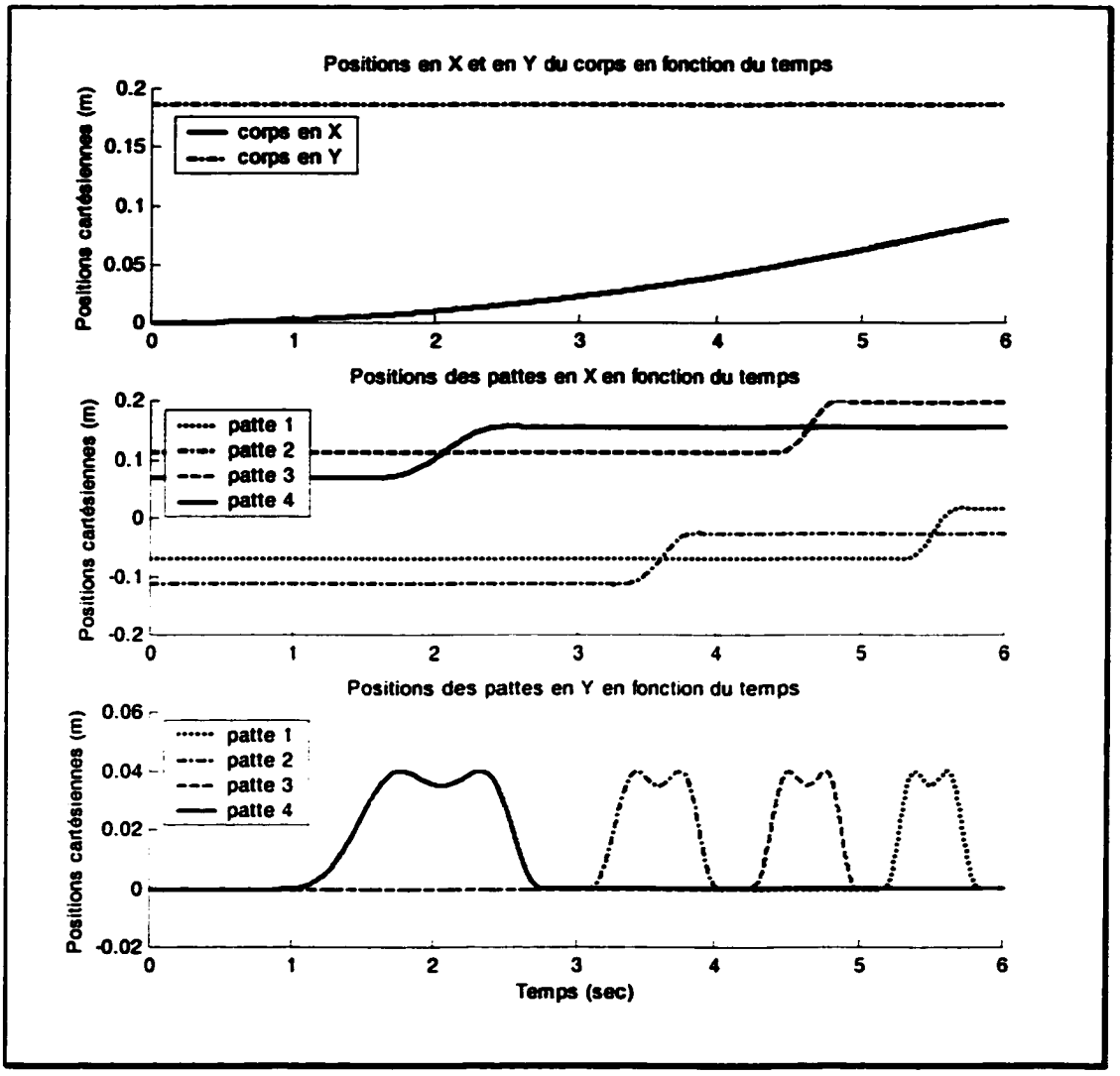


Figure 3.3 Positions du robot suivant une démarche "Amble"

Pour vérifier la performance du contrôleur, la figure 3.4 présente les erreurs de positions dans le temps en X et en Y. Les erreurs mesurées sont de l'ordre du centième de millimètre. L'erreur sur la position et l'orientation du corps demeure nulle tout au long de la course alors que lorsqu'une patte se

dépose au sol, elle conserve une légère erreur. L'erreur la plus grande survient lorsque la patte est au milieu de sa trajectoire.

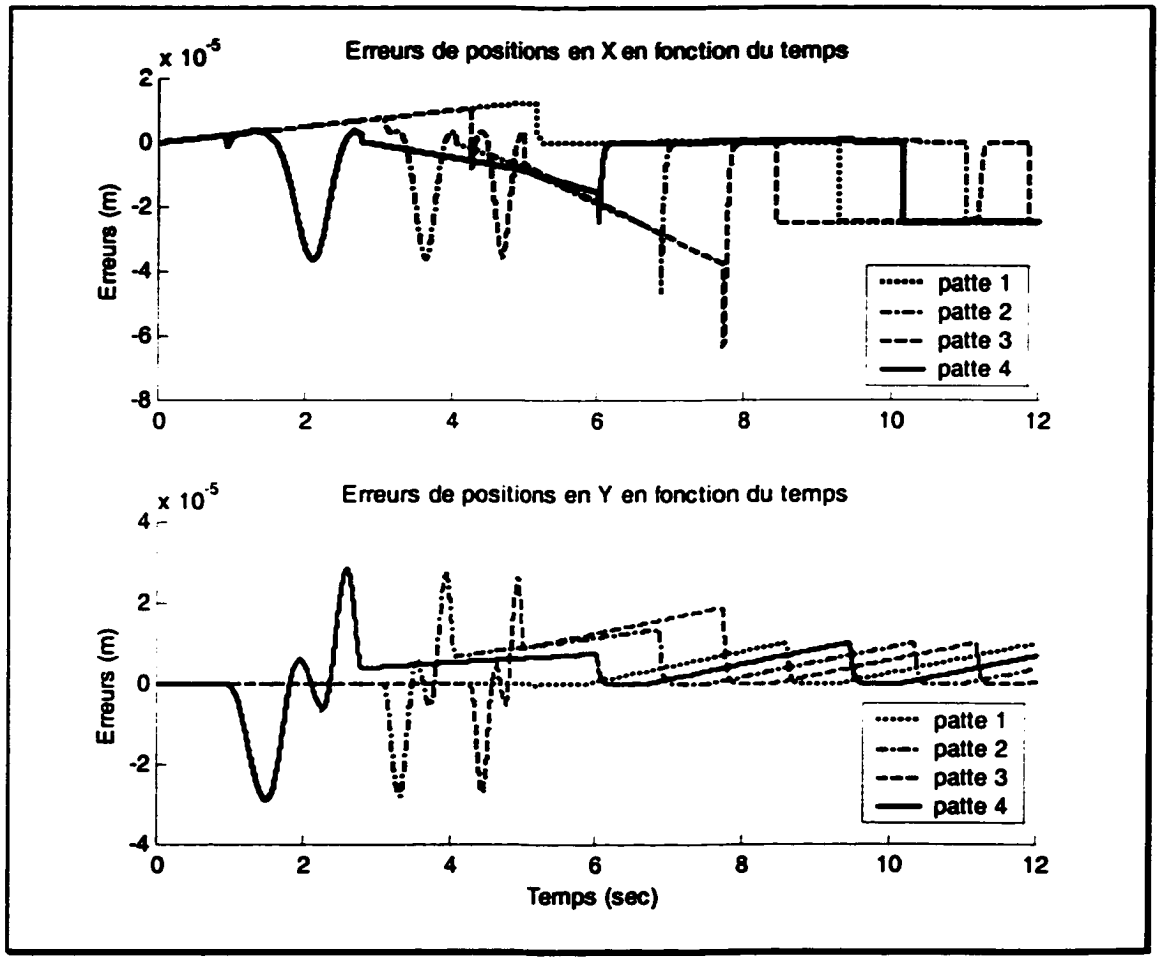


Figure 3.4 Erreurs de positions cartésiennes

Les vitesses cartésiennes sont présentées en X et en Y sur la figure 3.5. Encore une fois, la vitesse de croisière est atteinte à la 5e seconde comme le démontre la courbe de vitesse du corps en X. Les trajectoires de pattes suivantes sont donc identiques à la dernière figurant entre la 5e et la 6e seconde de

simulation. Il est important de vérifier la vitesse des pattes libres demandée suivant les paramètres de marche utilisés. Par exemple, sur ce graphique, la vitesse en X pour la patte libre est 16 fois plus grande que la vitesse du corps. Cela impose la vitesse avec laquelle le robot peut se déplacer. Cet écart peut être diminué par l'emploi d'un coefficient de recouvrement plus faible, des pas plus grands et une trajectoire plus près du sol.

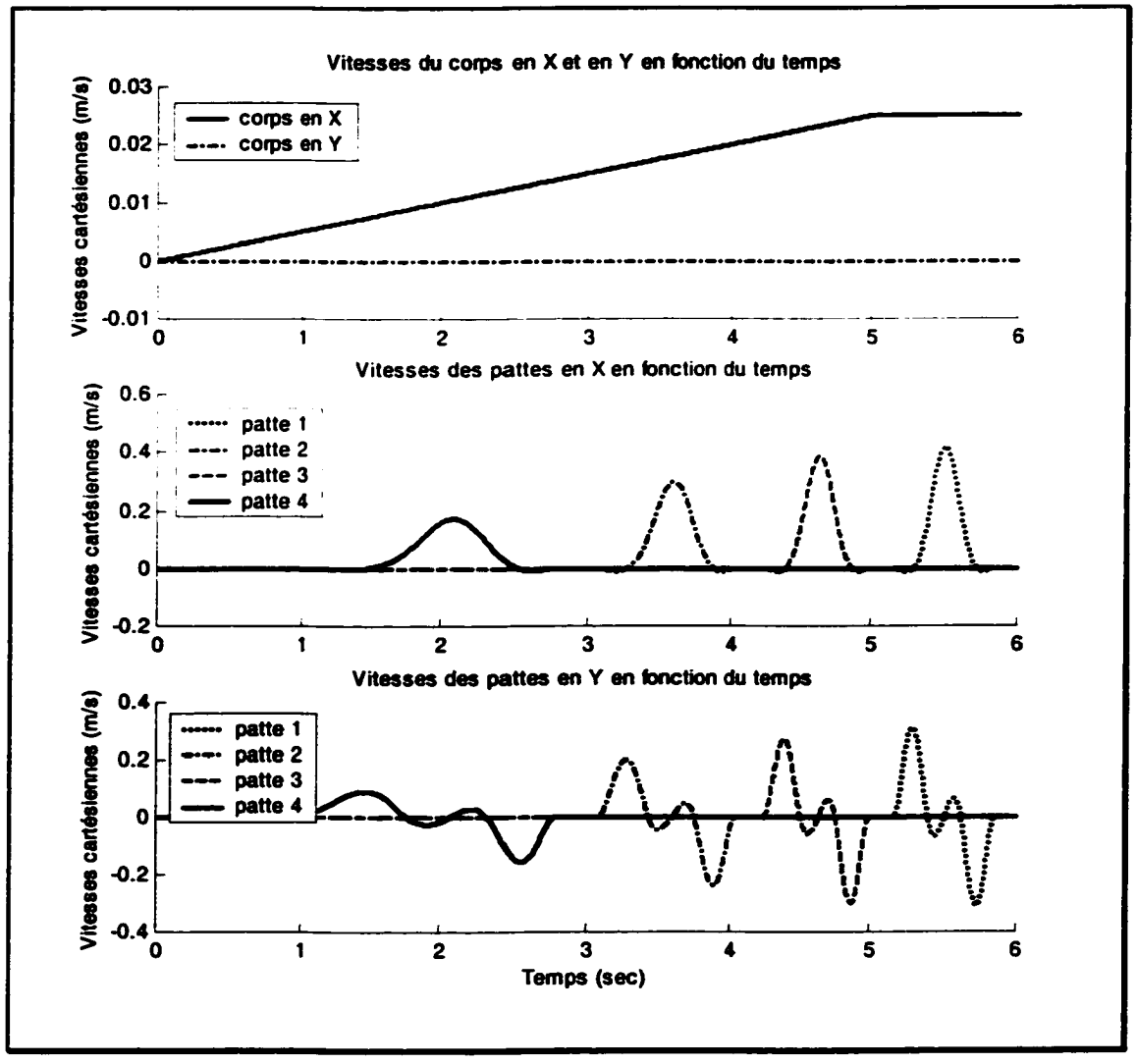


Figure 3.5 Vitesses du robot suivant une démarche "Amble"

Les erreurs de vitesses sont aussi très faibles. De l'ordre du dixième de millimètre par seconde, elle augmentent proportionnellement à la vitesse de déplacement des pattes. tel qu'illustré sur la figure 3.6. Une fois que le corps a atteint une vitesse constante, les erreurs diminues comme c'est le cas de la patte 1.

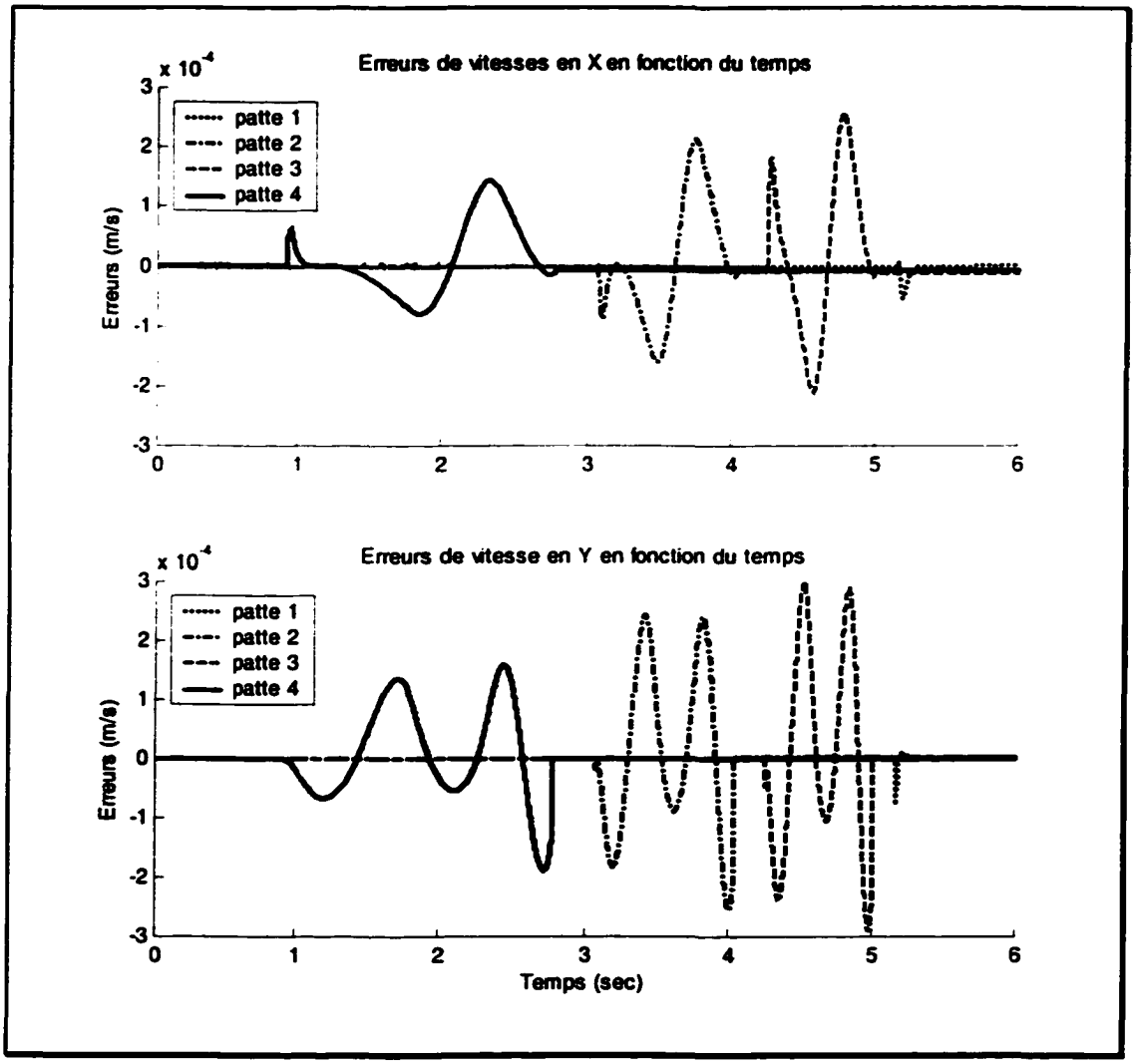


Figure 3.6 Erreur de vitesses cartésiennes

Les couples demandés pour la démarche "Amble" avec les paramètres de marche utilisés apparaissent sur la figure 3.7. Les discontinuités s'expliquent par les changements de pattes. Les couples sont aussi plus grands lorsque la patte supporte le corps comparativement au moment où elle suit une trajectoire libre. Il est important de vérifier que les couples demandés tout au long de la démarche demeurent inférieurs à ce que les actionneurs du robot peuvent donner. Pour réduire ces couples, la hauteur désirée pour le corps peut être augmentée et les pas réduits en longueur et en hauteur.

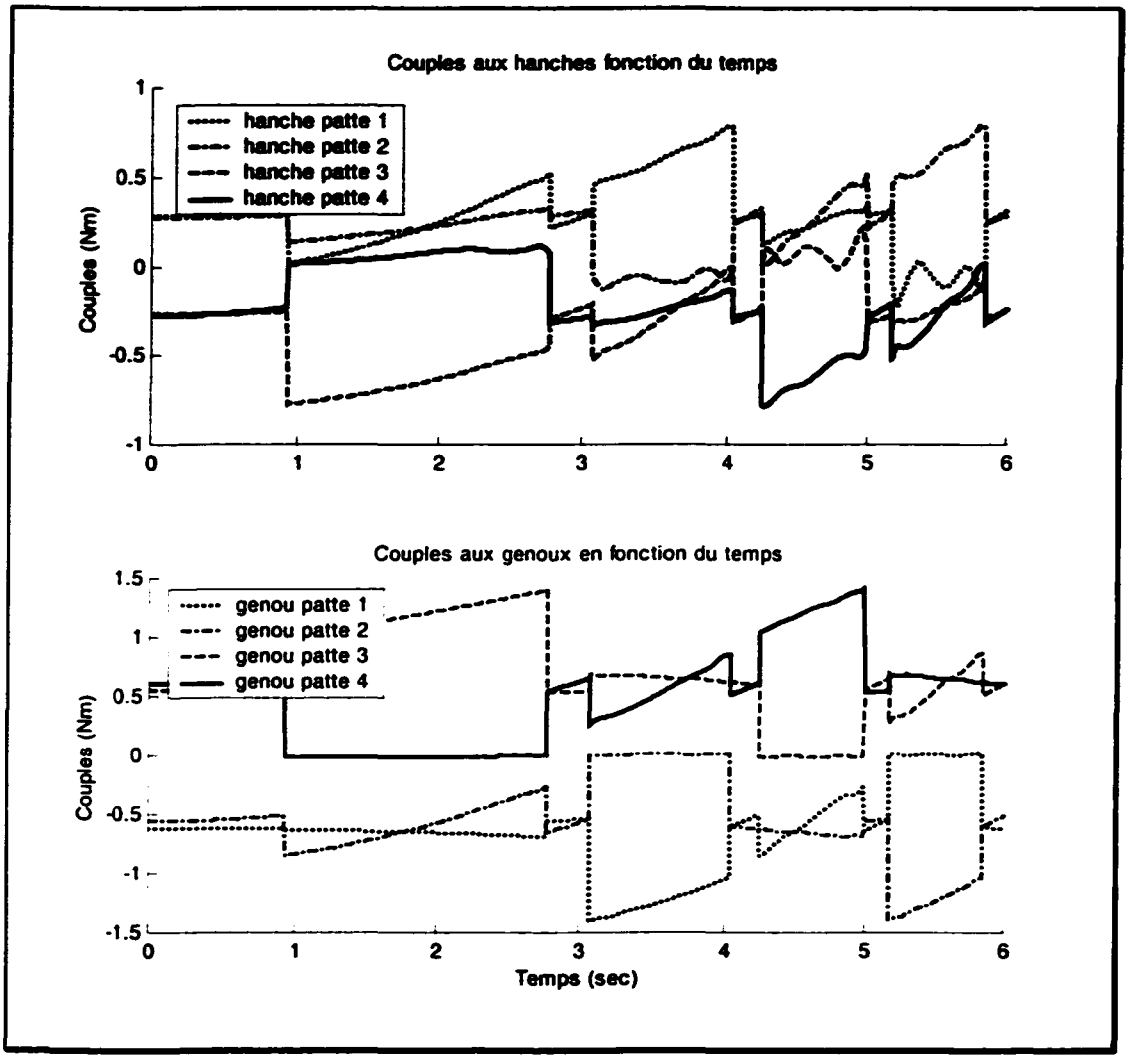


Figure 3.7 Couples appliqués au robot pour la démarche "Amble"

Cette démarche est exactement celle empruntée par le robot lors de l'expérimentation présentée à la section 4.6.3. C'est pourquoi la vitesse de marche a été limitée à 0.025 m/sec.

En simulation, il serait facile de croire qu'il n'y a pas de vitesse maximale, mais en réalité, au-delà d'une certaine vitesse, le robot quitte le sol et les conditions de marches ne sont plus respectées. De plus, la démarche "Amble"

est assez limitée en vitesse car la patte libre demande des accélérations trop grandes. À titre indicatif, la vitesse maximale obtenue pour le corps avec la démarche "Amble" en changeant les paramètres de marche était de 0.8 m/sec, soit plus de cinq pas à la seconde. Les couples demandés atteignaient des pointes de 15 Nm pour la patte libre. Pour les pattes portantes, ils demeuraient en dessous de 1 Nm.

Plus le corps est bas, plus les couples sont grands pour les pattes portantes. L'augmentation des couples est proportionnelle à la hauteur du corps. Par exemple, à une hauteur équivalente à 95% de la longueur des pattes, les couples pour les pattes portantes sont au maximum de 0.6 Nm, alors qu'à 80% ils doublent.

Plus les pas sont grands et le recouvrement faible, plus la patte libre a de temps pour effectuer sa trajectoire ; les accélérations diminuent alors beaucoup. Il en va de même pour les couples requis pour le mouvement. La hauteur du pas est aussi un facteur qui peut devenir exigeant au niveau de la trajectoire de la patte libre.

Une fonction de reconnaissance de la position et de la vitesse du robot à partir des pas effectués et de la vitesse des articulations « dead reckonning » a été développée et validée en simulation. Elle a pour but d'évaluer la position et l'orientation du robot lors de l'exécution de la démarche en temps réel. La différence entre la position donnée par la simulation et la position donnée par la fonction « dead-reckonnnig » était en moyenne pour la position verticale et l'orientation. De façon similaire, les erreurs de vitesses demeuraient très faibles. Par contre, en X, l'erreur augmentait légèrement dans le temps, puisqu'aussi faible soit l'erreur de l'estimateur de position, la sommation des positions et par

le fait même, des erreurs amène une dérive dans le temps par rapport à la position réelle.

3.5 Conclusion

Ce chapitre présentait le contrôleur d'Haedus dont l'approche est basée sur le modèle vu dans le chapitre 2. Les trajectoires doivent assurer le synchronisme des pattes pour faire bouger le corps tandis que les efforts fournis par les pattes au sol sont répartis de façon à minimiser l'énergie dépensée par le robot et à faire bouger le corps de celui-ci. Avec l'approche de contrôle présentée dans le présent chapitre, il est possible de contrôler la trajectoire du corps et des pattes libres en position, vitesse et accélération en tenant compte de la dynamique du robot. En plus de marcher, le robot peut ainsi prendre une configuration particulière, une posture, pour effectuer par exemple un travail sur place.

CHAPITRE 4

EXPÉRIMENTATION ET VALIDATION

4.1 Introduction

L'expérimentation est une partie importante de ce travail. Elle permet de valider l'approche et les hypothèses de départ dans un monde réel. Avant de présenter tout le matériel nécessaire à la réalisation de ce projet, il est bon de rappeler le but initial : développer une approche pour concevoir des contrôleurs de robots marcheurs basée sur un modèle dynamique complet. Ce chapitre présente l'environnement de travail RT-LAB, une description du matériel et des logiciels utilisés, les méthodes adoptées pour réaliser les expériences et les résultats obtenus.

4.2 Description de la plate-forme matérielle

Le système de contrôle temps réel sur lequel les expérimentations ont été faites est constitué d'une console de programmation liée à un ordinateur appelé nœud de calcul dans lequel les cartes d'acquisitions permettent le lien avec la partie électrique et mécanique du robot Haedus. (Voir l'annexe I pour la description mécanique du robot Haedus). Voici une photo du montage ainsi qu'un schéma conceptuel des raccordements. Notez que les composants illustrés sur la figure 4.1 sont identifiés selon la même disposition sur la figure 4.2.

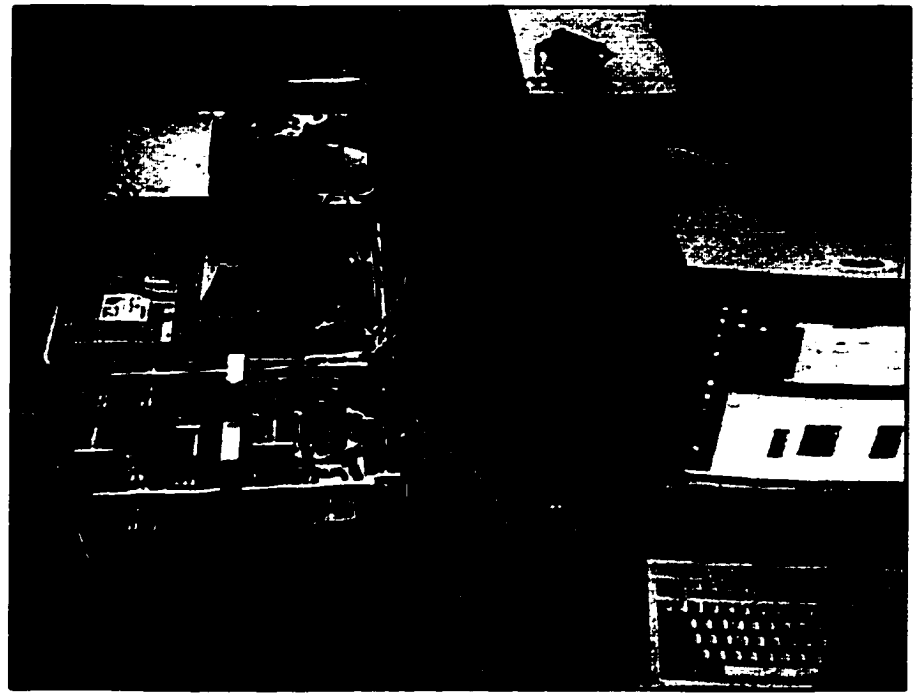


Figure 4.1 Photo du montage expérimental

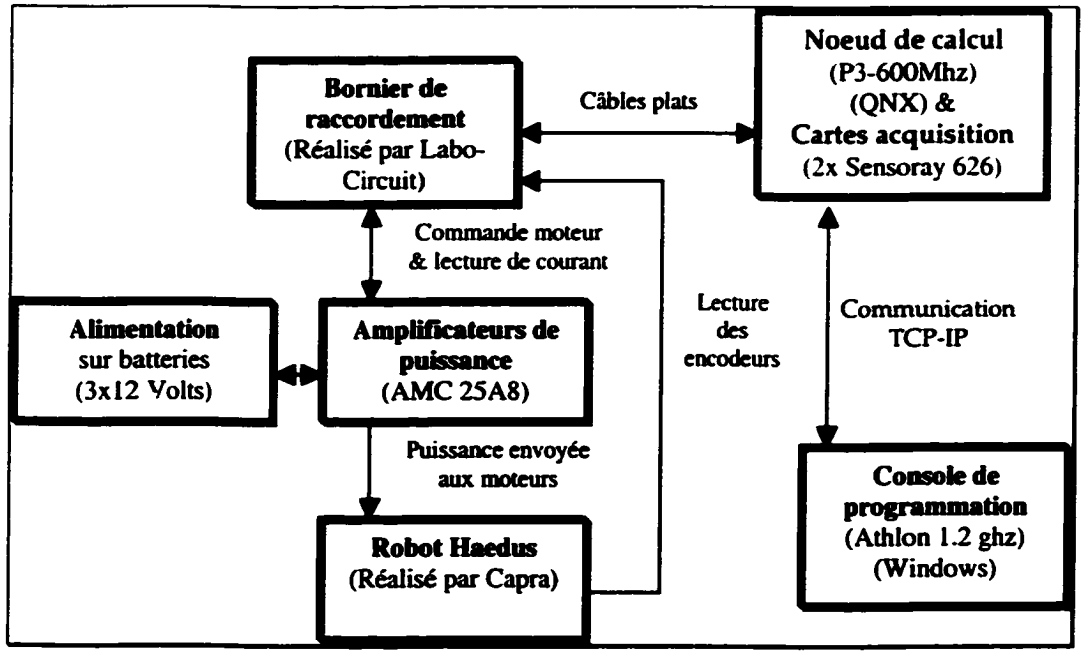


Figure 4.2 Schéma de raccordement du matériel

4.3 Description de la plate-forme logicielle

L'implantation temps réel est effectuée à l'aide de l'environnement de travail de haut niveau de la compagnie Opal-RT, nommé RT-Lab. La programmation se fait sur MatLab/Simulink et le code généré par Real Time Workshop (RTW) de MatLab est transféré par RT-Lab dans le nœud de calcul QNX avant d'être compilé par Watcom C++ et exécuté par QNX sur le même nœud de calcul. Lorsque le programme compilé est exécuté en temps réel sur le nœud de calcul, il communique certains signaux présélectionnés par l'usager à la console pour une visualisation temps-réel. Ceci ajoute une grande facilité pour la compréhension et l'analyse du système lors de son exécution. La figure 4.4 expose les étapes à franchir pour arriver à l'exécution dans un environnement temps réel tel que définit par RT-Lab avec l'incorporation d'une bibliothèque de fonctions représentant le modèle défini par SYMOFROS. Ce dernier est présenté au chapitre 2. (Voir l'annexe B pour la liste des logiciels utilisés et leurs versions)

4.3.1 Étapes de génération du code temps-réel

Avant d'arriver à l'exécution du programme final en temps-réel, les étapes présentées sur la figure 4.4 furent exécutées. D'abord la définition du modèle a été effectuée avec le logiciel SYMOFROS par l'usager (voir la section 2.9.1). Ensuite les calculs conduisant au modèle dynamique du robot ont été effectués par MAPLE. Ce dernier a également traduit les calcul en code C. Par la suite, une librairie DLL a été compilée sur MATLAB avec ce même code C. Cette librairie utilisé par WINDOWS permettait de créer un programme de simulation en temps différé sur SIMULINK et développer les contrôleurs. Le tout étant exécuté sur MATLAB. Une fois cette étape de développement fonctionnelle et

satisfaisante, le programme SIMULINK comprennent la modèle dynamique fut traduit en C par RTW de MATLAB sous la commande de l'interface RT-LAB. Ce dernier a ensuite transféré le code C par réseau y compris le code C du modèle provenant de MAPLE au nœud de compilation et de calcul QNX. Une fois le transfert complété, RT-LAB lance la compilation du code C sur QNX avec le logiciel WATCOM C++ de QNX. Enfin, l'interface RT-LAB permet de charger en mémoire le programme de simulation/expérimentation et de l'exécuter. Ce processus est répété tout au long du développement.

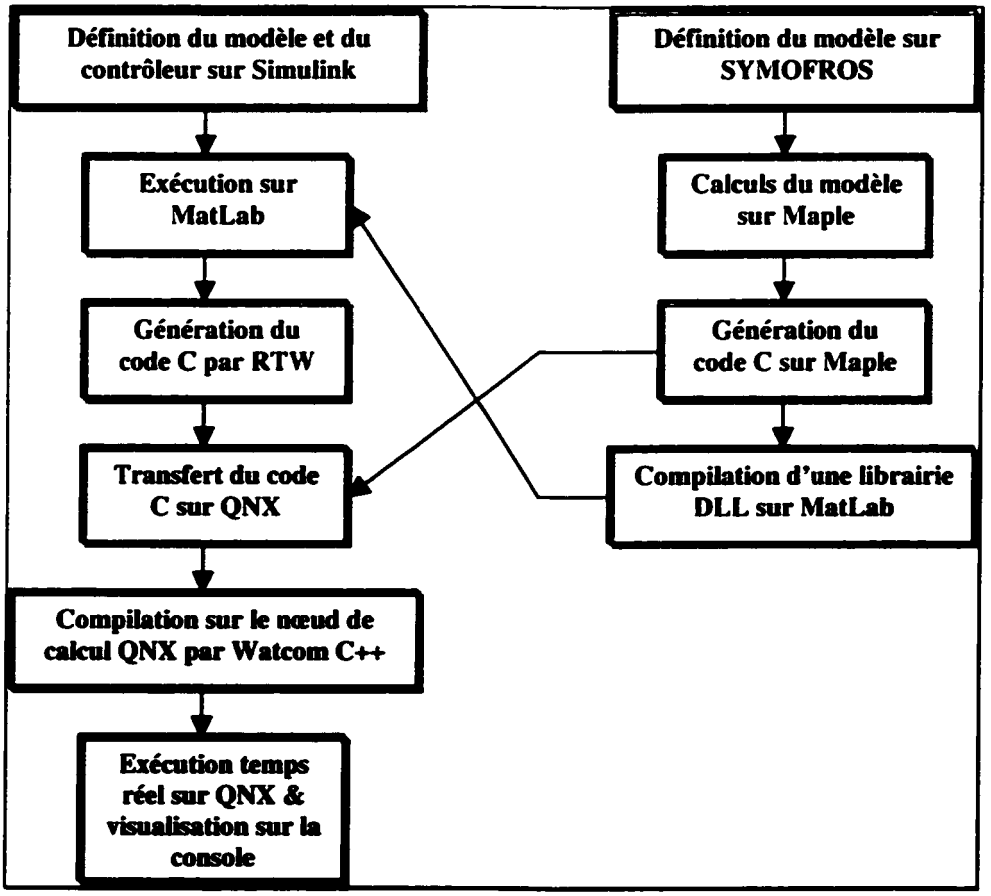


Figure 4.4 Étapes de développement logiciel et mise en exécution temps réel

4.4 Environnement de travail RT-Lab

« Real-Time Laboratory » (RT-Lab) est un logiciel développé par Opal-RT inc. Ce logiciel procure à l'usager un véritable environnement de travail pour l'expérimentation temps réel de programmes créés sur Simulink. RT-Lab effectue l'intégration des diverses composantes dans le but de rendre transparent à l'usager la phase de génération, de transfert et compilation du code développé sur Simulink en vue de préparer l'exécution sur un ou plusieurs nœuds de calcul. De plus, RT-Lab effectue la synchronisation des signaux entre les différents nœuds de calculs et la console. En règle générale, un modèle Simulink, peut être exécuté dix fois plus rapidement une fois compilé. Voici à quoi ressemble la fenêtre du panneau de contrôle de la console :

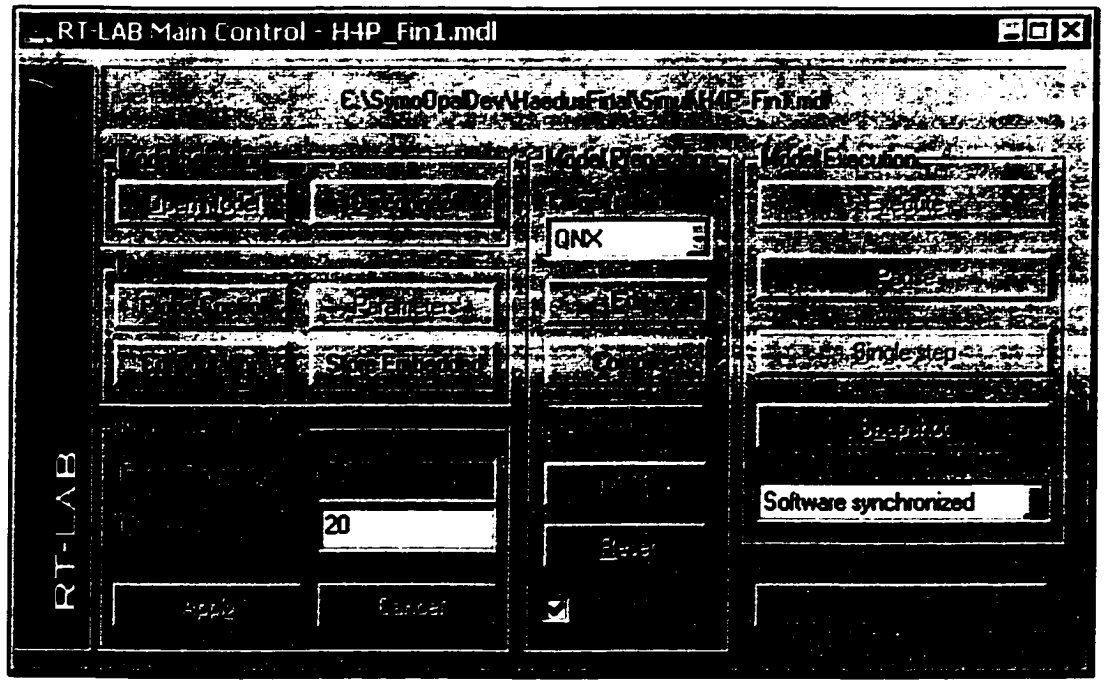


Figure 4.5 Fenêtre du panneau de contrôle principal de RT-Lab

Normalement, les étapes pour arriver à exécuter un programme sont : ouvrir le modèle, l'éditer, le compiler et l'exécuter. Lors de la première compilation, il faut assigner les nœuds de calculs et configurer différents paramètres, lesquels sont présentés dans l'annexe C. Pour optimiser la communication ou pour sauver dans un fichier les données envoyées à la console, certains paramètres peuvent être changés dans *Probe Control*.

Il y a quatre modes d'exécution : *simulation, simulation without data loss, software synchronized et harware synchronized.* En simulation, les calculs sont effectués le plus rapidement possible sans synchronisation. La simulation sans perte de données assure que tous les signaux sont bien envoyés avant de passer au pas de calcul suivant. En synchronisation logicielle, le temporisateur de QNX sert de base de temps alors que dans la synchronisation matérielle, c'est la carte d'acquisition qui sert de base de temps.

Les icônes RT-Lab servent à définir les ports de communication, font les liens avec les différents modules d'entrées/sorties et donnent des informations sur le programme tel que le temps de calcul par cycle.

Les programmes Simulink doivent être adaptés pour utiliser RT-Lab. Le code à exécuter sous chaque ordinateur est identifié par un sous-système dont le nom porte un préfix particulier (SC_, SM_ ou SS_). Par exemple, dans le cas où il y a un seul nœud de calcul, il portera le nom de sm_master et la console portera le nom de sc_console. Lorsque l'on ajoute d'autres ordinateurs comme nœuds de calcul, les noms utilisés peuvent être sl_slave avec un numéro par exemple. Cette première division permet de définir comment sont répartis le code et les signaux de communication. Voici un exemple :

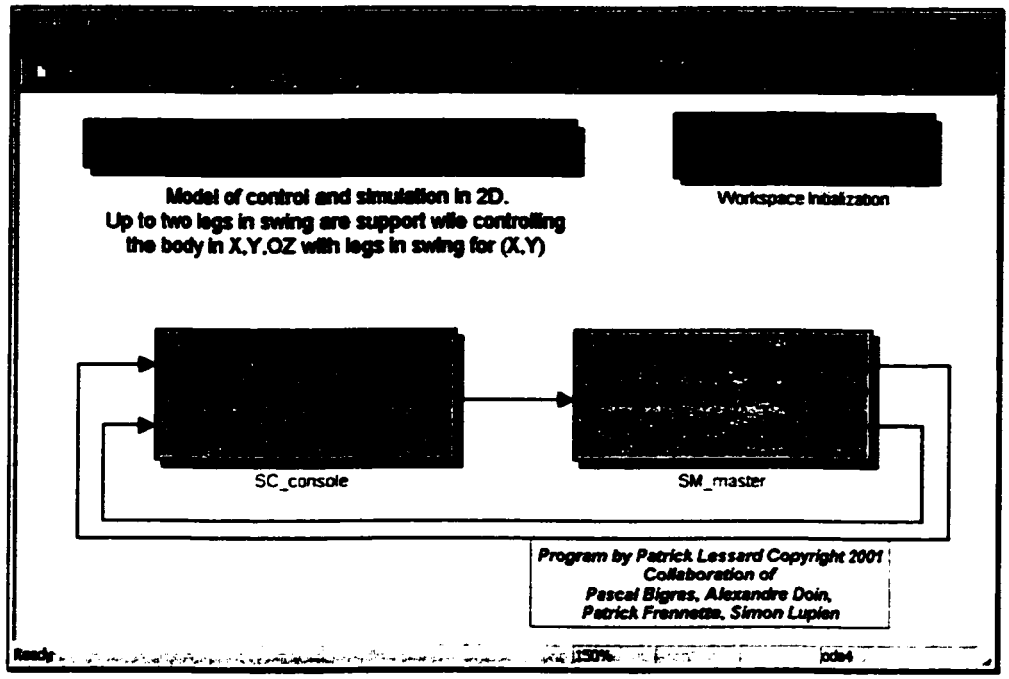


Figure 4.6 Premier niveau de programmation adapté à RT-Lab

Ceci n'est qu'un survol pour donner une idée au lecteur de ce qu'est l'interface RT-Lab. Pour plus de détails, veuillez vous référer au manuel d'utilisation de la compagnie Opal-RT.

4.5 Description des programmes réalisés

L'annexe A présente la liste des programmes réalisés au cours de ce projet. Les quatre principaux sont les suivants :

- SOT1.mdl
- SOT4.mdl
- H4P_Fin1.mdl
- H4P_FinR1.mdl

Le premier programme, SOT1, a permis de valider la méthode de modélisation employant SYMOFROS et l'utilisation de la plate-forme temps réel RT-Lab. Il comporte un modèle dynamique très simple d'une patte du robot Haedus. Les expériences présentent la comparaison entre le modèle théorique d'une patte libre et les résultats expérimentaux.

Le deuxième programme SOT4, comporte un modèle simple des pattes du robot sans le corps. Il a été utilisé pour une démonstration effectuée devant un public au mois de mars 2001. Il est composé d'un contrôleur PID appliqué directement aux articulations du robot ainsi que de la cinématique du passage des articulations aux moteurs et de la dynamique des moteurs (inertie). La démarche présentée alors n'était pas très fluide.

Le troisième est le modèle complet du robot Haedus dans un plan. Il est utilisé sur MatLab pour effectuer les simulations en temps différé.

Le quatrième est le programme prévu pour l'expérimentation finale de Haedus. Ce modèle est exactement le même que le précédant, le bloc simulant le robot, le « *Forward Dynamic* », ayant été remplacé par le robot réel via le lien avec les cartes d'entrées/sorties.

L'annexe D présente les modèles SYMOFROS utilisés et l'annexe E donne un aperçu des programmes mentionnés ci-dessus.

4.6 Description des expériences

Parmi la série d'expériences réalisées, trois sont particulièrement importantes pour démontrer la validité des concepts empruntés : la validation

des techniques de modélisation, la validation du contrôleur de couple précalculé et la réalisation d'une démarche stable de type "Amble".

4.6.1 Validation du modèle dynamique d'une patte

Pour effectuer cette première expérience, un modèle d'une patte du robot Haedus a été défini avec SYMOFROS pour être comparé en temps réel avec la patte du robot. La même trajectoire fut assignée aux articulations des deux robots (modèle et réel) pour comparer par la suite, les positions, les vitesses et les couples du modèle avec ceux du robot.

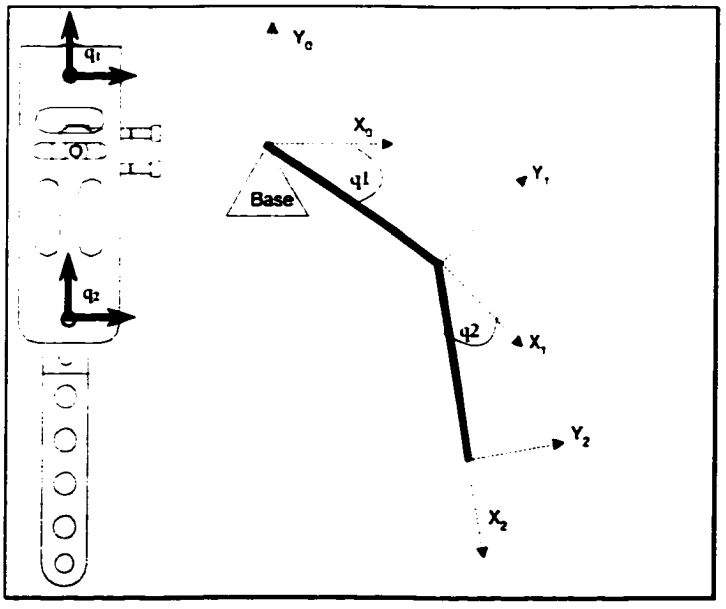


Figure 4.6 Comparaison d'une patte avec le modèle d'une patte

4.6.1.1 Objectifs atteints

Cette première étape a permis de valider les aspects suivant :

- plate-forme QNX;
- cartes d'acquisitions;
- amplificateurs de puissance;
- réponse du système : moteur, ampli, carte acquisition;
- moteurs et encodeurs;
- environnement RT-Lab;
- création de modèle SYMOFROS;
- commande de position à l'aide d'un PID;
- transformation des positions et des couples de l'espace articulaire à l'espace des moteurs, et inversement;
- comparaison du système réel avec la simulation;
- méthode de comparaison.

4.6.1.2 Méthodologie

Les trajectoires sont définies en position seulement. Une loi de commande PID, dont les gains ont été calculés avec un temps de réponse désiré de 0.1 seconde et un dépassement nul, permet de suivre cette trajectoire. Le robot comporte deux particularités dont le modèle original ne tient pas compte : la friction, et le couplage entre les actionneurs et les articulations. La friction est un phénomène complexe et non linéaire comportant une partie statique, une partie dynamique. Elle ne sera pas traitée comme telle dans le modèle étant donné que ce phénomène n'apporte en général aucun effet déstabilisant. Le couplage par contre doit être considéré pour la commande du robot.

Le modèle à contrôler est approximé par la relation 4.1 suivante :

$$\boldsymbol{\tau}_r = \mathbf{Mjtot} \cdot \ddot{\mathbf{q}} + \mathbf{F}(\dot{\mathbf{q}}) \qquad (4.1)$$

où $\tau_{r,}$ est le couple nécessaire pour faire bouger le robot et **Mjtot** est la sommation d'une approximation de l'inertie des membres dans l'espace articulaire et de la matrice **MMJ**. **MMJ** est définie pour ajouter l'inertie des actionneurs transposée du côté articulaire. Pour compléter le modèle de simulation, cette dernière matrice est ajoutée à la matrice de masse du système pour la simulation. (Une explication supplémentaire sur la formation de ces matrices se trouve à l'annexe L).

Pour bien comprendre la distinction entre l'espace des actionneurs et l'espace articulaire, il faut analyser la figure 4.7 suivante illustrant la relation entre les deux actionneurs et les deux articulations de la patte. Le moteur de la hanche est directement relié à la hanche tandis que l'actionneur du genou fait tourner ce dernier relativement à la hanche en raison du mécanisme d'arbre double concentrique au niveau de celle-ci.

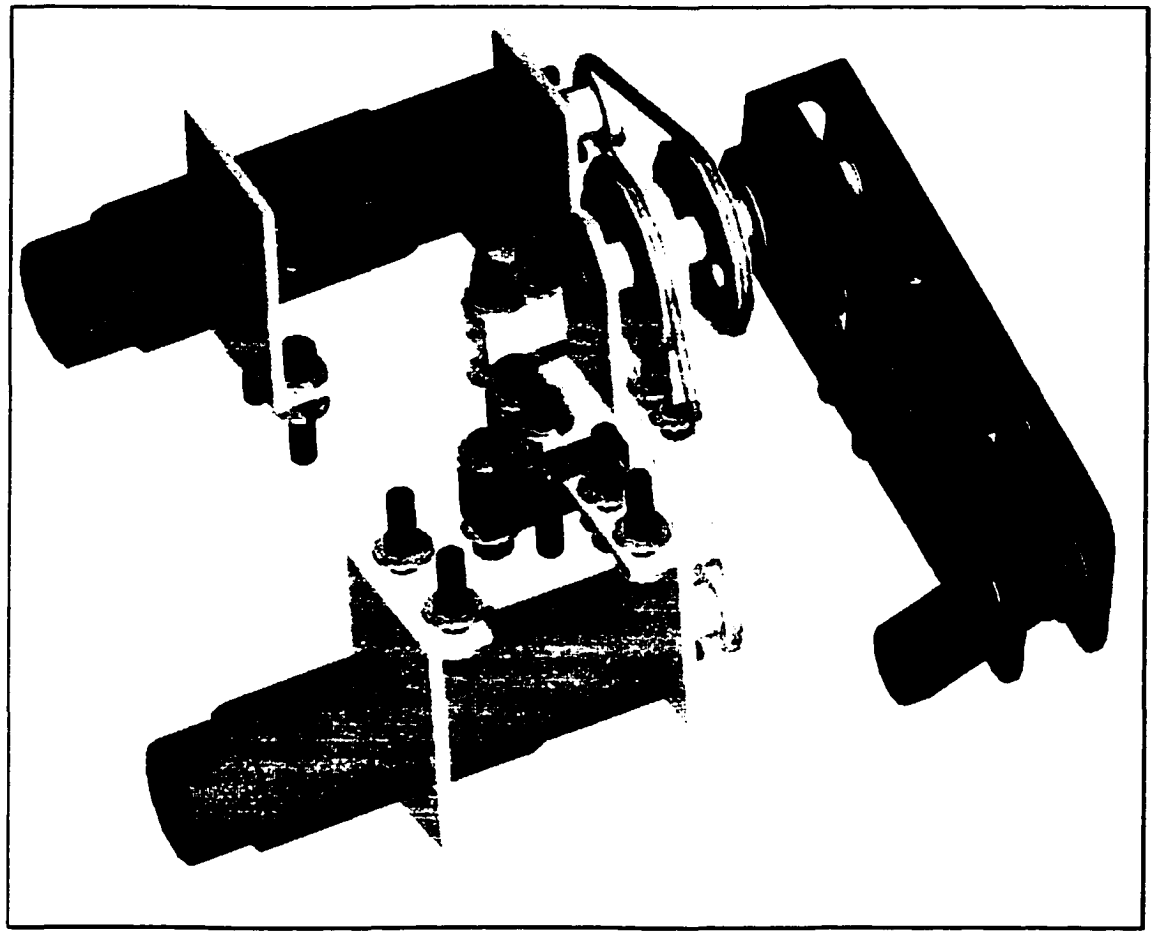


Figure 4.7 Schéma exprimant le couplage entre les moteurs et les articulations

Au niveau cinématique, quatre matrices sont nécessaires pour calculer la transformation qu'implique ce couplage entre les actionneurs et les pattes du robot : Tv_MJ, Tv_JM, Tt_MJ et Tt_JM. Ces matrices permettent de transformer les vitesses ou les couples de l'espace des articulations à celui des actionneurs (Voir l'annexe L pour la définition de ces matrices)

L'entrée du PID est une erreur de position en radians au niveau de l'articulation Hanche et Genou pour chaque patte. La sortie du PID est

multipliée par la matrice **Mjtot**. La sortie du contrôleur est en Nm, soit le couple à appliquer à l'articulation pour corriger l'erreur.

La loi de commande découplante est donc donnée par :

$$\boldsymbol{\tau}_r = \mathbf{Mjtot} \cdot \mathbf{u} \tag{4.2}$$

Où **u** est la loi de commande PID définie ci-contre :

$$\mathbf{u} = \mathbf{K}i \int \mathbf{e} + \mathbf{K}p \cdot \mathbf{e} + \mathbf{K}d \cdot \dot{\mathbf{e}} \tag{4.3}$$

En appliquant cette loi de commande au système, on obtient la dynamique de l'erreur de suivi donnée par:

$$\left(s^3 + \mathbf{K}ds^2 + \mathbf{K}ps + \mathbf{K}i\right)e = 0 \tag{4.4}$$

L'équation caractéristique correspondant à un temps de réponse Tr est alors donc donnée par :

$$s^3 + 3wn \cdot s^2 + 3wn^2 \cdot s + wn^3 = 0 \tag{4.5}$$

où :

$$wn = \frac{6.27}{Tr} \tag{4.6}$$

Les gains sont obtenus en posant l'égalité entre l'équation caractéristique du système et celle correspondant au temps de réponse désiré (Tr) pour atteindre 95% de la consigne sur une entrée de type échelon:

$$Kd = 3 \cdot wn; \qquad Kp = 3 \cdot wn^2; \qquad Ki = wn^3 \tag{4.7}$$

Ainsi, les trois pôles sont identiques et ce choix se justifie par sa simplicité de calcul et sa stabilité puisque les trois pôles sont négatifs et se retrouvent sur l'axe imaginaire.

4.6.1.3 Résultats

Les résultats obtenus sont très satisfaisants et les comportements sont très similaires à ce que présente le modèle. La figure 4.8 est un exemple de suivi de trajectoire sinusoïdale pour l'articulation de la hanche et du genou. La courbe de suivi obtenue par simulation superpose la courbe de suivi obtenue avec le robot. La commande de position utilisée est un sinus d'amplitude pi/4 pour la hanche et de -pi/3 pour le genou à une fréquence de 5 rad/sec.

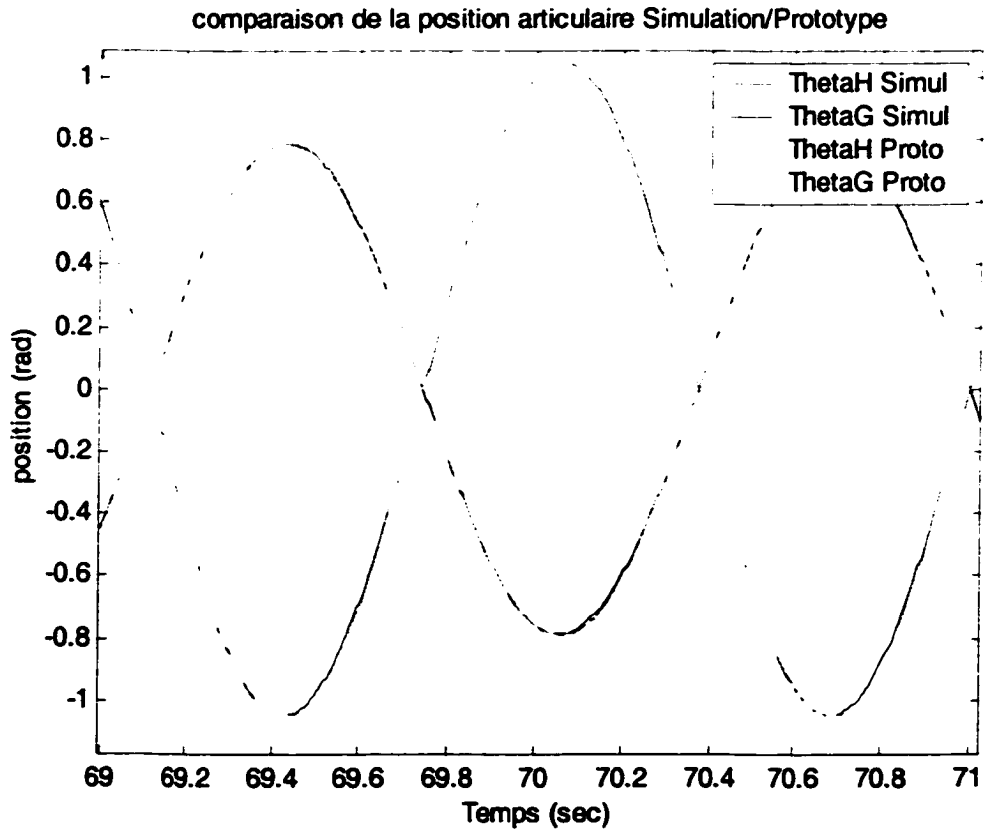


Figure 4.8 Comparaison de positions articulaires

La comparaison des vitesses est assez bonne également malgré l'ajout d'un facteur d'erreur, la vitesse réelle des articulations n'est pas disponible, elle est estimée par une approximation de la dérivée de la position obtenue à l'aide d'un filtre de la forme suivante :

$$\frac{s}{\Gamma_f s+1} \tag{4.8}$$

Les expériences effectuées ont confirmé que la constante de temps Γ_f de l'estimateur de vitesse pouvait être une source d'erreur considérable. (Plus de détails sur la fonction de transfert sont donnés à l'annexe M)

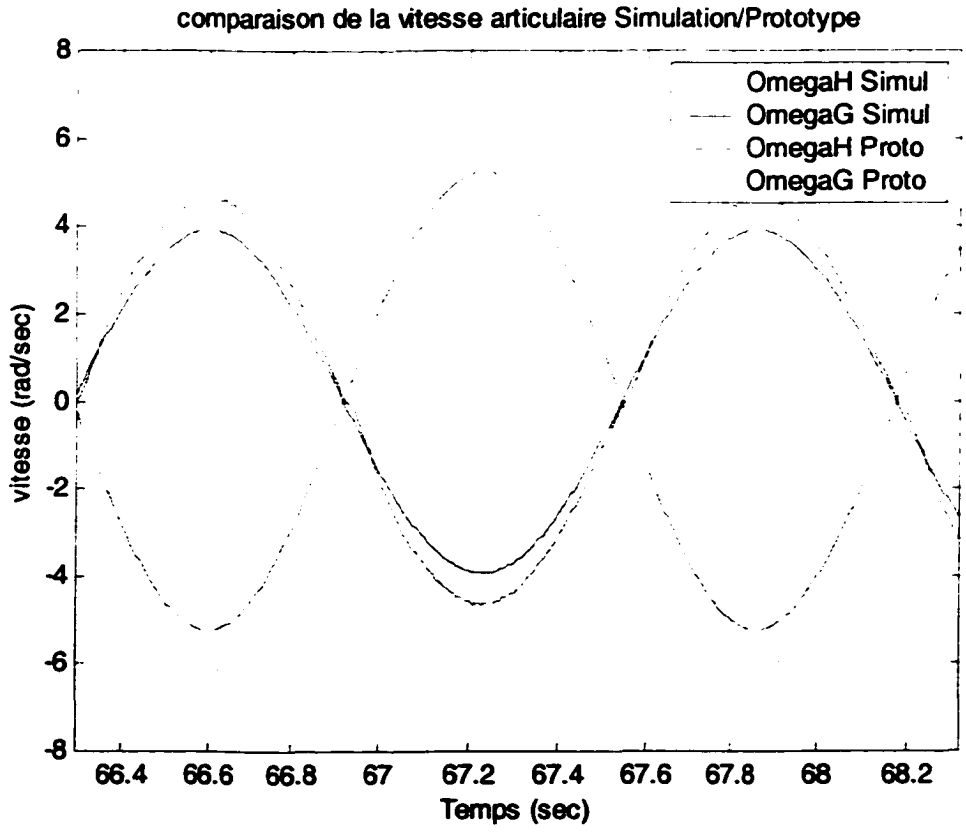


Figure 4.9 Comparaison de vitesses articulaires

Les couples réels sont bien plus grands que les couples estimés. Ceci est principalement dû à la friction au niveau des articulations. Une légère différence peut aussi être introduite par l'inertie des poulies qui a été négligée et surtout, par le rendement de toute la chaîne de transmission de puissance : moteur, engrenage, poulie. Un engrenage planétaire a un faible rendement de puissance lorsque le couple demandé est faible. Dans bien des cas, il peut passer de 75% à 30% de rendement. La hanche est représentée sur le graphique du haut et le genou, sur le graphique du bas sur la figure 4.10.

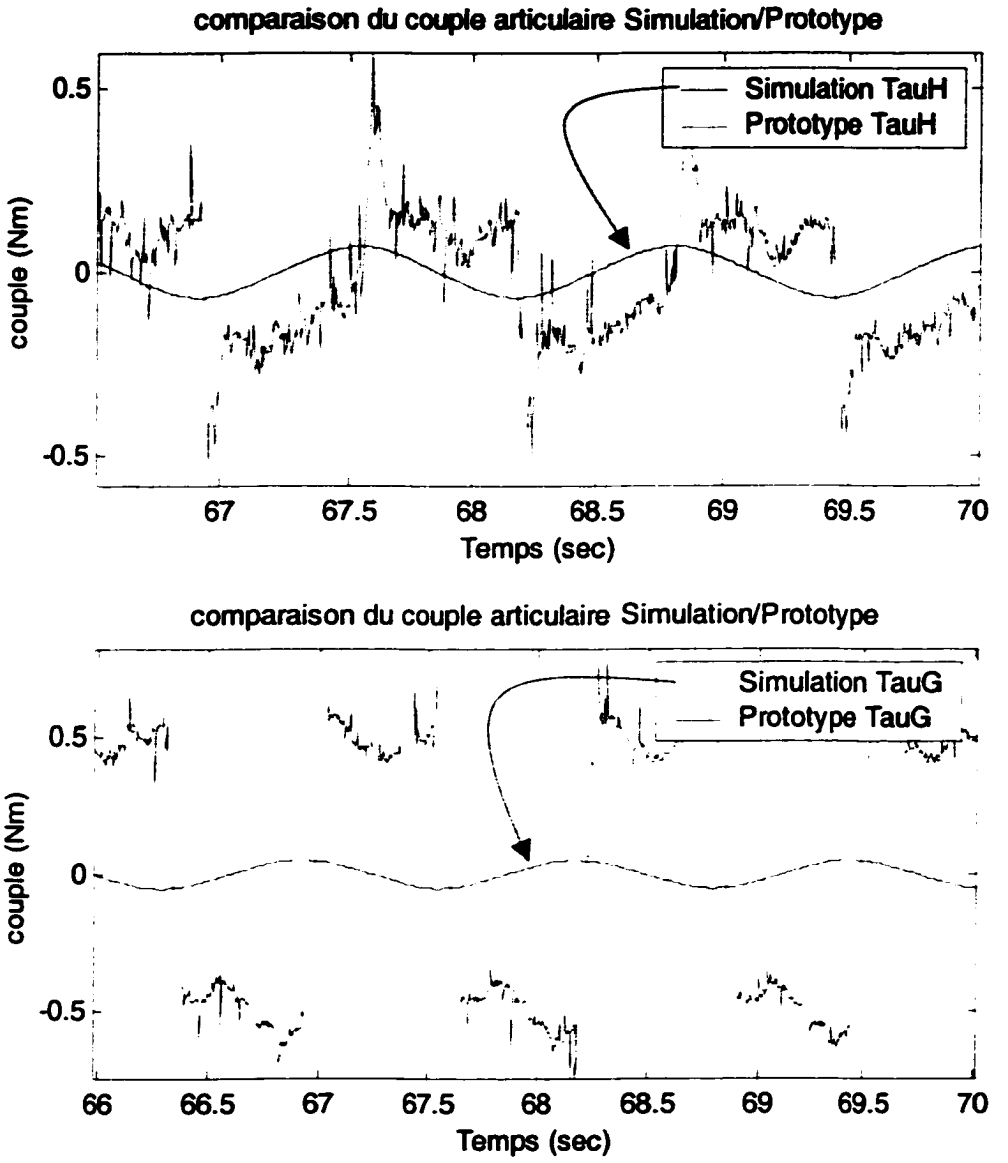


Figure 4.10 Comparaison des couples articulaires

Les couples demandés sont mesurés à la sortie du PID et les couples réels sont estimés à partir de la lecture des courants injectés aux moteurs, c'est pourquoi le signal paraît aussi bruité. L'amplificateur est un hacheur de courant, alors le bruit est d'autant plus remarquable. La dérivée du signal de position amène aussi une part de bruit de haute fréquence dans le signal de couple désiré.

On peut voir sur la figure 4.11 un léger retard entre les deux formes d'onde. Ce retard est principalement dû au temps de réponse de la carte électronique de commande de courant du moteur et de la partie magnétique du moteur. Un retard de 20 à 30 ms a été mesuré entre la commande de couple et la lecture de l'estimateur de couple via le courant appliqué aux moteurs. (Voir l'annexe K sur le contrôleur de couple)

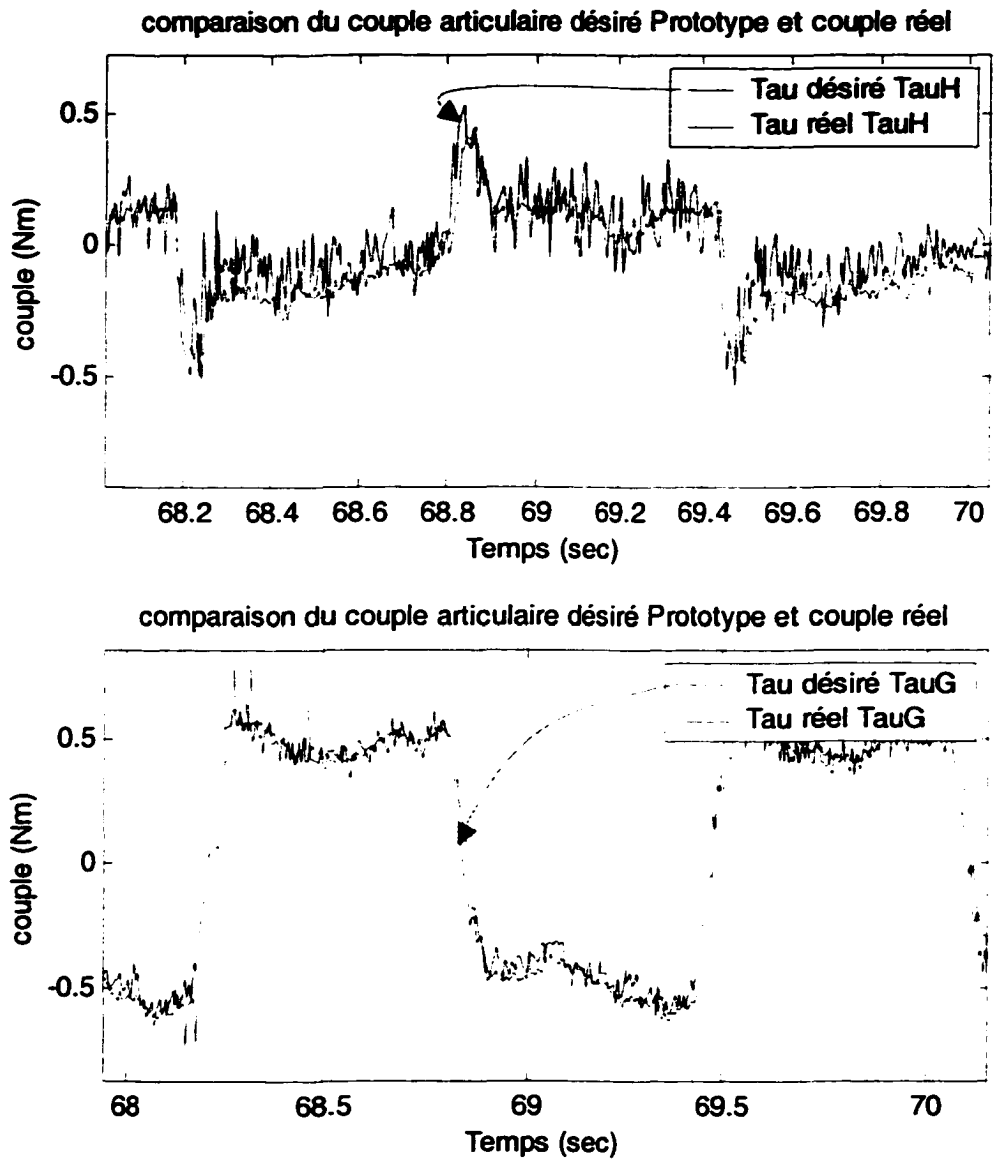


Figure 4.11 Comparaison des couples articulaires demandés et réels

Voici maintenant un tableau qui résume les différentes expériences effectuées avec ce programme. Le critère de comparaison choisi est la moyenne des différences en valeur absolue entre le vecteur de données provenant de la simulation et le vecteur de données provenant du prototype.

Tableau 4.1

Résumé des résultats de l'expérience sur une patte libre

Pour une entrée sinus d'amplitude pi/4 et pi/3 à une fréquence de 5 rad/sec

	Position (deg)	Vitesse (deg/sec)	Couple (Nm)	Couple désiré (Nm)
Hanche	0.3173	22.5672	0.1388	0.0749
Genou	0.6151	32.2747	0.4573	0.0595

Pour signal périodique composé d'une rampe d'amplitude 0.11 rad avec une pente de 0.44 rad/sec suivi d'un plat

	Position (deg)	Vitesse (deg/sec)	Couple (Nm)	Couple désiré (Nm)
Hanche	0.0191	0.6559	0.3688	0.0576
Genou	0.0469	0.8187	0.3015	0.0366

Pour un signal périodique composé d'une entrée rampe d'amplitude 0.24 rad avec une pente de 0.44 rad/sec suivi d'un plat

	Position (deg)	Vitesse (deg/sec)	Couple (Nm)	Couple désiré (Nm)
Hanche	0.0192	1.2444	0.4124	0.0666
Genou	0.0454	1.4227	0.3116	0.0382

Pour une entrée échelon

	Position (deg)	Vitesse (deg/sec)	Couple (Nm)	Couple désiré (Nm)
Hanche	0.0965	2.3293	0.1876	0.0638
Genou	0.2070	4.0223	0.1947	0.0609

La plus grande erreur de couple a été obtenue avec la rampe et la plus grande erreur de vitesse et de position a été obtenue avec la commande la plus exigeante en terme de fréquence de déplacement.

4.6.2 Validation du contrôleur basé sur le modèle du robot

Dans cette deuxième expérience, deux essais ont été effectués : le maintient d'une posture constante et le suivi d'une trajectoire circulaire assignée au corps supporté par les quatre pattes. Le modèle complet d'Haedus a été utilisé pour cette expérience et il est présenté à l'annexe D sous le nom H4P_2D.

4.6.2.1 Objectifs atteints

Cette deuxième étape a permis de valider les aspects suivants :

- méthode utilisant le couple précalculé comme terme anticipatif;
- commande employant la dynamique complète appliquée au robot réel.

4.6.2.2 Méthodologie

Le programme H4P_FinR1 était destiné à cette expérience ainsi qu'à la suivante mais ce dernier demandait une puissance de calcul supérieur à ce que les ordinateurs disponibles au moment de l'expérimentation pouvaient fournir. Pour tout de même valider le contrôleur tel qu'il avait été déterminé, une modification a été apportée sur la commande de couple précalculé définie par l'équation (3.13). La méthode consiste à calculer en temps différé les couples

nécessaires pour suivre la trajectoire assignée. Une fois ces couples enregistrés à partir du modèle dynamique du robot, ils sont introduits dans la loi de commande d'un contrôleur plus simple basé uniquement sur une description cinématique du robot. Cette méthode de couple précalculé modifiée procure deux principaux avantages : elle permet de contrôler le robot directement dans l'espace articulaire et offre la possibilité de calculer les couples préalablement. Ces avantages ont rendu possible l'expérimentation sur le robot réel avec l'équipement disponible. Il existe plusieurs variantes à cette méthode d'approximation, Paden (1988) a démontré l'une de celles-ci, à toutes fins semblable à celle utilisée dans cette expérience.

Le modèle du robot incluant les inerties des actionneurs est donné par la relation suivante :

$$\tau_r = \mathbf{MMJ} \cdot \ddot{\mathbf{q}}_p + \mathbf{F}(\mathbf{q}_p, \dot{\mathbf{q}}_p) + \tau \tag{4.9}$$

où $\mathbf{q_P}$ est le vecteur des coordonnées généralisées des pattes du robot dans l'espace articulaire, **F** est la friction, τ_r est le vecteur des couples appliqués aux articulations du robot par les actionneurs et τ représente le modèle dynamique du robot. Ce vecteur τ représente les couples nécessaires pour faire bouger le robot sans tenir compte des actionneurs. Il peut être exprimé selon les relations (2.40) et (3.13) :

$$\tau = [\mathbf{B}]^{+}[\mathbf{M}\ddot{\chi} + \mathbf{N}(\chi, \dot{\chi})] \tag{4.10}$$

Maintenant, en négligeant les forces de frottement, le modèle décrit par la relation (4.9) devient :

$$\tau_r = \mathbf{MMJ} \cdot \ddot{\mathbf{q}}_p + \tau \tag{4.11}$$

La loi de commande de couple précalculé modifié est ainsi définie comme suit :

$$\boldsymbol{\tau}_r = \mathbf{MMJ} \cdot \mathbf{u} + \boldsymbol{\tau}_d \tag{4.12}$$

où $\boldsymbol{\tau}_d$ est le couple calculé à partir du modèle du robot donné par la relation (4.10) dans lequel les coordonnées généralisées ainsi que leurs dérivées ont été remplacées par la trajectoire désirée. Dans la loi de commande (4.12), le vecteur **u** correspond au contrôleur PID suivant :

$$\mathbf{u} = \ddot{\mathbf{q}}_p{}^d + \mathbf{K}_p e + \mathbf{K}_i \int e + \mathbf{K}_d \dot{e} \tag{4.13}$$

En remplaçant la loi de commande (4.13) dans le modèle (4.12), l'équation d'erreur obtenue est alors :

$$\mathbf{MMJ}(\ddot{\mathbf{q}}_p{}^d + \mathbf{K}_p e + \mathbf{K}_i \int e\, dt + \mathbf{K}_d \dot{e} - \ddot{\mathbf{q}}_p) = \boldsymbol{\tau} - \boldsymbol{\tau}_d \tag{4.14}$$

Finalement, l'erreur entre la loi de commande et le vecteur d'accélération convergeant vers 0, la différence entre les couples nécessaires pour faire bouger le robot sans tenir compte de l'inertie des actionneurs et les couples calculés à priori doit nécessairement tendre vers 0.

C'est donc à partir de cette technique, que les deux prochaines expériences ont été réalisées. Le robot a d'abord pris la position voulue pour ensuite être déposé sur la table. Ensuite, pour le premier essai où le robot devait maintenir une posture précise, le contrôleur de position a été retiré pour ne laisser que le couple calculé à priori. Lors du deuxième essai, où le corps du robot décrivait une trajectoire circulaire, le contrôleur était toujours présent et ce

sont les couples calculés à priori qui ont été retirés pour effectuer la comparaison.

4.6.2.3 Résultats

Les couples calculés à priori utilisés dans le programme de contrôle simplifié du robot sont valides. Il est difficile de mesurer leur importance, mais leur contribution est significative. En effet, il est possible de mettre le robot dans une posture précise et de retirer le contrôleur pour qu'il conserve cette position avec seuls les couples calculés à partir du modèle.

Cependant, pour faire bouger le robot, les couples calculés à priori ne sont pas suffisants. Le contrôleur PID sur la position articulaire est nécessaire pour combattre les forces de friction et les inerties des actionneurs. De bons résultats ont été obtenus avec cette simulation et le mouvement du robot était beaucoup plus fluide et précis en ajoutant les couples calculés à priori. Maintenant, il est logique de se poser la question : « est-ce que le robot pourrait bouger sans utiliser les couples calculés à priori, c'est-à-dire, sans tenir compte de la dynamique du corps et des pattes? ». La réponse est oui, mais les performances sont moindres. Sur le suivi de cercle, les erreurs de positionnement des articulations étaient deux à trois fois plus grandes sans les couples calculés à priori. Le mouvement demandé était relativement lent, en raison des limitations mécaniques du robot, mais il est facile de prévoir que pour une démarche rapide avec un robot plus lourd, l'effet dynamique prendrait encore plus d'importance et ainsi la méthode utilisant la dynamique complète se justifierait d'elle-même par les performances obtenues.

Ce premier graphique présente les positions articulaires que le robot doit suivre pour que son corps décrive un cercle. Les positions ainsi que les positions désirées sont superposées à la figure 4.12. Les courbes de plus grande amplitude sont celles des genoux et les courbes de plus faible amplitude sont celles des hanches. Remarquer à quel point les deux courbes sont superposées, démontrant ainsi avec quelle précision le contrôleur permet de suivre la trajectoire assignée.

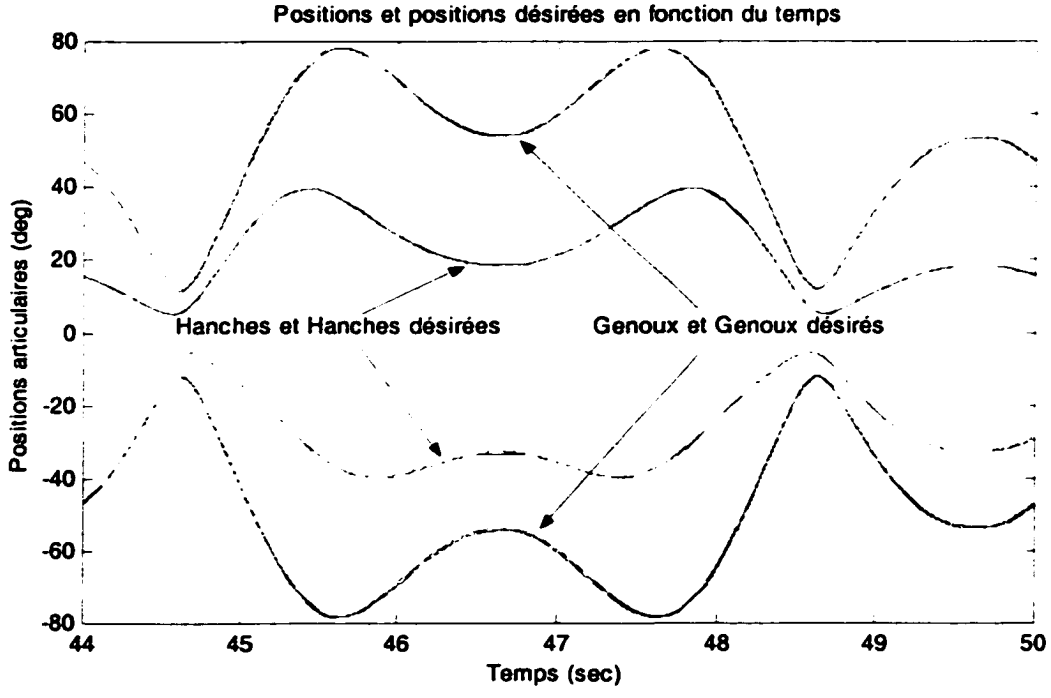


Figure 4.12 Positions désirées et obtenues pour une trajectoire circulaire

Comme les courbes désirées et obtenues sont très près les unes des autres, il est difficile de voir les erreurs. La figure 4.13 présente un graphique de

l'ensemble des erreurs entre les trajectoires désirées et obtenues qui permet d'évaluer la constance et l'amplitude de cet ensemble :

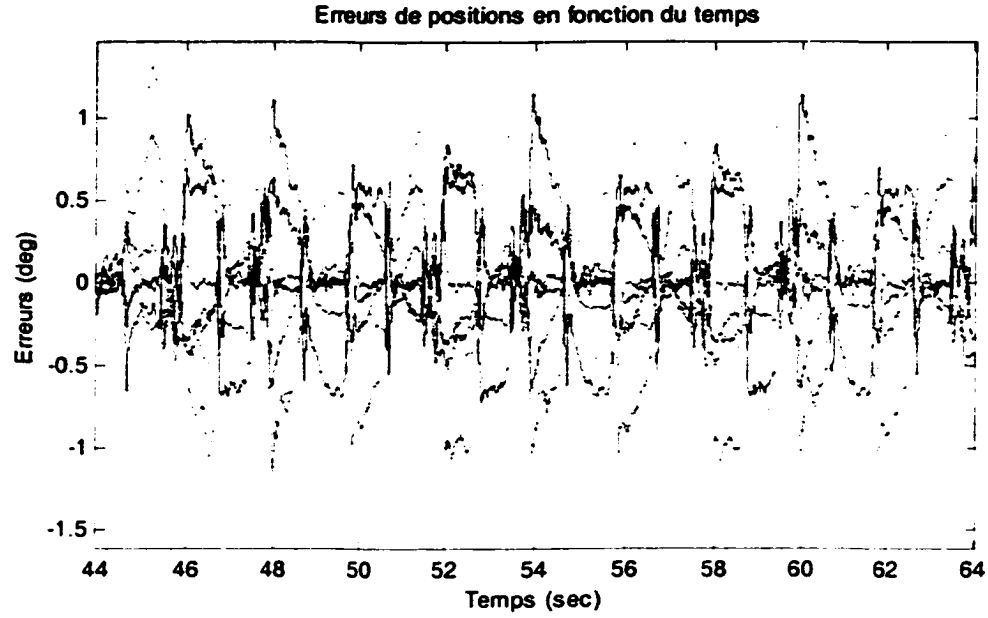


Figure 4.13 Erreur de position avec une trajectoire circulaire

Le comportement de chaque patte est similaire. Voici un exemple d'erreur de position pour la patte 1. Dans tous les cas, l'erreur est beaucoup plus grande pour le genou que pour la hanche. Ceci est probablement dû au couplage entre la hanche et le genou, à la friction qui est plus grande et à l'amplitude du mouvement demandé. En effet, le genou doit compenser pour le mouvement de la hanche et sa mécanique entraîne une poulie de plus que la hanche.

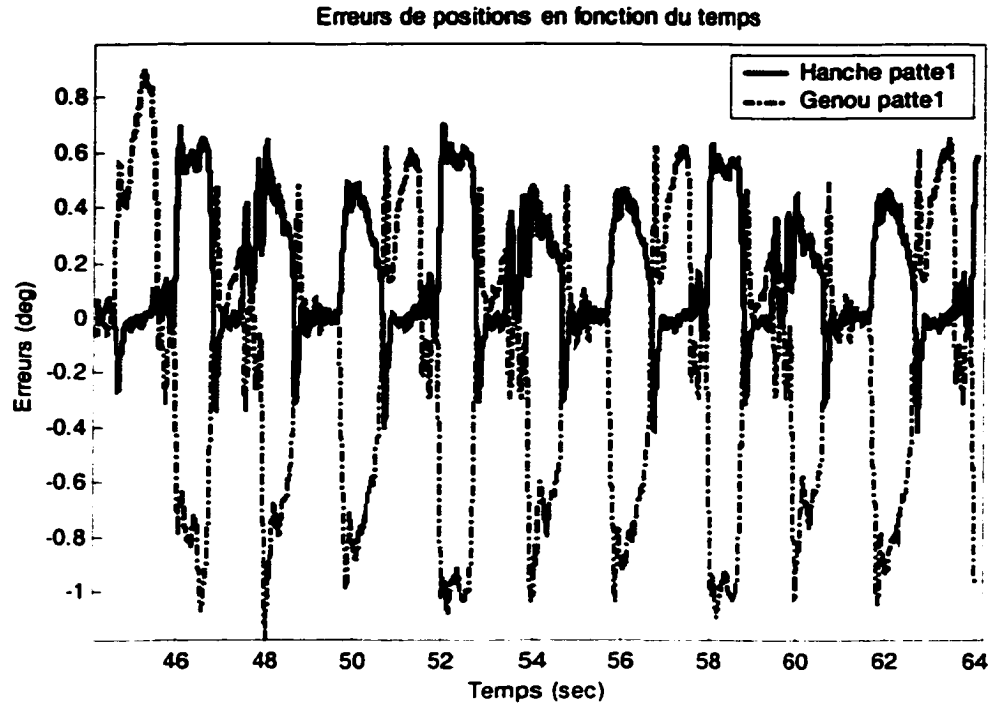


Figure 4.14 Erreur de position pour une patte avec une trajectoire circulaire

En retirant les couples calculés à priori, l'erreur augmente considérablement. Voir les figure 4.14 et 415.

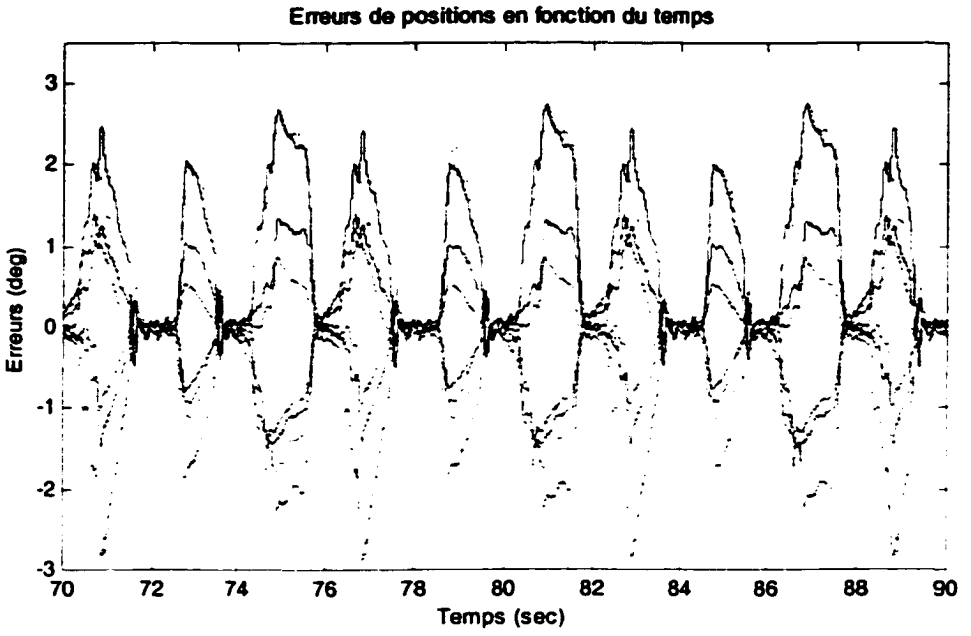


Figure 4.14 Erreur de position avec une trajectoire circulaire sans l'apport des couples calculés à priori

En isolant une patte du système il est plus facile de voir les erreurs tout au long du parcours :

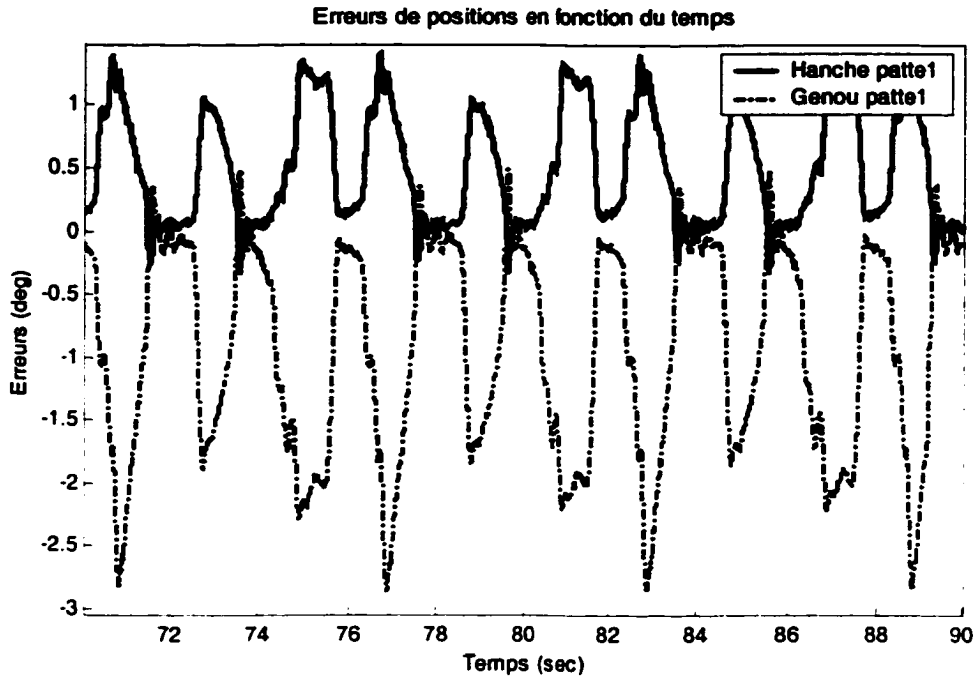


Figure 4.15 Erreur de position pour une patte avec une trajectoire circulaire sans les couples calculés à priori

Enfin, il est important de s'assurer que les moteurs n'atteignent pas leur niveau de saturation. Voici un graphique des couples et couples désirés pour une patte lors du suivi de la trajectoire circulaire. Les couples réels sont les signaux bruités; la hanche est la courbe du haut et le genou, la courbe du bas.

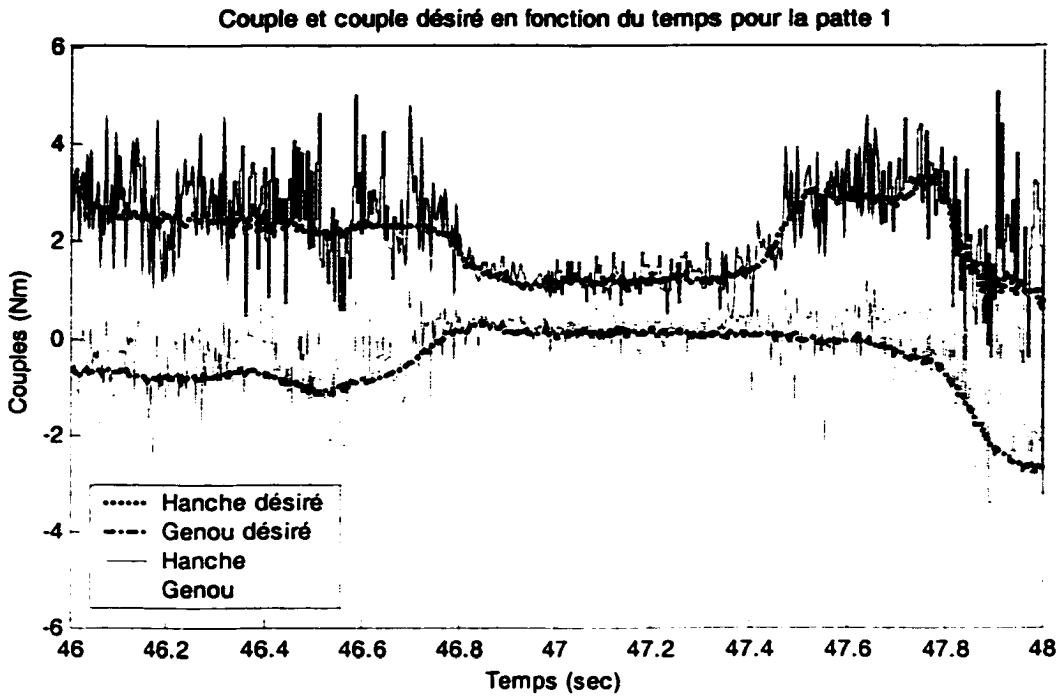


Figure 4.16 Courbe des couples pour une patte avec la trajectoire circulaire

4.6.3 Validation du contrôleur avec la démarche "Amble"

Pour réaliser cette troisième expérience, le robot est initialisé à une position stable et la démarche débute à une vitesse nulle pour accélérer jusqu'à atteindre la vitesse de marche de 0.025 m/s à la cinquième seconde de simulation. Le vecteur de couples précalculés ainsi que le vecteur de positions définissant la trajectoire à suivre ont été générés à l'aide du modèle dynamique selon les même paramètres utilisés par la simulation présentée à la section 3.4.2.

La séquence de marche "Amble" utilisée ressemble à ce qui est illustré à la figure ci-contre. Sur la première photo, la patte 4 est libre et s'apprête à se redéposer en avant de la patte 3 en préparation du prochain pas. Sur la

deuxième photo, la patte 2 se lève et se déposera devant la patte 1. La patte 3 s'apprête à se déposer sur la troisième photo, puis la patte 1 sur la quatrième photo. La cinquième expose la position quatre pattes au sol, qui est la position adoptée entre chaque pas d'une durée variable selon la période de recouvrement. La sixième photo est une vue en perspective du robot en marche.

Figure 4.17 Séquence de marche "Amble" avec posture sur 4 pattes entre chaque pas et une prise de vue en perspective

4.6.3.1 Objectifs atteints

Cette dernière étape a permis de valider les aspects suivants :

- **la démarche "Amble" en temps réel avec Haedus;**
- **l'efficacité du contrôleur pour l'exécution de la démarche;**
- **la concordance entre les résultats expérimentaux et les résultats de simulation.**

4.6.3.2 Méthodologie

Dans la simulation, le robot est contraint dans un plan, ce qui l'empêche de basculer d'un côté ou de l'autre pendant la démarche. Tel que mentionné à la section 3.4, le triangle de support formé par les trois pattes au sol contient à la limite, la projection du centre de masse au sol. Avec les effets dynamiques, la projection de ce centre de masse se déplace et le point sort à l'occasion du triangle de support assurant normalement la stabilité du robot. Ce problème ne faisant pas partie de l'étude présentée dans ce travail, les pattes du robot ont été élargies avec une tige d'acier pour remédier à la situation et faire les expériences. Ceci a eu pour effet de rapprocher son comportement à celui d'un robot contraint dans un plan.

Figure 4.18 Photo d'Haedus avec ses pattes élargies

4.6.3.3 Résultats

Le comportement du robot est excellent. Le corps conserve une vitesse constante et reste en parfait équilibre durant la phase stable de la démarche. En effet, la démarche n'est pas parfaite sur toute sa période, malgré les correctifs apportés au robot. Elle doit être retravaillée pour obtenir une séquence parfaitement stable. Notamment, elle doit tenir compte de la dynamique du robot.

La figure 4.19 illustre les trajectoires désirées au niveau des articulations en superposition avec les trajectoires obtenues. Cela donne une idée de l'amplitude des mouvements et de leur constance, en parallèle avec la constance

et la faible amplitude des erreurs à peine perceptibles le graphique de la figure 4.19.

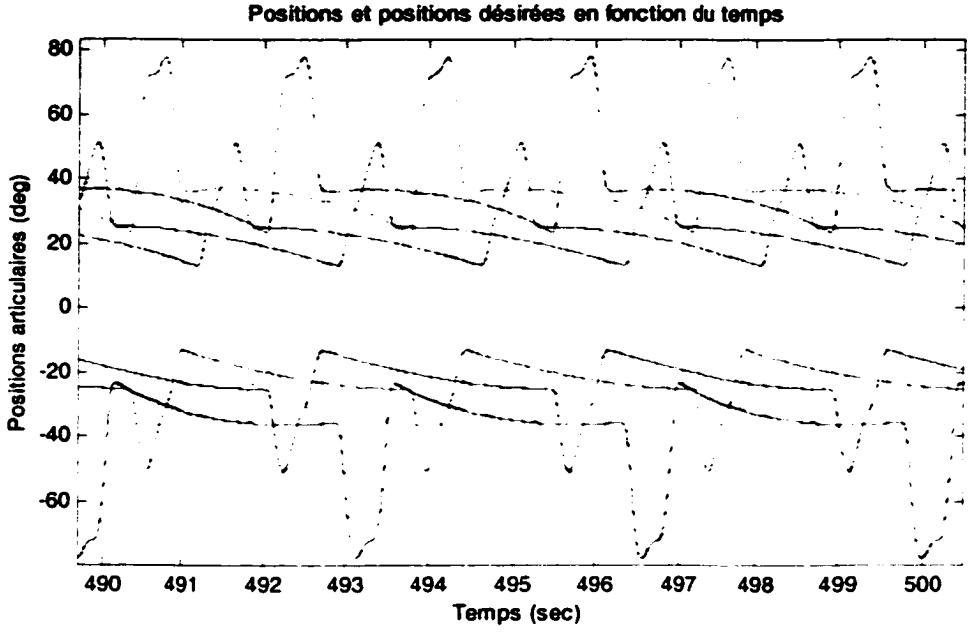


Figure 4.19 Positions désirées et obtenues pour une démarche "Amble"

Évidemment, pour mesurer l'amplitude des erreurs, il est utile d'illustrer la différence entre la trajectoire désirée et la trajectoire obtenue. Les erreurs sont de l'ordre de 1 à 2.6 degrés d'amplitude comparé à un mouvement de 35 à 50 degrés d'amplitude, ce qui donne de 3 à 5 % d'erreur sur la position. Il faut tout de même mentionner que la friction au niveau des articulations est très grande, ce qui entraîne nécessairement une erreur de position plus élevée. À titre indicatif, pour combattre la friction sur certaines articulations, la commande envoyée au moteur doit atteindre près de 20%.

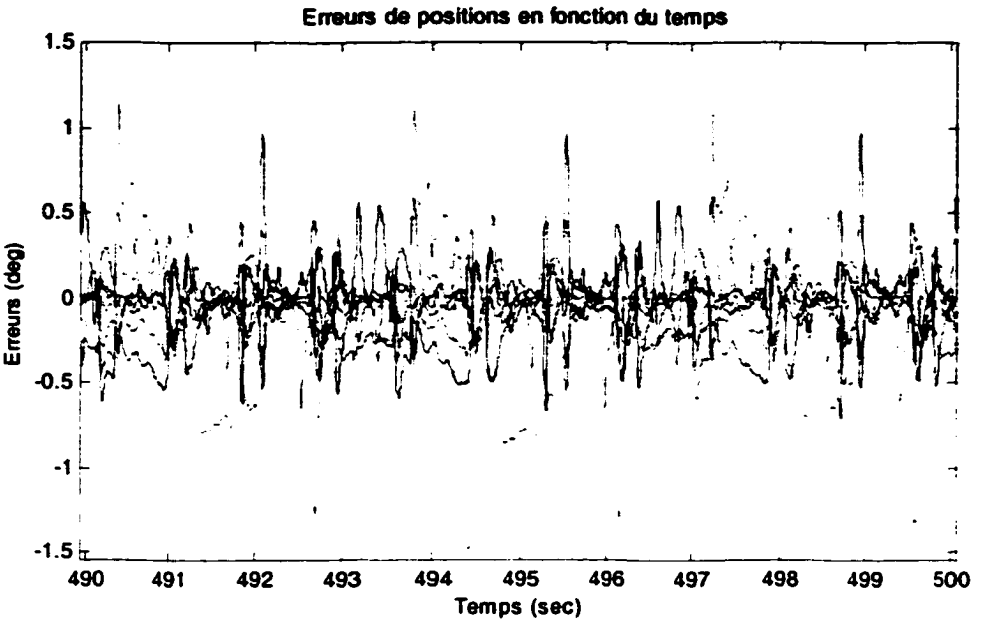


Figure 4.20 Erreurs de positions pour une démarche "Amble"

Les erreurs de positions sur une seule patte ressemblent à ceci :

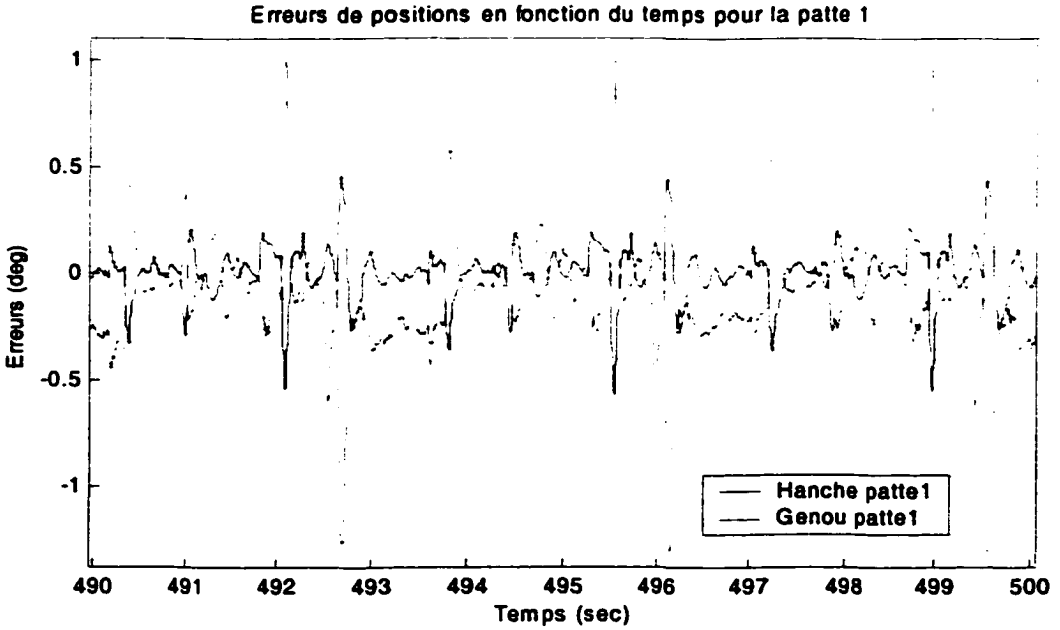


Figure 4.21 Erreur de position pour une patte avec la démarche "Amble"

Encore une fois, pour vérifier si les moteurs n'atteignent pas leur niveau de saturation, il est bon de comparer les couples désirés et les couples réels, par exemple pour la patte 1 :

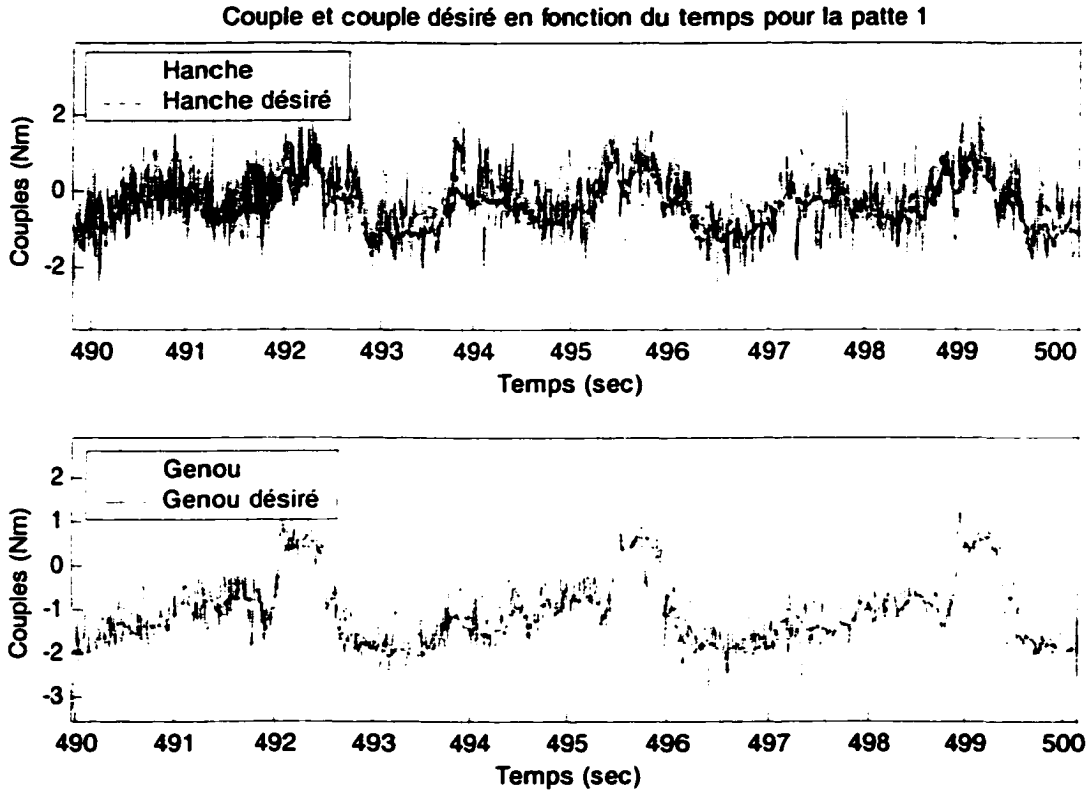


Figure 4.22 Courbe des couples pour une patte avec une démarche "Amble"

4.6.4 Discussion sur l'ensemble des résultats

Les erreurs de positionnement sont relativement faibles et la précision atteinte est supérieure à ce que l'ensemble de la mécanique peut donner en raison de son élasticité. En effet, la lecture de position provient des encodeurs installés sur les moteurs, (voir figure 4.7) alors que la position de l'extrémité des

pattes dans l'espace cartésien peut varier en raison de l'élasticité des membres et des câbles. La plus grande erreur enregistrée au niveau de la position des articulations était de 5%. Sommes toutes, les erreurs de positionnement n'empêcheront pas le robot de suivre la trajectoire assignée ni de perturber son équilibre. Même l'élasticité du système qui présente visuellement plus de variation ne saurait influencer la démarche suffisamment pour diminuer de façon considérable la démarche.

4.7 Conclusion

Ce chapitre présentait toute la partie expérimentale du travail : le matériel utilisé, les logiciels, la méthodologie empruntée pour chaque expérience, accompagnée d'explications sur leur mise en œuvre et pour terminer, les résultats obtenus. Les objectifs ont été atteints et le contrôleur a démontré son efficacité sur le robot réel sans avoir à lui apporter de modifications par rapport au contrôleur utilisé en simulation.

DISCUSSION ET INTERPRÉTATION DES RÉSULTATS

Les résultats obtenus par expérimentation et par simulation sont très encourageants. Les erreurs en moyenne enregistrées sont inférieures à 5% sur les trajectoires assignées. Ce travail de recherche a démontré qu'il était possible modéliser un robot marcheur en utilisant SYMOFROS et de réaliser un contrôleur adéquat basé sur le modèle dynamique. Nous avons en plus vérifier la concordance des résultats de simulation avec les résultats expérimentaux obtenus avec le robot Haedus et le système d'acquisition utilisant RT-Lab. Cependant, une étape reste à accomplir avant de conclure : l'exécution en temps réel.

Le programme original permettant de coordonner la démarche n'a pas pu être exécuté en temps réel en raison de son temps de calcul trop long. En effet, à l'heure actuelle, il reste à répondre à deux questions à ce sujet : est-ce que la méthode est trop exigeante ou est-ce que ce sont les générateurs de codes qui ne sont pas assez performants? Une simplification pourrait être apportée au modèle pour réduire le nombre de calculs : les forces de Coriolis et les forces centrifuges pourraient être négligées. Pour ce qui est du calcul matriciel, nous avons employé des algorithmes profitant des propriétés des matrices pour accroître l'efficacité, alors il y peu d'améliorations à apporter à ce niveau.

En deuxième lieu, une fois que le contrôleur pourra être exécuté en temps réel, il faudrait procéder à l'installation de capteurs pour la détection du contact

au sol. Dans la même veine, ajouter les protections nécessaires pour les limites physiques du robot.

Pour ce qui est de la démarche, elle est à retravailler afin d'obtenir une séquence parfaitement stable. Ce que nous avons observé, c'est qu'avec l'algorithme de génération de trajectoires dont nous disposions, il était impossible d'assurer la stabilité en tout temps. À chaque changement de patte, la projection du centre de masse atteignait la limite du triangle de support. Pour résoudre ce problème et augmenter la marge de stabilité, plusieurs correctifs peuvent être apportés. Notamment, changer l'orientation du corps pour déporter son poids vers le centre du triangle de support. Quoi qu'il en soit, un travail important devra être réalisé pour obtenir une bonne trajectoire stable. Le mouvement des pattes synchronisé avec celui du corps, les inerties et les accélérations doivent être considérées pour obtenir une stabilité et de bonnes performances de marche.

En ce qui concerne la réduction du modèle, cette approche simplifie grandement le problème dans l'ensemble. Elle repose cependant sur le vecteur d'identification des pattes libres. L'état de ce vecteur a été défini à partir de la trajectoire désirée assumant que le robot suit parfaitement la trajectoire. Cela ne pose pas de problème dans le cas de la simulation, mais devrait tout de même être amélioré pour être plus fidèle à la réalité. Pour ce faire, il faudrait deux conditions dans le cas de la simulation: une patte doit être considérée au sol, lorsqu'elle atteint la position où elle entre en contact avec le sol; À l'inverse lorsque la force désirée à appliquer au bout de la patte devient négative, cette patte doit être considérée comme ayant quitté le sol. Pour le robot, deux types de capteurs pourraient être utilisés : un simple capteur de contact bien ajusté ou un capteur de force. Le robot fut privé de ces capteurs pour une question de

coût et de temps. Dans le programme de simulation, ce détail ne posait pas de problème pour faire l'obtention des résultats, du moment que les trajectoires respectaient les conditions nécessaires pour que les pattes ne génèrent pas de forces négatives au niveau du sol.

Pour ce qui est du sol, nous l'avons considéré comme étant parfaitement rigide. Ceci pourrait poser un problème de transfert de quantité de mouvement instantané si la patte entrait lourdement en contact avec le sol. Ce qui n'est pas le cas puisque la vitesse des pattes est réduite à presque nulle au moment de toucher le sol. Cela limite les effets d'impact que pourraient subir les pattes en touchant le sol et amener des erreurs. Lors des premiers essais nous avions utilisé un modèle du sol présent dans la bibliothèque SYMOFROS. Nous avons toutefois remplacé le modèle de contact par un sol rigide pour deux raisons : le robot dérapait même en position neutre après un certain temps et surtout, nous voulions pouvoir mesurer les erreurs de positions et de vitesse du contrôleur en simulation sur la trajectoire assignée sans l'interférence de mouvement naturel entre le sol est les pattes, aussi minime soit-il.

Ceci nous amène donc à parler de l'hypothèse de départ sur la friction au niveau du sol. Elle est considérée comme infinie, comme si les pattes au sol étaient littéralement sous contraintes. Cette considération est importante, puisque c'est de cette façon que le modèle a pu être réduit. Par contre, le contrôleur ne tient pas compte du fait que le robot pourrait éventuellement déraper s'il applique une force trop grande. Il existe plusieurs méthodes qui permettent d'assurer un contact ferme en limitant les efforts de sorte qu'ils restent toujours à l'intérieur du cône de friction. Dans la répartition des efforts effectuée avec l'approche présentée dans ce travail, il existe un ensemble de forces internes qui pourrait être utilisé à cet effet. Par contre, l'implantation

d'une telle méthode, nécessite la connaissance du coefficient de friction entre les pattes et le sol. Cette valeur est difficile à obtenir en général et varie selon la texture et la nature du sol. Il est donc très probable que le robot glisse à un moment inattendu. Tout ce qu'il reste à implanter pour palier au problème, c'est un moyen de détecter qu'une patte dérape. Le modèle n'aurait alors qu'à considérer que la patte a quitté le sol et prendre appuis sur ses autres pattes.

Haedus a permis de valider les hypothèses à l'exception de la pertinence de modéliser la dynamique des pattes. Étant donnée la vitesse avec laquelle il se déplace, la dynamique des pattes devient négligeable. Mais ceci n'est qu'une supposition que nous n'avons pas pu valider expérimentalement et qui ne sera surtout pas valide sur tous les robots. Sur Capra, par exemple, il n'y aura pas de réducteur sur les actionneurs hydrauliques et les pattes seront beaucoup plus lourdes.

Il pourrait être intéressant d'ajouter un terme de commande pour compenser la friction. Cependant, vu les résultats obtenus, les améliorations potentielles ne justifieraient pas nécessairement l'effort et le temps de calcul supplémentaire engendré. Le contrôleur actuel se charge bien de ce phénomène perturbatoire.

CONCLUSION

L'essence de ce travail de recherche était d'étudier et d'expérimenter une approche de contrôle basée sur un modèle dynamique complet appliqué au robot marcheur Haedus. Rappelons que le contrôleur avait pour but de coordonner les efforts des pattes du robot en vue de suivre une trajectoire assignée tout en minimisant les efforts. Les résultats obtenus en simulation et l'expérimentation faite sur Haedus ont permis de valider l'ensemble des hypothèses de départ. De plus, nous avons démontré qu'un modèle de robot marcheur tel que celui d'Haedus peut être obtenu avec SYMOFROS et exécuté sur l'environnement RT-Lab. Un seul aspect reste à valider, à savoir si un tel contrôleur peut être exécuté en temps réel dans un système d'ordinateur d'aujourd'hui.

La méthode développée a l'avantage d'utiliser le même modèle pour toutes les configurations possibles du robot. Un autre avantage est la possibilité d'effectuer la planification de trajectoire directement dans l'espace cartésien. À l'opposée, une importante limitation de l'approche proposée est qu'elle nécessite la position du robot dans l'espace. Or, nous savons qu'à l'heure actuelle aucun capteur ne permet de donner avec précision cette information dans des conditions d'utilisation générale. Une deuxième limitation se situe au niveau du nombre de pattes qui peuvent quitter le sol en même temps. Actuellement, le minimum de patte au sol est au nombre de deux puisque c'est

le minimum requis pour manipuler le corps. Cette limitation pourrait être éliminée en retirant temporairement le corps de la liste des objets à commander ce qui permettrait au robot de sauter par exemple et de quitter le sol complètement.

RECOMMANDATIONS ET TRAVAUX À VENIR

En mettant en perspective les validations complétées et la revue de littérature effectuée, nous sommes en mesure de recommander cette approche de contrôle en suggérant quelques modifications. D'abord, en considérant le mode de commande nommé « téléopération », il faut trouver une astuce pour que le suivi de trajectoire ne dépende pas directement de la position du robot dans l'espace, mais plutôt de sa vitesse de déplacement comme un jouet téléguidé par exemple. Ensuite il faut implanter un système de diagnostique et de vérification des conditions pour assurer le respect des limites du contrôleur comme les positions physiquement inaccessibles et le dérapage des pattes au niveau du sol.

Rappelons que ce travail devait être une étape préliminaire à la réalisation du contrôleur pour Capra. Étant donné que la méthode et le programme ont été développés de façon générique, il serait relativement facile de reprendre le modèle d'Haedus et de l'adapter pour Capra en ajoutant un degré de liberté par patte ainsi que le cou (Cervix). Une fonction de détection de l'état des pattes, à savoir si elles sont en appuis ferme au sol ou non, doit être implantée absolument. Pour garder un bon équilibre, des capteurs d'orientation doivent être installés dans le corps du robot. La fonction de reconnaissance de la position basée sur les pas effectués « dead reckonning » doit aussi être implantée.

Maintenant, devrait-on utiliser SYMOFROS et RT-Lab pour la réalisation du prochain contrôleur? C'est une très bonne question à laquelle il nous est difficile de répondre vu la quantité de travail qui fut nécessaire pour utiliser ces logiciels de haut niveau correctement. Certes, ce sont de très bons outils et chacun offre des avantages alléchants à l'utilisateur avisé. Il serait naïf cependant de s'imaginer que de tels outils s'utilisent avec une parfaite transparence lors de la réalisation de programmes complexes comme ceux requis pour un robot marcheur. Une leçon certaine à retenir, c'est d'utiliser le bon matériel avec les bons logiciels. Évitez les combinaisons incertaines et les incompatibilités sources de cauchemars.

Finalement, la couche de contrôle supérieur effectuant la planification de trajectoire avec comportements et réflexes devrait également être développée. Elle permettrait de pouvoir marcher sur un terrain accidenté et possiblement expérimenter diverses démarches comme le trot ou des variantes de celle-ci. La vitesse de marche et la demande énergétique pourrait être optimisée grâce à une meilleure planification de la forme de la trajectoire empruntée par la patte libre. Cette trajectoire doit être réexprimée en temps réel suivant la forme du terrain et la dynamique du robot pour assurer la stabilité de ce dernier et minimiser les mouvements pour conserver une consommation d'énergie minimum. Cette troisième étape demande un travail colossal et pourrait sans aucun doute faire l'objet d'un travail de maîtrise ou de doctorat.

RÉFÉRENCES BIBLIOGRAPHIQUES

Ahmadi, M., Buehler, M. (1999). The ARL Monopod II Running Robot: Control and Energetics. IEEE International Conference on Robotics and Automation, May, pp. 1689-1694.

Alexander, R. (1990). Three Uses for Springs in Legged Locomotion. The International Journal of Robotics Research, volume 9(2), April, MIT, pp. 53-61.

Bai, S. et al. (1999). Quadruped Free Gait Generation Base on the Primary/Secondary Gait. Robotica, volume 17, Cambridge University Press, January, pp. 405-412.

Beer, R.D. et al. (1991). An artificial Insect, American Scientist, volume 79, pp. 444-452.

Berns, K., (1994). Adaptive, Neural Control Architecture for the Walking Machine LAURON. International Conference On Advanced Robotics Systems and the Real World, IROS '94. IEEE/RSJ/GI, volume 2, pp. 1172 – 1177.

Berns, K., et al. (1999). Mechanical construction and computer architecture of the four-legged walking machine BISAM. Transactions on Mechatronics, IEEE/ASME, volume 41, pp. 32 –38, mars.

Brooks, R.A., (1991). Intelligence without representation. Artificial Intelligence, volume 47, pp. 139-159.

Bruneau, O. (1998). Dynamic Walk Simulation of Various Bipeds via Ankle Trajectory. International conference on Intelligent Robots and Systems, Victoria, B.C., October, pp. 58-63.

Buehler, M. et al, (1999). Stable Open Loop Walking in Quadruped Robots with Stick Legs. IEEE Internatinal Conference on Robotics and Automation, May, pp. 2348-2353.

Buehler, M. et al. (1998). SCOUT: A simple quadruped that walks, climbs, and runs. IEEE International Conference on Robotics and Automation, volume 2, May, pp. 1707-1712.

Chevallereau, C., Formal'sky, A., Perrin, B. (1997) Control of a walking robot with feet following a reference trajectory derived from ballistic motion, IEEE International Conference on Robotics and Automation, volume 2, pp. 1094 -1099.

Cordes, S. et al. (1997). On The Design of a Four-Legged Walking Machine. ICAR '97, IEEE pp. 65-70.

Craig, J. J., (1989). Introduction to Robotics (2e éd.). New York: Addison-Wesley.

De Lafontaine, J. Kassing, D. (1996). Technologies and concepts of lunar surface exploration. Acta Astronautica, volume 38(2), pp. 125-129.

Deblois, S. Lambert, G., Lessard, P., Lupien, S., Sankar, T. (2000)"Design Of Capra - A Quadruped Robot With Improved Agility", may International Symposium on robotics, May, Montréal, pp. 392-397.

Dickinson, M. et al., (2000). How Animal Move : An Intergrative View. Science Magazine, volume 288, 7 april, pp. 100-106.

Doerschuk I. P. (1996). Intelligent Ballistic control of a Jointed Leg. IEEE International Joint Symposia on Intelligence and Systems (IJSIS).

Dunn, E. et al. (1996). Foot Placement and Velocity Control in Smooth Bipedal Walking. IEEE, pp. 578-583.

Formal'sky A., Chevallereau C., Perrin, B. (2000) "On Ballistic Walking Locomotion of a Quadruped. The International Journal of Robotics research, volume 19 (8), August, pp. 743 (NOT USED YET)

Furosho, J. et al. (1995). Realisation of Bounce Gait in a Quadruped Robot with Aritcular -Joint Type Legs. International Conference on Robotics and Automation, IEEE, pp. 697-702.

Gao, X.C. (1990). Stiffness Matrix Method for Foot Force Distribution of Walking Vehicles. Mechanical Engineering Department of university of Illinois at Chicago, pp. 1470-1475.

Hirai, K, et al. (1998). The Development of Honda Humanoid Robot. International Conference on Robotics & Automation, IEEE, may, pp. 1321-1326.

Hirose, S. (1984). A Study of Design and Control of a Quadruped Walking Vehicle. The International Journal of Robotics research, volume 3(2), Summer, pp. 113-133.

Hirose, S. Kato, K. (2000). Study on Quadruped Walking Robot in Tokyo Institute of Technology – Past, Present and Future -. International Conference on Robotics & Automation, San Fransisco, CA, april, pp. 414-419.

Hong. Y-S. et al. (1999). The design an control of a jointed-leg type of a quadruped robot for locomotion on irregular ground. Robotica, Cambridge University press, volume 17, pp. 383-389.

Hugel, V., Blazevic, P. (1999). Towards efficient implementation of quadruped gaits with duty factor of 0.75. International Conference on Robotics & Automation, Detroit, Michigan, May, pp. 2360-2365.

Inagaki, K. Kobayashi, H. (1993). A Gait Transition for Quadruped Walking Machine. IEEE/RSJ International Conference on Intelligent Robots and Systems, Yokohama, Japan, july, pp. 525-531.

Juang, J.-G. (1998). Bipedal Trajectory Control Based on Neurofuzzy Networks. International Conference on Control Applications, IEEE, Triest, Italy, September, pp. 802-806.

Kang, D. et al, (1997). A Study on an Adaptive Gait for a Quadruped Walking Robot under External Forces. International Conference on Robotics an Automation, Albuquerque, New Mexico, april, pp. 2777-2782.

Khalil, W. Dombre, E. (1999). Modélisation, identification et commande des robots (2e éd.). Hermes Science, Paris.

Kimura, H. et al. (1998). Dynamic walking and Running of the Quadruped Using Neural Oscillator. International Conference on Intelligent Robots and Systems, Victoria, B.C., October, pp 50-57.

Kimura, H., Fukuoka, Y. (2000). Adaptive Dynamic Walking of the Quadruped on Irregular Terrain – autonomous adaptation using neural system model, IEEE Conference on Robotics and Automation, San Fransisco, pp. 9.

Kumar V.R. Waldron, K.J. (1990). A review of Research On Walking Vehicles. pp. 243-265.

Lehtinen, H. (1994). Force based motion control of a walking machine. Ph.D. Thesis, Technical research Center of Finland, VTT publications 179, 150 p.

Lehtinen, H. (1996). Force control for walking on soft terrains. Robotica, volume 14, Cambridge University Press, pp. 165-172.

Lewis, M.A. (1992). A Genetic programming approach to the construction of a neural network for control of a walking robot. International Conference on Robotics and Automation, IEEE, volume 3, pp. 2618 -2623.

Lin, B-S. et al. (1993). Dynamic Modeling, Stability and Energy efficiency of a Quadrupedal Walking Machine. Mechanical Engineering Department of university of Illinois at Chicago, pp. 367-373.

Lupien, S. (2001). Développement d'un logiciel d'aide à la concepion pour les robot quadrupèdes, École de Technologie supérieure, 123 p.

Lupien, S. Lessard, P., Demers, J-C., Lambert, G., Bigras, P., Sankar, S. (2001) "Versatile Robot Platform For Hazardous Environment And Automation Applications", Proceedings of the 2001 ASME Design Engineering Technical Conferences, Pittsburgh, PA, Sept. 2001.

M'sirdi, N.K. et al. (1998). Methodology based on CLC for Control of fast Legged Robots. International Conference on Intelligent Robots and Systems. Victoria, B.C., October pp. 71-76.

Mandzel , P. (2000). Robot Sapiens, MIT Press, Boston.

Margaria, R. (1976). Biomechanics and energetics of muscular exercise. Clarendon press, Oxford pp. 65-146.

Matsuoka, K. (1987). Mechanisms of frequency and pattern control in the neural rhytm generators. Biol. Cybern., volume 56, pp. 345-353.

Matthies, L. et al. (1995). Mars Microrover Navigation: Performance Evaluation and Enhancement. IEEE, pp. 433-440.

McClamroch, H. N., Wang, D., (1988), Feedback Stabilisation and Tracking of Constrained Robots. , IEEE Transactions on Automatic Control, volume 33(5), p.419-426.

McMahon, T. (1975). Using body size to understand the structural design of animals: quadrupedal locomotion. Journal of Applied Physiology, volume 39(4), October, pp 619-627.

Menzel, P., D'aluisio, F. (2000). « Evolution of a new species Robo sapiens ». The MIT press, Cambridge, Massachusetts, 240 p.

Miura H. et al. (1984). Dynamic Walk of a Biped. The International Journal of Robotics Research, volume 3(2), Summer, MIT, pp. 60-74.

Miyashita, T. et al. (1998). Hybrid structure of Reflective Gait Control and Visual Servoing for Walking. International Conference on Intelligent robots and Systmes Victoria, B. C., October, pp. 229-234.

Mosher, R. S. (1968). Test and evaluation of a versatile walking truck, in proc. Off-Road Mobility Research Symposium, pp. 359-379.

Murray, M. R., Li Z., Sastry, S. S., (1994), A Mathematical Introduction to Robotic Manipulation, New York : CRC Press.

Opal-RT, (2001). RT-LAB User's Manual.

Paden, B. Panja, R. (1988). Globally Asymptotically Stable « PD + » Controller for Robot Manipulators, International Journal of Control, volume 47(6), pp. 1697-1712.

Papadopoulos, D., Buehler, M. (2000), Stable Running in a Quadruped Robot with Compliant Legs. IEEE International Conference on Robotics and Automation, San Francisco, 6 p.

Pearson, K. (1991). The Control of Walking. pp. 82-86.

Piedboeuf, J.-C. (1998). Recursive modelling of flexible manipulators. The journal of Astronautical Sciences, volume 46, January-March.

Pratt J.E. (1998). Exploiting Natural Dynamics in the Control of a Planar bipedal Walking Robot. 36th Conference on Communication, Control and computing, Monticello, Illinois, September, pp. 10.

Raibert, M. (1984). Hopping in Legged Systems – Modeling and Simulation for the Two-Dimensional One-Legged Case. IEEE Transaction on systems, volume SMC-14(3), May/June, pp. 451-463.

Raibert, M.H., et al. (1986). Running on four legs as though they were one. IEEE Journal of robotics and Automation, volume RA-2, Avril, pp. 70-82.

Snaith, M., O. Holland, (1991). Quadrupedal Walking Using Trained and Untrained Neural Models. IEEE, volume 2, pp. 715-720.

Song, S-M, Waldron, J.K. (1988). Machines That Walk : The Adaptive Suspension Vehicle. MIT Press, 327 p.

Spong, W., M. & Vidyasagar, M. Robot, (1989), Robot Dynamics and Control, New York : John Wiley & Sons.

Stanley, A. et al. (1992). Object Impedance Control for Cooperative Manipulation : Theory and Experimental Results, IEEE Transactions on robotics and automation, volume 8 (3), Juin, pp. 383-394.

Takanishi, A. et al. (1985). The realization of dynamic walking by the biped walking robot WL-10RD. ICAR 85, Department of Mechanical Engineering, Waseda University, Tokyo, Japan, pp. 459-466.

Tatsuo, A et al. (1995). Integrated Arm and Leg Machanism and Its Kinematic Analysis, IEEE Conference on Robotics and Automation, pp. 944-999.

Veloso, M. et al. (1998). Playing Soccer with Legged Robots. Intenational Conference on Intelligent Robots and Systems, Victoria, B.C., October, pp. 437-442.

Wang, L.T. & Chen, C.C. (1991) A Combined Optimization Method for Solving the Inverse Kinematics Problem of Mechanical Manipulators, IEEE Transactions on Robotics and Automation, volume 7(4), pp. 489-499.

Xu, X-R. et al. (1998). A New approach for Modeling and Computation of Dynamics of Robots Containing Closed Chains. International Conference on Intelligent Robots and Systems, Victoria, B.C. October, pp.618-623.

Yoneda, K. et al. (1996). Intermittent Trot Gait of a Quadruped Walking Machine Dynamic Stability Control of an Omnidirectional Walk. International Conference on Robotics and Automation, Minneapolis, Minnesota, april, pp. 3002-3006.

Zhou, D, (1999). A Friction Constraint Method for the Force Distribution of Quadruped Robots. Proceedings of the IEEE/ASME, International Conference on Advanced Intelligent Mechatronics, September 19-23, Atlanta, USA, pp. 866-871.

ANNEXE A

Liste des fichiers et répertoires

LISTE DES FICHIERS ET RÉPERTOIRES

Le répertoire SymoOpalDev (Symofros et Opal en Développement) renferme les fichiers et sous-répertoires suivants :

_Archives **anciens modèles**

CapraPlan Modèle de Capra dans un plan avec les démarches de Simon Lupien

HaedusPlan Modèle de Haedus dans un plan avec les démarches de Simon Lupien

HaedusFinal

Derniers modèles pour contrôler le robot et la simulation sur RT-Lab comportant les modèles :

H4P_Fin1.mdl Modèle complet utilisé pour la simulation prêt à fonctionner en temps réel sur RT-Lab

H4P_FinR1.mdl Contrôleur d'Haedus avec entrées sortie et librairie du modèle sans la simulation

HaedusOp1

Modèle précédant faisant le passage entre la simulation et la compilation Opal

Sensoray

Modèle de base des deux cartes d'acquisition entrées/sorties complet

SymoOpalT1

Modèle d'une patte d'Haedus sur opal avec comparaison du vrai robot

SymoOpalT4

Modèle de 4 pattes et robot

SymoOpalT4-Proto-Sim

Modèle 4 pattes sans le corps, contrôle du robot et simulation des pattes en temps-réel :

SOT4.mdl

Structure de répertoires essentiels à la compilation et à l'exécution :

LIB contient les fonctions, les librairies de modèles et les animations

anim fonction d'affichage pour la console animation "alumette"

function fonction c à compiler et fonction mex utilisée par la console

lambda2_c calcul les forces de contraintes du sol (sim seul)

MexFun fonction mex pour lambda2 et tau

tau_c calcul les couples par pseudo-inverse

traj trajectoire états initiaux

traj_c fonction c pour générer les trajectoires en temps réel

model modèle symofros sur lesquels les simulations reposes

H4P Haedus sur 4 pattes

P4 4 pattes rattachées ensemble (sans le corps)

walk Démarches enregistrées sur QNX

ANNEXE B

Logiciels utilisés

LOGICIELS UTILISÉS

Voici une liste complète des logiciels nécessaires à la mise en œuvre du système temps réel :

Console d'opération

Système d'exploitation :

Windows 2000

Environnement de travail :

MatLab 5.3.0

Simulink 3.0 R11 avec RTW et DSP blockset installé

RT-Lab 5.2.3

Autres logiciels de génération de code :

Maple 5 release 5

SYMOFROS 4.1.11a

Autres librairies "maison":

S-Fonction de démarche Amble réalisé par Alexandre Doin

S-Fonction Tau1.c Lambda2.c Matrix.c par Patrick Frenette

Trajsim générant des trajectoires Amble par Simon Lupien

Nœud de calcul

Système d'exploitation:

QNX 4.25

Autres logiciels et librairies à installer :

TCP IP

Watcom C++

Librairies SYMOFROS 4.1.11a

Librairies RT-Lab 5.2.3

ANNEXE C

Configurations RT-Lab et RTW

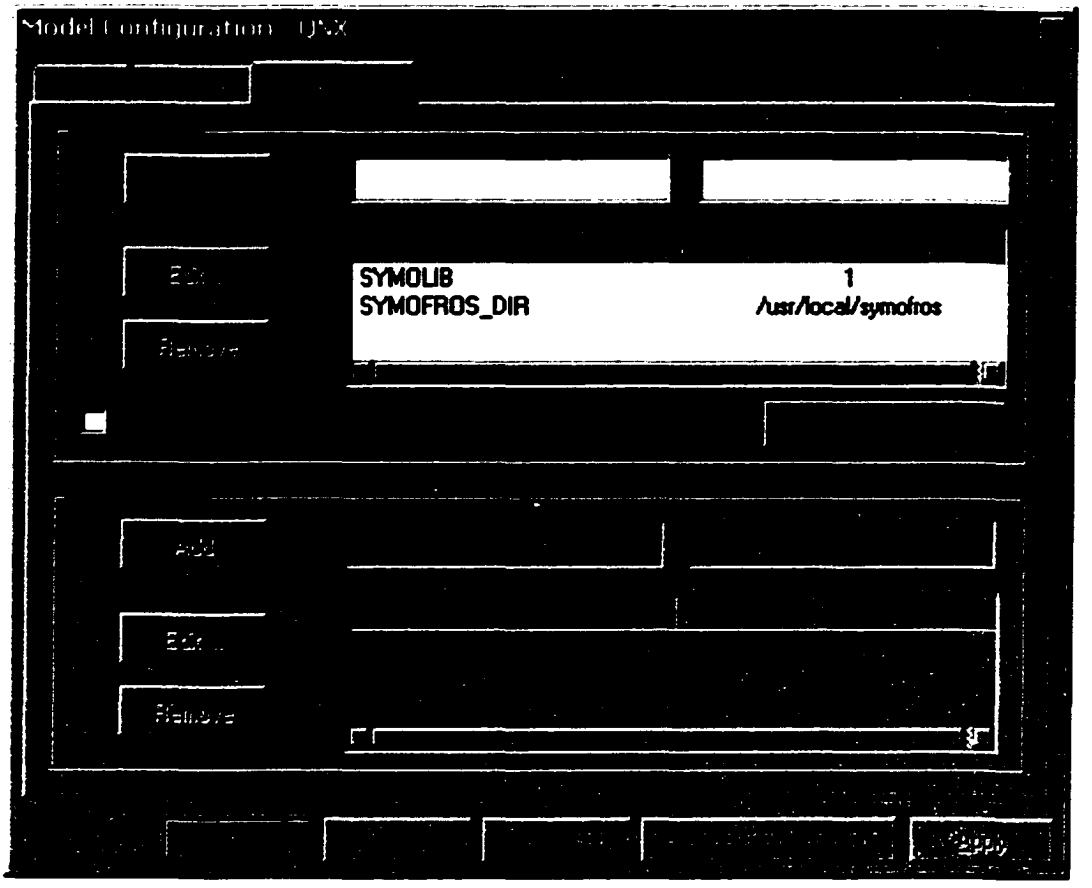
CONFIGURATION RT-LAB ET RTW

Voici les configurations utilisées pour le modèle H4P_Fin1 et H4P_FinR1. Ces paramètres doivent être configuré ainsi dans la console de RT-Lab.

User Var :

SYMOLIB = 1

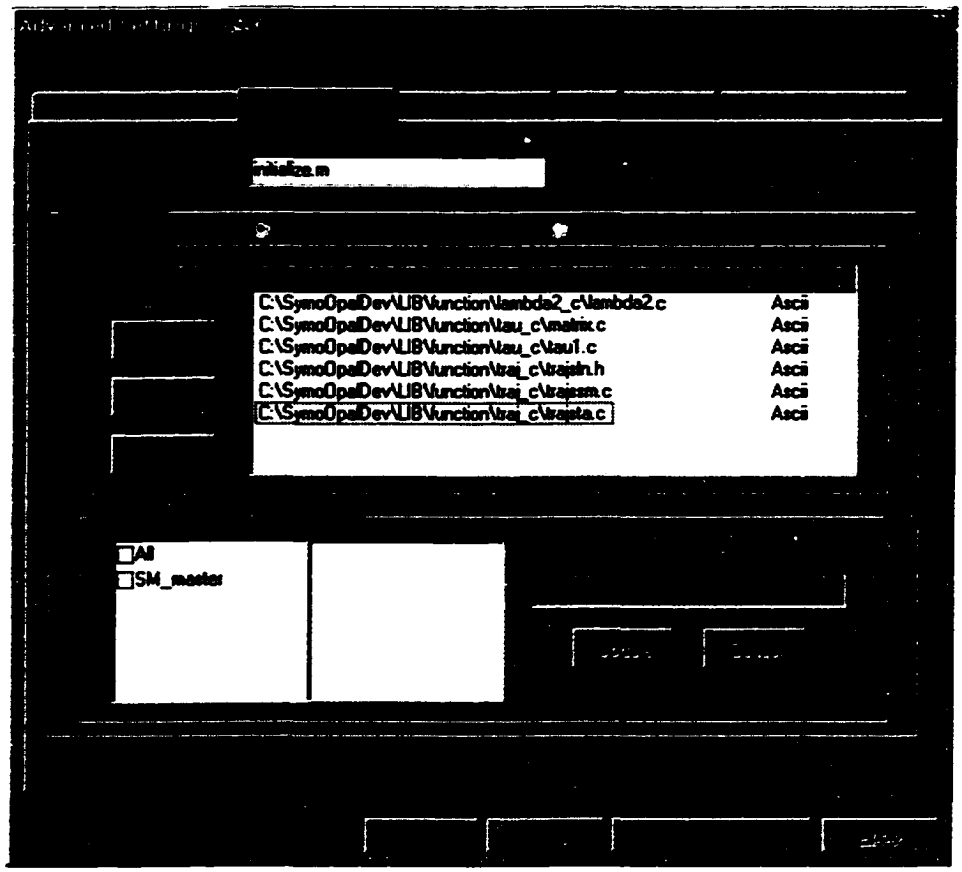
SYMOFROS_DIR = /usr/local/symofros



Advanced :

Prebuild command: initialize.m

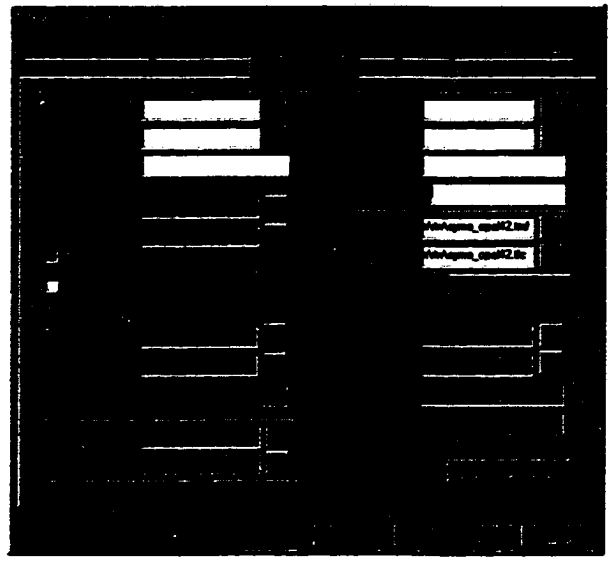
Add Files: matrix.c; tau1.c; trajln.h; trajssm.c; trajsta.c



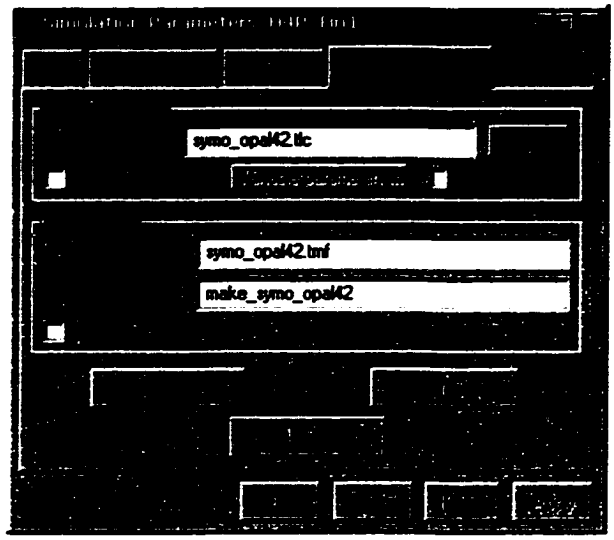
Compilation option:

RTW: c:\matlabr11\toolbox\rtw\symoopal42.tmf

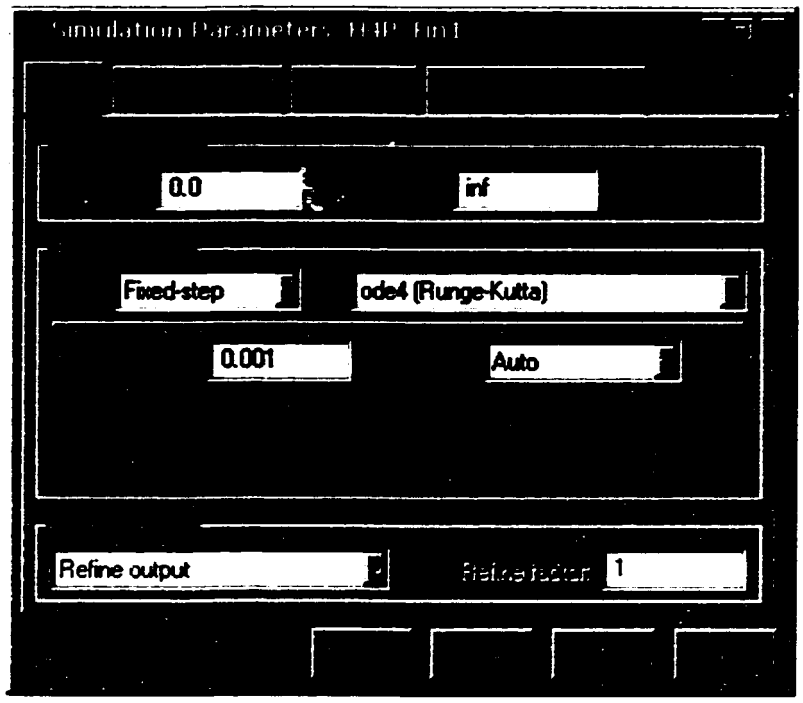
c:\matlabr11\toolbox\rtw\symoopal42.tlc



Pour le programme Simulink, les paramètres de simulation dans la section *Real-Time Workshop* doivent être configurés ainsi :



Les paramètres utilisés de la section *solver* sont :

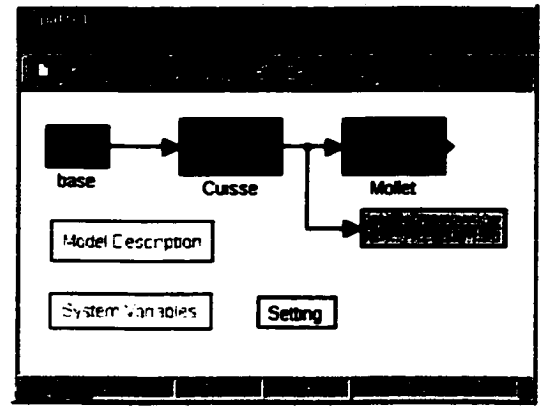


ANNEXE D

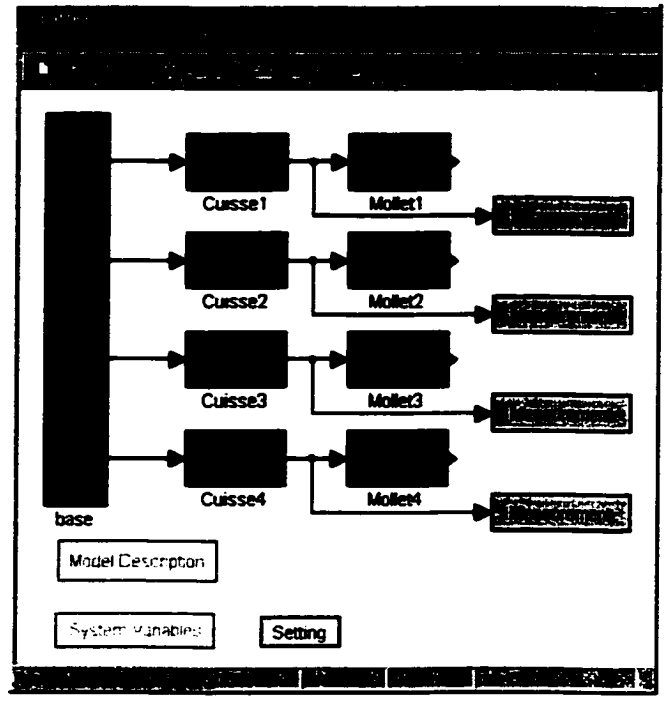
Modèle SYMOFROS associés aux principaux programmes

MODÈLES SYMOFROS ASSOCIÉS AUX PRINCIPAUX PROGRAMMES

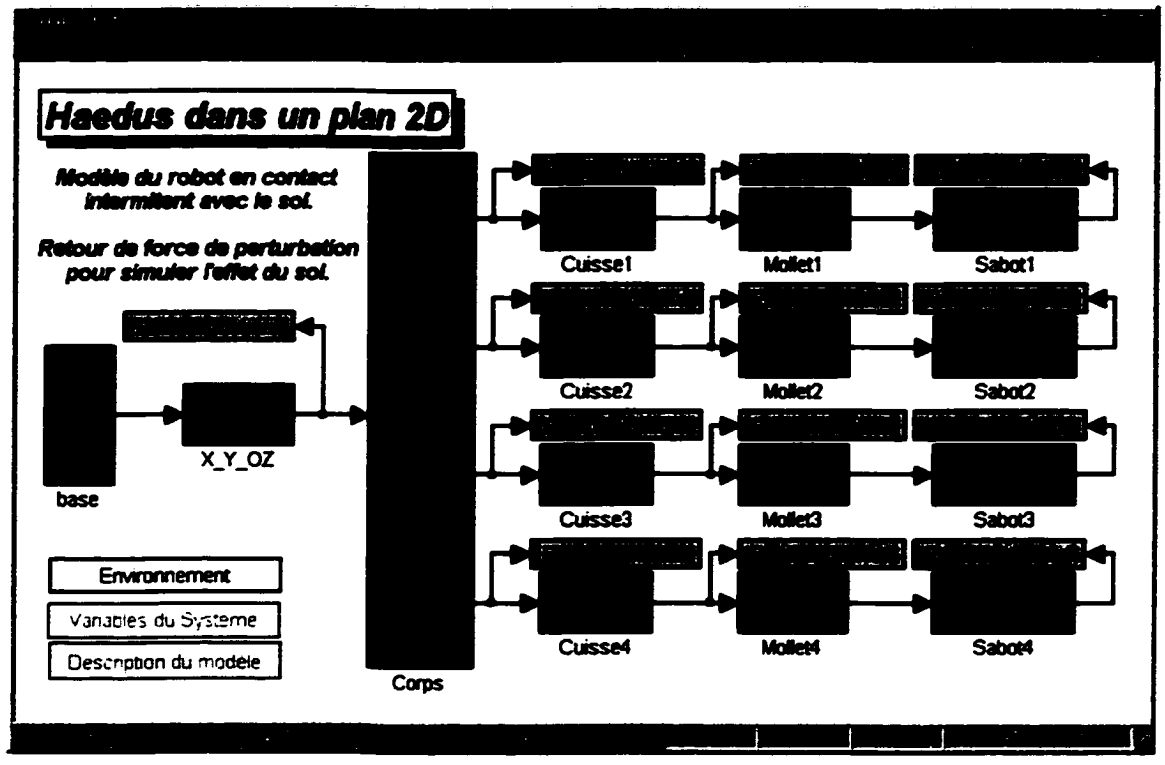
PATTE1 MODÈLE D'UNE PATTE DU ROBOT HAEDUS :



PATTE4 MODÈLE 4 PATTES DU ROBOT HAEDUS :



H4P_2D MODÈLE COMPLET DU ROBOT HAEDUS :

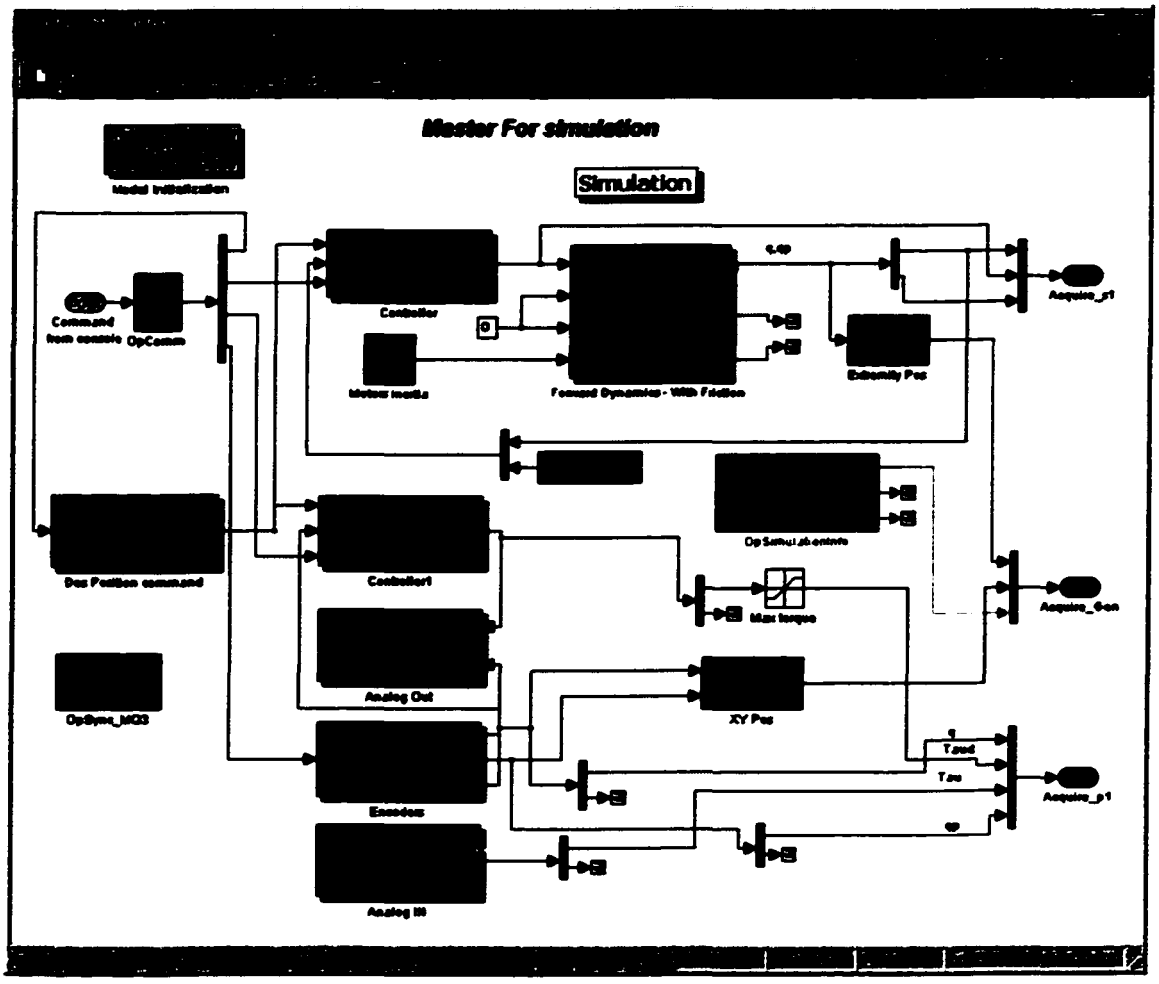


ANNEXE E

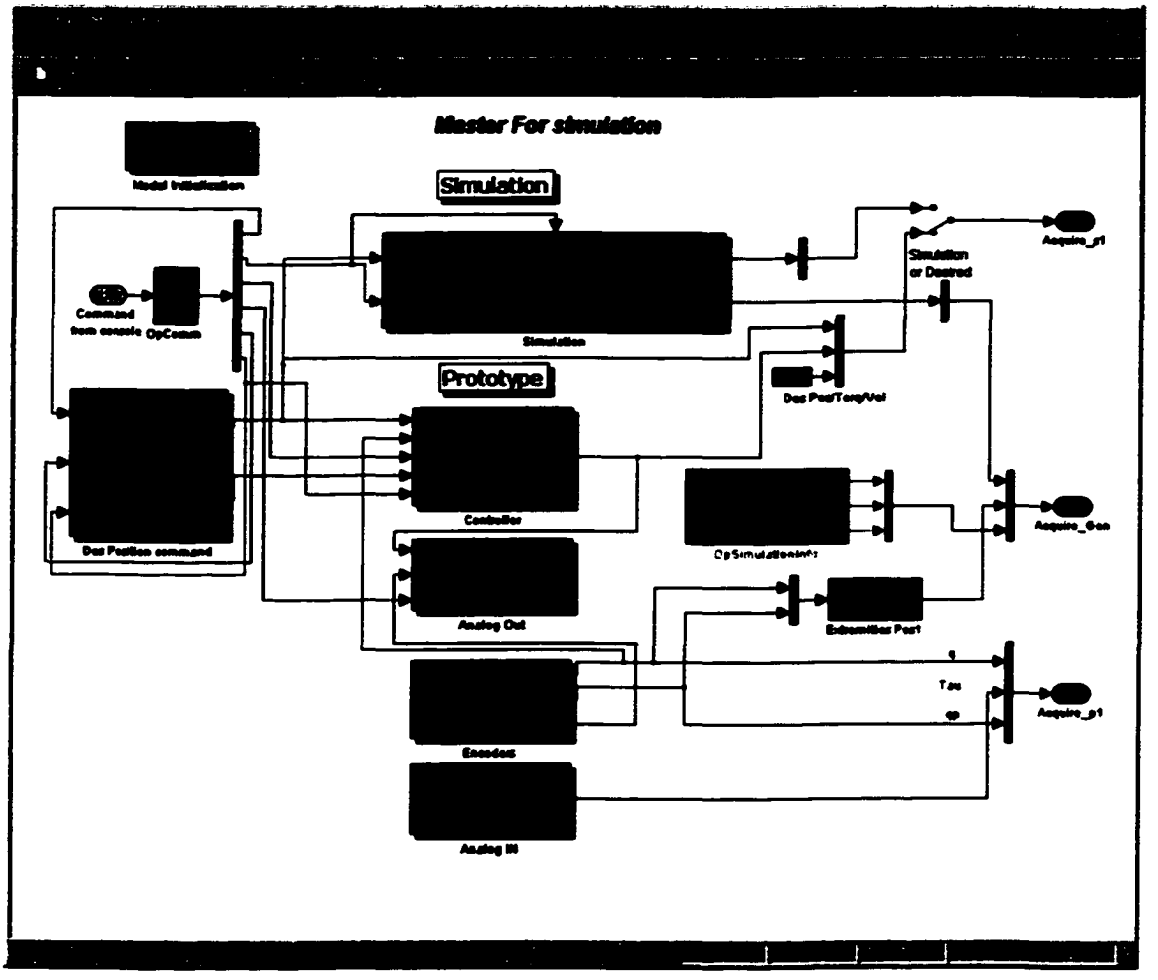
Appercu des programmes présentés dans ce travail

APPERCU DES PROGRAMMES PRÉSENTÉS DANS CE TRAVAIL

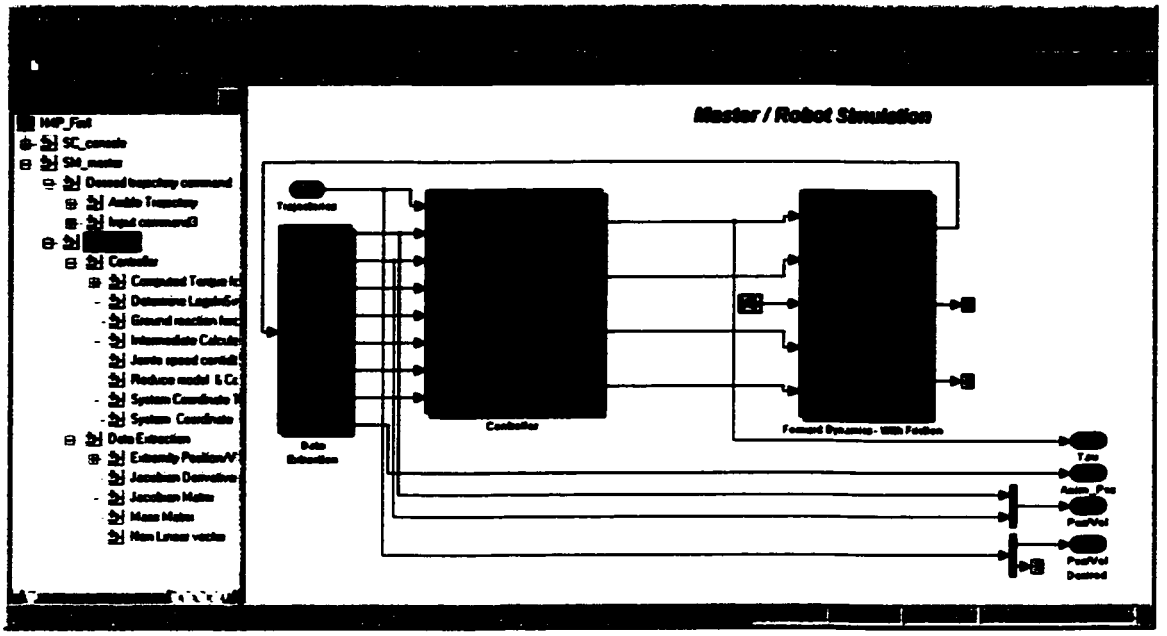
SOT1/Master :

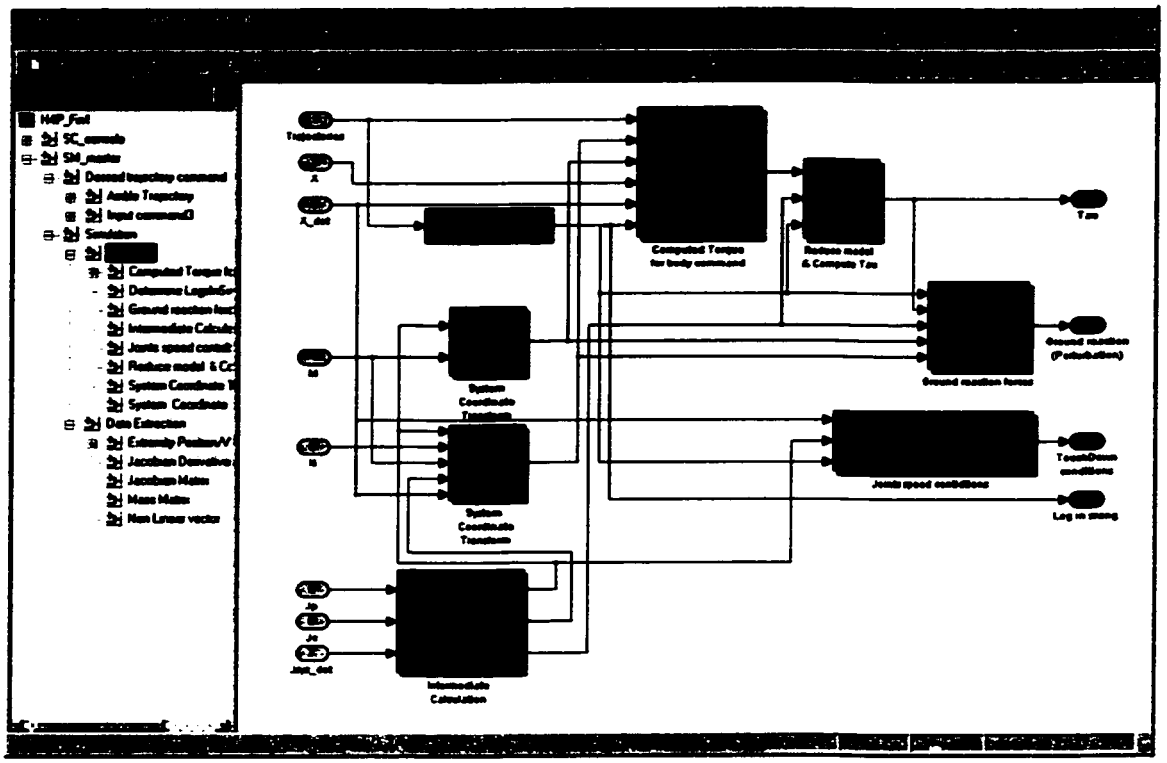


SOT4/Master :

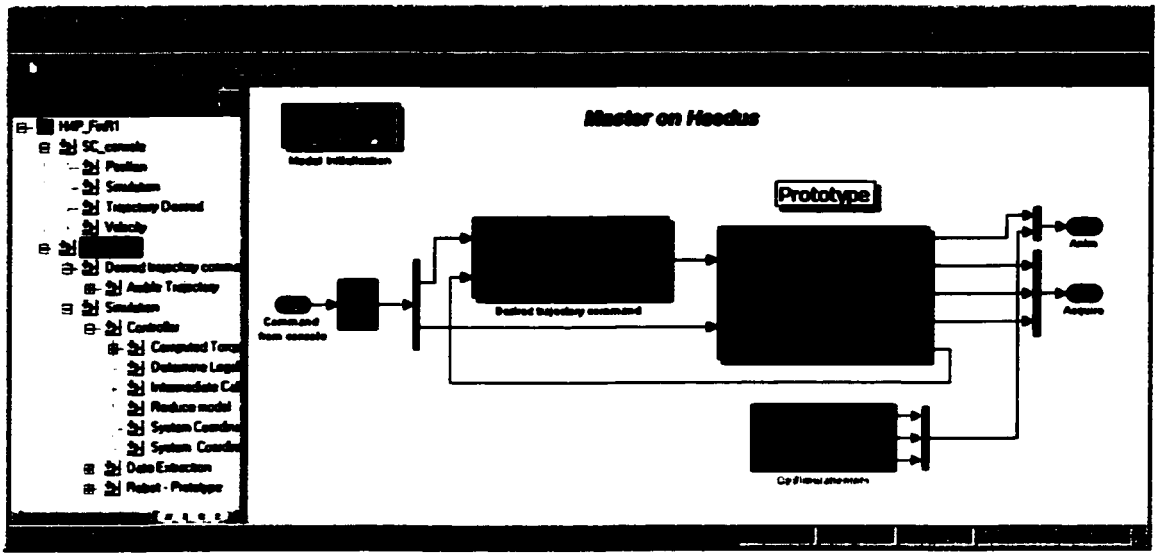

Master For simulation
Simulation
Prototype
Simulation or Desired
Command from console
OpComm
Controller
Analog Out
Encoder
Analog In
Tau

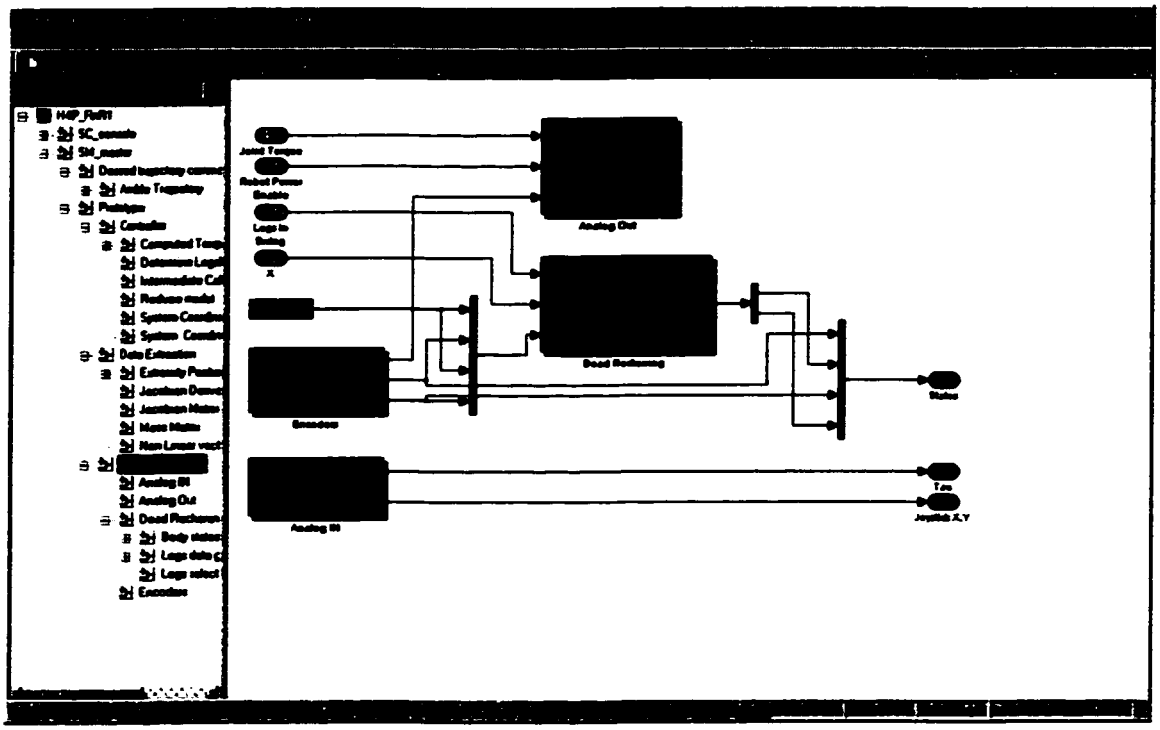
H4P_Fin1/Master :





H4P_FINR1 :





ANNEXE F

Étapes exécutées lors de l'initialisation du modèle

ÉTAPES EXÉCUTÉES LORS DE L'INITIALISATION DU MODÈLE

Voici un organigramme textuel pour mieux comprendre comment sont organisées les informations pour initialiser le modèle, la simulation et le contrôleur.

Le fichier d'initialisation du système est structuré de la façon suivante :

Initialisation du modèle symofros
Initialisation des variables du modèle et celles du programme
Initialisation des constantes du modèle : masse, inertie, longueurs,...
Définition des noms des points de mesures du modèle
Calcul des gains du PID pour les trajectoires dans l'espace de travail
Lance Param pour définir les états initiaux et les paramètres de la s-function traj pour les trajectoires de patte pour la démarche Amble walk.
Initialisation de la matrice de compatibilité entre les moteurs et les articulations

ANNEXE G

Fichier d'initialisation des modèles : Initialize.m

FICHIER D'INITIALISATION DES MODÈLES : INITIALIZE. M

```
% Initialization File for the robot "H4P_Finl"
%
% Description: Load the H4P_finx.mdl file and push the initialize
%              button in order to setup the simulation.  Then start
%              the simulation itself.
%
% Copyright (c) 1998-2000 by the Canadian Space Agency (CSA)
% $Revision: 1.1 $ $Date: 2000/06/14 13:11:43 $
% First modification done by Patrick Lessard July 12 2000
% Last modifications done by Patrick Lessard January 12 2002

clear all
TRUE = 1; FALSE = 0;
IPOS = 1;  % 1 for Amble ready; 2 for ground position;
CTRL = 1;  % Controller type 1 = PD;  0 = PID

% Initialize the simulation environment.
symoInitialize

% Model name
ModelName = 'H4P';

% Determination of the model dimensions
% In Matlab, type: "help query" for information
% ============================================
[SysDim, SysSet, FlexLinkDim, Param, Frame, Info] = feval(ModelName,0);

% Other constants of initialisation
% ===================================
ndl = 2; %number or generalised coordinate per legs (TetaHip,TetaKnee)
nl  = 4; %number of legs (4 legs)
ndb = 3; %number of generalised coordinate for the body (X,Y,Oz)

% Constant Matrix of propagation of torque on the right joint
% Bp is a ndl*nl X SysDim.nInputForceTorque
% [O1H,O1G,O2H,O2G,O3H,O3G,O4H,O4G] x [Tau1H,Tau1G,Tau2H,Tau2G,Tau3H,Tau3G,Tau4H,Tau4G]
Bc=zeros(ndb,ndl*nl);
Bp=eye(ndl*nl);
B=[Bc;Bp];

% Set the unassigned parameters to their nominal values
% ====================================================
% It is at this point that unassigned parameter can be varied
% Theses name can be acces by Param.Name

g=9.81;
% Haedus (m, kg & kgm^2) D3=0.031  Real weight is 8.9 kg mean 20% more than predict by
CAD
D1=0.203; D2=0.178; D3=0.031; Ixx0=3.71e-5*1.2; Iyy0=1.54e-4*1.2; Izz0=1.38e-4*1.2;
M0=6.449*1.2;
L1=0.120; M1=0.194*1.2; Ixx1=4.99e-7*1.2; Iyy1=1.09e-7*1.2; Izz1=1.748e-3*1.2; cx1=0.002;
cy1=-0.043;
L2=0.109; M2=0.05*1.2; Ixx2=9.37e-8*1.2; Iyy2=4.16e-9*1.2; Izz2=0.279e-3*1.2; cx2=0;
cy2=-0.03;

AssignedParam.g_base  = g;
AssignedParam.D1_Body = D1;
AssignedParam.D2_Body = D2;
AssignedParam.D3_Body = D3;
AssignedParam.M0_Body = M0;
AssignedParam.Ixx0_Body = Ixx0;
```

```
AssignedParam.Iyy0_Body = Iyy0;
AssignedParam.Izz0_Body = Izz0;

AssignedParam.L2_LowerLeg1    = L2;
AssignedParam.M2_LowerLeg1    = M2;
AssignedParam.Ixx2_LowerLeg1  = Ixx2;
AssignedParam.Iyy2_LowerLeg1  = Iyy2;
AssignedParam.Izz2_LowerLeg1  = Izz2;
AssignedParam.cx2_LowerLeg1   = cx2;
AssignedParam.cy2_LowerLeg1   = cy2;

AssignedParam.L2_LowerLeg2    = L2;
AssignedParam.M2_LowerLeg2    = M2;
AssignedParam.Ixx2_LowerLeg2  = Ixx2;
AssignedParam.Iyy2_LowerLeg2  = Iyy2;
AssignedParam.Izz2_LowerLeg2  = Izz2;
AssignedParam.cx2_LowerLeg2   = cx2;
AssignedParam.cy2_LowerLeg2   = cy2;

AssignedParam.L2_LowerLeg3    = L2;
AssignedParam.M2_LowerLeg3    = M2;
AssignedParam.Ixx2_LowerLeg3  = Ixx2;
AssignedParam.Iyy2_LowerLeg3  = Iyy2;
AssignedParam.Izz2_LowerLeg3  = Izz2;
AssignedParam.cx2_LowerLeg3   = cx2;
AssignedParam.cy2_LowerLeg3   = cy2;

AssignedParam.L2_LowerLeg4    = L2;
AssignedParam.M2_LowerLeg4    = M2;
AssignedParam.Ixx2_LowerLeg4  = Ixx2;
AssignedParam.Iyy2_LowerLeg4  = Iyy2;
AssignedParam.Izz2_LowerLeg4  = Izz2;
AssignedParam.cx2_LowerLeg4   = cx2;
AssignedParam.cy2_LowerLeg4   = cy2;

AssignedParam.L1_UpperLeg1    = L1;
AssignedParam.M1_UpperLeg1    = M1;
AssignedParam.Ixx1_UpperLeg1  = Ixx1;
AssignedParam.Iyy1_UpperLeg1  = Iyy1;
AssignedParam.Izz1_UpperLeg1  = Izz1;
AssignedParam.cx1_UpperLeg1   = cx1;
AssignedParam.cy1_UpperLeg1   = cy1;

AssignedParam.L1_UpperLeg2    = L1;
AssignedParam.M1_UpperLeg2    = M1;
AssignedParam.Ixx1_UpperLeg2  = Ixx1;
AssignedParam.Iyy1_UpperLeg2  = Iyy1;
AssignedParam.Izz1_UpperLeg2  = Izz1;
AssignedParam.cx1_UpperLeg2   = cx1;
AssignedParam.cy1_UpperLeg2   = cy1;

AssignedParam.L1_UpperLeg3    = L1;
AssignedParam.M1_UpperLeg3    = M1;
AssignedParam.Ixx1_UpperLeg3  = Ixx1;
AssignedParam.Iyy1_UpperLeg3  = Iyy1;
AssignedParam.Izz1_UpperLeg3  = Izz1;
AssignedParam.cx1_UpperLeg3   = cx1;
AssignedParam.cy1_UpperLeg3   = cy1;

AssignedParam.L1_UpperLeg4    = L1;
AssignedParam.M1_UpperLeg4    = M1;
AssignedParam.Ixx1_UpperLeg4  = Ixx1;
AssignedParam.Iyy1_UpperLeg4  = Iyy1;
AssignedParam.Izz1_UpperLeg4  = Izz1;
AssignedParam.cx1_UpperLeg4   = cx1;
AssignedParam.cy1_UpperLeg4   = cy1;

Param.Value = Param.NomVal;
if exist('AssignedParam','var'),
   Param = symoSetModelParam(AssignedParam,Param);
end;
```

```
% Extremity point name    (Can be access by Frame.Extremity)
% ========================================================
CenterOfMass = 'X_Y_OZ[1]';
Hip1            = 'Body[1]';
Hip2            = 'Body[2]';
Hip3            = 'Body[3]';
Hip4            = 'Body[4]';
Knee1           = 'UpperLeg1[1]';
Knee2           = 'UpperLeg2[1]';
Knee3           = 'UpperLeg3[1]';
Knee4           = 'UpperLeg4[1]';
Sabot1      = 'Foot1[1]';
Sabot2      = 'Foot2[1]';
Sabot3      = 'Foot3[1]';
Sabot4      = 'Foot4[1]';


% Set variables for simulation
% ============================
dsr0 = zeros(SysDim.ndsr,1); % initial condition for velocity of Generalysed Coordinate
u0 = zeros(SysDim.nInput,1); % initial condition for input
lambda0 = zeros(SysDim.nPC,1); % initial value on constraint forces


% Initial robot position
% Initial conditions on the rigid coordinates
% in order: Body(X,Y,Oz), Legs(Xi,Yi) i= 1..4
% ========================================
param              % Initialize parameters to define trajectory size

xoff=+0.02;     %used to offset trajectories

                   % Establish initial position for forward dynamic qr0
switch (IPOS)
case(1)        % This position is get from alex trajectory definition for Amble Walk
   InitPos=[0, hauteur-D3, 0,Pos_prec(1)-xoff,-hauteur,Pos_prec(3)-xoff,-
hauteur,Pos_prec(2)+xoff,-hauteur,Pos_prec(4)+xoff,-hauteur];
   kinematic
   qr0=qr1;

case(2)        % that is a precalculate position for half stand up
   bodyh =(L1+L2)*.75;
   InitPos=[xoff,bodyh,0, -xoff, -(bodyh+D3), -xoff, -(bodyh+D3), -xoff, -(bodyh+D3), -
xoff, -(bodyh+D3)];
   kinematic
    qr0=qr1;
   clear bodyh xoff;

end


% Trajectory definition
% Simulink block "Repeating sequence" use
% timeseq for vector of time and
% tx for Trajectory in X, typp for Trajectory of Y dot dot etc.
% ============================================================

   r=0.025;     %ray of body circle
   Tf=6;        %Sequence duration (secs)
   timeseq=0:0.001:Tf;
   Teta=pi*(1-cos(timeseq*2*pi/Tf));
   Tetap=sin(timeseq*2*pi/Tf)*2*pi*pi/Tf;
   Tetapp=cos(timeseq*2*pi/Tf)*2*pi^2*2*pi/Tf^2;
   tx=r.*cos(Teta)-r+qr0(1);
   ty=r.*sin(Teta)+qr0(2);
   txp=-r.*sin(Teta).*(Tetap);
   typ=r.*cos(Teta).*(Tetap);
   txpp=-r.*cos(Teta).*(Tetap).^2-r.*sin(Teta).*(Tetapp);
   typp=-r.*sin(Teta).*(Tetap).^2+r.*cos(Teta).*(Tetapp);
```

```
   clear Tf Teta Tetap Tetapp r ;

% Set the parameters for the contact dynamics model
% This is the model of the ground use in the very first models
% ============================================================
if (0)
RWall = [0, 0, 0];     % Vector from inertial frame origin to wall frame origin (m)
RotWall=[1 0 0;0 0 1;0 -1 0]; % Rotation Matrix defining orientation of wall frame wrt
fix frame
                              % z axis must be perpendicular to the contact plane
KWall = 2.0e4;                % Contact Stiffness N/m
FWall = 200.0;                % Contact damping M/m/s
Muk = 3;                      % Kinetic Friction coefficient
Mus = 4;                      % Static Friction coefficient
Sigma0 = 3.0e3;               % Bristtle stiffness (N/m)
Sigma1 = 0.01;                % Bristtle damping (M/m/s)
FViscous = 0.1;               % Surface Friction Viscous Damping (N/m/s)

% Calculations to determine quaternion
% ------------------------------------
vectRotWall=0.5*[RotWall(3,2)-RotWall(2,3),RotWall(1,3)-RotWall(3,1),RotWall(2,1)-
RotWall(1,2)];
u=vectRotWall/norm(vectRotWall);
sin_alpha=norm(vectRotWall);
cos_alpha=(trace(RotWall)-1)/2;
alpha=atan2(sin_alpha,cos_alpha);
PWall=[cos(alpha/2),u*sin(alpha/2)]; % Euler quaternion defining orientation of wall
frame wrt fix frame
end


% Cartesian space PID Controllers for absolute trajectory following
% =================================================================
if (CTRL)                 % Controller type 1 = PD;  0 = PID
Trd = 0.1;                % Desired response time
zeta=1;                   % Critical damping
Omega= 4.7/Trd;           % Wn Natural frequency desired for the system
Kp = ones(ndb+ndl*nl,1)*Omega^2;
Kd = ones(ndb+ndl*nl,1)*2*zeta*Omega;
Ki = zeros(ndb+ndl*nl,1);

else
TrB = 0.1*5;        %Body Response Time  NOTE: for first tests, *by 5 response time for
protect hardware
TrL = 0.1*5;        %Legs Response Time
zeta =1;            %Critical Overshoot
WnB=6.3/TrB;        %PID for Body
KdB=3*WnB;
KpB=3*WnB^2;
KiB=WnB^3;
WnL=6.3/TrL;        %PID for Legs
KdL=3*WnL;
KpL=3*WnL^2;
KiL=WnL^3;
Kp=[ones(ndb,1)*KpB; ones(ndl*nl,1)*KpL];
Ki=[ones(ndb,1)*KiB; ones(ndl*nl,1)*KiL];
Kd=[ones(ndb,1)*KdB; ones(ndl*nl,1)*KdL];
end
PID_Int_limit=1000;  %Saturation on I of the PID


% Motor initialization
% See Maxon catalog P53 for 113778 Moteur with gearhead 110367
% ============================================================
R  = 2.07;                %Résistance interne (Ohms)
L  = 0.62e-3;             %Inductance interne (Henri)
bm = 0;                   %Amortissement/friction statique (Nm//rad/sec)
Kmt= 52.5e-3;             %Constante de couple (Nm/A)
Kms= 1/(182*2*pi/60);     %Constante de vitesse (Volt//rad/sec)
Jm = (69.6+1.87)*1e-7;    %Inertie du moteur + inertie de l'engrenage (Kg.m^2)
```

```
Ng  = 23;                   %Rapport d'engrenage (23:1)
JL = 0;                     %Inertie de la charge
bL = 0;                     %friction statique de la charge
b  = bm+bL/Ng^2;            %friction total
J  = Jm + JL/Ng^2;          %inertie total
St = 1.070;                 %Stall torque (Nm)
Ct  = .113;                 %Maximum continuous torque (Nm)

Rm = 0.85;                  %Rendement du moteur
Re = 0.75;                  %Rendement engrenage
Ke = 23*1.5;                %Rapport engrenage et poulie
Gas =4;                     %Gain à la sortie de l'Ampli (Amps/Volts)
Gae =2.5;                   %Gain à l'entrée de l'Ampli  (Amps/Volts)
MaxTau  =St*Ke;             %Max joint stall torque (Nm)
MaxCTau =Ct*Ke;             %Max continuous torque at joint (Nm)

% Those variables was used for motor model controller in the first test with maxon motor
% ---------------------------------------------------------------------------------------
%Ka = 42/5;                 %Gain de l'amplificateur (volts)
%K = Ka*Kmt/L;              %Gain total du système
%Tr = 3*L/R;                %Temps de réponse partie électrique
%Trd= 3*Tr;                 %Temps de réponse désiré pour calculer les gains
%Wn = 4.73/Trd;             %Calcul des Gains P et I pour le contrôleur de couple
%Kim=Wn^2/K;                %Motor torque control integral gain
%Kpm=(2*Wn R/L)/K;          %Motor torque control proportional gain

% Drive amplificator caracteristics
% Check datasheet for the Electromate 25A8 drive
%
Ka = 5/2;                   % Volts/Amps
Ga = Ka/Kmt/Re;             % Nm * Volts/Amps * Amps/Nm = Volts

% Coupling matrix for position, speed and torque mapping from motor to joint and joint to
motor
% Convention to obtain matrix is CCW positive for Joint according to left side view
% Convention for motor is CCW positive for looking from the output shaft
% The convention of the Motors is opposite to convention of the Joints
%
================================================================================
Tv_MJ=[0 0 -1 0 0 0 0 0;
       0 0 -1 -1 0 0 0 0;
       1 0 0 0 0 0 0 0;
       1 1 0 0 0 0 0 0;
       0 0 0 0 -1 0 0 0;
       0 0 0 0 -1 -1 0 0;
       0 0 0 0 0 0 1 0;
       0 0 0 0 0 0 1 1];   %(motors is horizontal, joints is vertical)
Tv_MJ =-Tv_MJ*Ke;           % Transform Velocity From Joints to Motors
Tv_JM=inv(Tv_MJ);           % Transform Velocity From Motors to Joints

Tt_JM=Tv_MJ';                  % Transform Torque Motors to Torque Joints
Tt_MJ=Tv_JM';                  % Transform Torque Joints to Torque Motors

MJm=eye(8)*Jm;
Jz1=Izz1+cy1^2*M1; Jz2=Izz2+cy2^2*M2;
MJj=diag([Jz1 Jz2 Jz1 Jz2 Jz1 Jz2 Jz1 Jz2]);
clear Jz1 Jz2
MJeq=Tt_JM*MJm*Tv_MJ;
MJtot=MJeq+MJj;

%Matrix to add to Mass Matrix for inertia of the motors
MMJ = MJeq;
MMJ_aug = [zeros(3,11);zeros(8,3),MMJ];  % MMJ with 11x11 format to add to Mass matrix

disp('Initialize Completed')
```

ANNEXE H

Convention de couleur utilisées dans Simulink pour la création de programme

CONVENTION DE COULEUR UTILISÉE DANS SIMULINK

Pour clarifier la lecture des programme Simulink développés, cette convention de couleur a été établie. Elle permet en un rien de temps de visualiser les entrées et le sorties d'un système, identifier les fonctions importantes en apportant une clarté au niveau de la présentation.

Code de couleurs:

Noir	(Black)	Contour des blocs
Blanc	(White)	Fond de l'écran
Rouge	(Red)	Sorties, *Forward Dynamic*, Animation
Vert	(Green)	PID, Contrôleur, Gain
Bleu	(Blue)	Icons RT-Lab
Cyan	(Cyan)	Icons SYMOFROS
Magenta	(Magenta)	*Enable, contiguous copy, delay, memory*
Jaune	(Yellow)	Entrée
Gris	(Gray)	*Scope*, fonction de transfert
Bleu pâle	(Light Blue)	*Slider*, constante, *input sequence*
Orange	(Orange)	Sous-système de fonction
Vert foncé	(Dark Green)	Sous-système d'information, pré-traitement

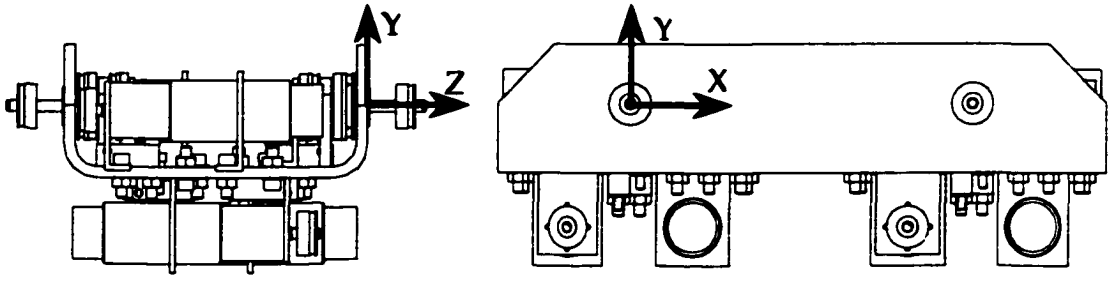
ANNEXE I

Mécanique du robot

MÉCANIQUE DU ROBOT

Paramètres techniques provenant des dessins assistés par ordinateur (DAO) d'Haedus utilisés pour la modélisation ajustés par mesure réelle

DONNÉES TECHNIQES – CORPS



Masse total du robot

19.58 Lbs = 8.9 kg

Masse du corps seulement

17.02 Lbs = 7.7 kg, (M0)

Centre de masse du corps référencé à l'axe de rotation de la patte 1

X: 4.00 po = 0.102 m, (D1/2)

Y: -1.222 po = -0.031 m, (D3)

Z: -3.500 po = -0.089 m, (D2/2)

Inerties (mesurées au centre de masse du corps)

I_{xx}: = 4.4520e-005 kg·m²

I_{yy}: = 1.8480e-004 kg·m²

I_{zz}: = 1.6560e-004 kg·m²

Position des axes de rotation des pattes par rapport référentiel de la patte 1

Patte 1 (avant gauche) :

X: 0 po = 0 m

Y: 0 po = 0 m

Z: 0 po = 0 m

Patte 2 (avant droite) :

X: 0 po = 0 m

Y: 0 po = 0 m

Z: -7 po = -0.178 m

Patte 3 (arrière droite) :

X: 8 po = 0.203 m

Y: 0 po = 0 m

Z: -7po = -0.178 m

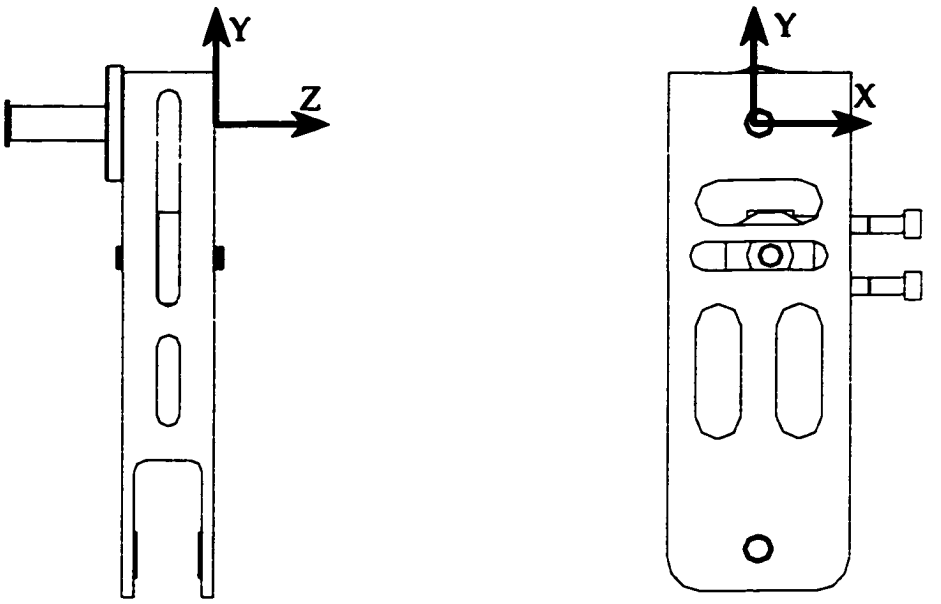
Patte 4 (arrière gauche) :

X: 8 po = 0.203 m

Y: 0 po = 0 m

Z: 0 po = 0 m

DONNÉES TECHNIQUES – CUISSE



Masse de la cuisse

0.429 Lbs = 0.194 kg (M1)

Centre de masse de la cuisse référencé à l'articulation de la hanche

Y: -1.690 po = -0.043 m (cy1)

Moment d'inertie effectif au centre de masse

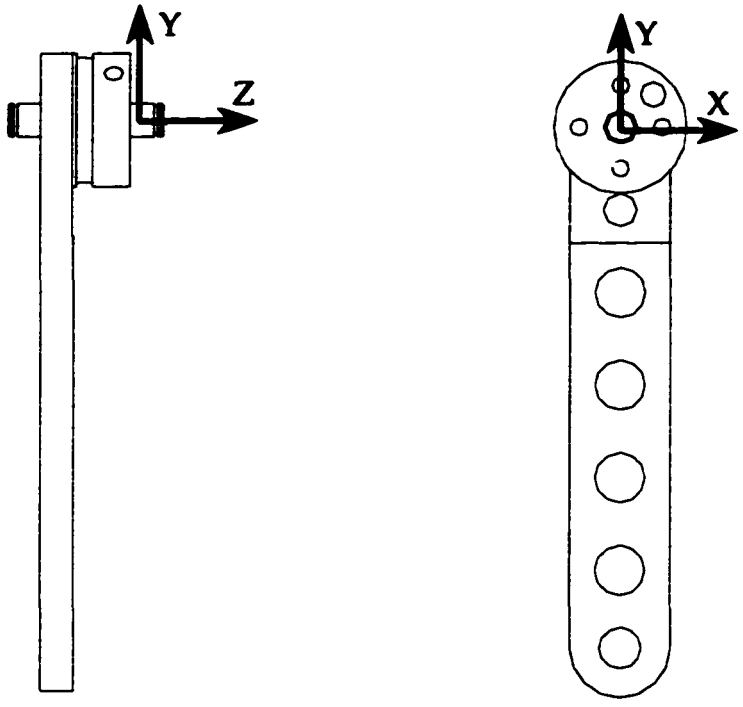
I_{zz}: = 0.0021 kg·m² (Izz1)

Position des axes

Axe du bas:

Y: -4.724 po = -0.120 m (L1)

DONNÉES TECHNIQES - MOLET



Masse du mollet

0.111 Lbs = 0.050 kg (M2)

Centre de masse du mollet appliqué au genou

Y: -1.166 po = -0.030 m (Cy2)

Moment d'inertie effectif appliqué au centre de masse

I_{zz}: 0.225 po^4 = 0.279 x 10^{-3} kg·m^2 (Izz2)

Position des axes

Axe du bas:

Y: -4.05 po = -0.104 m (L2)

ANNEXE J

Matériel utilisé

MATÉRIEL UTILISÉ

Le matériel utilisé pour la mise en œuvre du système est le suivant :

Console d'opération :

- AMD Athlon thunderbird 12 ghz
- Carte réseau D-Link 100 mb

Nœud de calcul :

- Intel P3 600 mhz
- Carte réseau 3Com 100 mb

Interconnexion réseau :

- Switch D-link de 5 ports

Système d'acquisition :

- 2 Cartes d'acquisition Sensoray 626 comprennent chacune : 4 sorties analogiques; 16 entrées analogiques; 48 entrées/sorties numériques et 6 entrées encodeurs quadrature.
- 2 borniers de raccordement conçu par Capra et réalisé par Circuit labo

Alimentation :

- 3 batteries acide-plomb 12 volts 7.2 amp-h

Amplificateurs de puissance :

- 8 amplificateurs Electromate Advanced Motion Control (AMC) 25A8 emprunté à l'ASC

Robot Haedus

- Robot réalisé par Capra; 4 pattes à 2DDL planaire par patte
    - 8 moteurs Maxon 118778 90 Watts avec engrenage planètaire 110367 et encodeur

ANNEXE K

Configuration des amplificateurs de puissance AMC 25A8

CONFIGURATION DES AMPLIFICATEURS AMC 25A8

Configuration matérielle :

Les amplificateurs sont utilisés en mode « courrant ». Pour ce faire, il faut configurer les interrupteurs SW1 à SW3 comme ceci : [Off Off On]

Ensuite, quatre potentiomètres sont à calibrer de la façon suivante :

P1 (Loop gain) mettre à zéro = maximum CCW

P2 (Current limit) 25kohms lecture entre le point central et le point près de la vis d'ajustement du potentiomètre.

P3 (Reference Gain) 2.7kohms

P4 (Test/Offset) calibrer pour que le moteur ne tourne pas à vide sans commande.

Gains de l'amplificateur :

Configuration du potentiomètre P2 : 25 kohms/52kohms = 0.4808

Le courrant maximal que l'amplificateur peut fournir est de 25A momentanément. Le courrant instantané maximum « Peak » est donc : 25A*0.4808 = 12.01 A

Comme la commande à l'entrée de la carte est +- 5V pour +- 12A, le gain est de 5/12= 0.4167, soit à peu près 0.4 ou à l'inverse : 2.5 = Gae(A/Nm).

Pour ce qui est de « ref in Gain », 2.7/52 = 0.0519 ou à l'inverse 19.26. Ceci devait faire en sorte que le +- 5V à l'entrée donne le +- 12 A à la sortie, ou le +- 20A que le moteur peut prendre.

Contrôleur de couple :

L'amplificateur de courant a été configuré en mode « current » pour contrôler le couple demandé au moteur. De cette façon, le système de contrôle est déchargé d'une tâche exigeante au niveau rapidité d'exécution. À rotor bloqué, le temps de réponse en courrant (couple) est de 20 ms à 63% de la consigne et de 30ms à 95%.

ANNEXE L

Définition des matrices de couplage, d'inertie et de transformation

MATRICES DE COUPLAGE, D'INERTIE ET DE TRANSFORMATION

Transposition des Positions/Vitesses et couples entre les moteurs et les articulations :

Le modèle ainsi que le contrôleur de couples travaillent dans l'espace des articulations des pattes. Pour commander ces mouvements, la consigne envoyée aux moteurs doit être transformée dans leur espace. Une matrice de transposition permet de faire ce passage de façon transparente au contrôleur et de faire la lecture des couples ou positions en les ramenant dans l'espace articulaire.

Pour définir cette matrice, deux conventions ont été considérées :

1- Le sens de rotation de l'articulation est positif tournant à l'inverse d'une horloge lorsque vu du côté (robot faisant face à gauche. C'est le sens positif autour de Z selon le modèle.

2- Le sens de rotation du moteur est positif tournant à l'inverse d'une horloge lorsque vu du côté de l'arbre sortant, encodeur derrière.

Une fois cette convention établie, la relation entre les actionneurs et les articulations a été établie à partir de la figure 4.7.

La matrice de transformation cinématique de l'espace articulaire à l'espace des actionneurs est décrite par la matrice T_MJ multipliée par le rapport de réduction des poulies de 1.5 :

$$T\_MJ = \begin{bmatrix} 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \cdot 1.5 \text{ (rapport des poulies)}$$

où l'axe vertical est décrit par les huit articulations et l'axe horizontal est décrit par les huit actionneurs.

Maintenant, la transformation de vitesse des articulations vers les moteurs peut être calculée en ajoutant le rapport d'engrenage Ke=23 provenant des engrenages planétaires multipliant la matrice précédente et en inversant le signe pour respecter la convention établie ci-haut :

$$\mathbf{Tv\_MJ} = -\mathbf{T\_MJ} \cdot \mathbf{Ke}$$

Ensuite, la transformation inverse pour obtenir les vitesses des articulations à partir des vitesses des moteurs est donnée par :

$$\mathbf{Tv\_JM} = \mathbf{Tv\_MJ}^{-1}$$

Puis, la transformation des couples moteurs vers les couples articulaires est donné par :

$$\mathbf{Tt\_JM} = \mathbf{Tv\_MJ}^{T}$$

Finalement, les couples moteurs peuvent être obtenus à partir des couples articulaires en utilisant la matrice obtenue suivante :

$$\mathbf{Tt\_MJ} = \mathbf{Tv\_JM}^{T}$$

Ces transformations peuvent être résumées par l'équation suivante :

$$\dot{\bar{\mathbf{q}}} = \mathbf{Tv\_MJ} \cdot \dot{\mathbf{q}}$$

$$\bar{\boldsymbol{\tau}} = \mathbf{Tv\_MJ}^{-t} \boldsymbol{\tau}$$

où $\mathbf{q}$, représente les moteurs, $\bar{\mathbf{q}}$, les articulations, **TV_MJ** la matrice de transformation et $\boldsymbol{\tau}$, le couple.

Matrices d'inertie :

Les couples à appliquer aux articulations sont directement proportionnels aux inerties des membres. Il y a deux inerties à prendre en considération : l'inertie du moteur avec la boîte d'engrenage et l'inertie du membre de la patte rattaché à l'articulation. Une fois ces deux inerties transformées dans l'espace articulaire, il apparaît que l'inertie du moteur est 3 fois plus grande que l'inertie de la patte. Ceci dû au rapport démultiplicatif. L'inertie des poulies a été négligée considérant son faible influence.

L'inertie de chacun des membres est estimée ainsi :

$$\mathbf{Izz_i} + \mathbf{Cy_i}^2 \cdot \mathbf{M_i} = \mathbf{J_i}$$

Où Izz_i est l'inertie au centre de masse de l'articulation i et Cy_i est la distance en y de l'articulation i au centre de masse du membre s'y rattachant et M_i est la masse du membre i.

Finalement, pour établir les matrices **MMJ** (matrice des inerties des moteurs vues par les articulations) et la matrice **Mjtot** où est ajouté l'inertie des membres vu par les articulations, il faut procéder au calcul suivant :

$$MMJ = Tt\_JM \cdot \begin{bmatrix} Jm & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & Jm & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & Jm & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & Jm & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & Jm & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & Jm & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & Jm & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & Jm \end{bmatrix} \cdot Tv\_MJ$$

où Jm est l'inertie totale de l'actionneur : moteur + engrenage.

La matrice d'inertie totale est ainsi estimée :

$$Mjtot = MMJ + \begin{bmatrix} J_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & J_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & J_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & J_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & J_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & J_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & J_1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & J_2 \end{bmatrix}$$

Cette dernière est une approximation du modèle réel puisqu'elle ne tient pas compte des non linéarités causées par le deuxième membre (mollet). Cette simplification est justifiable en raison du membre supérieur dont l'inertie est plus importante et l'inertie des actionneurs qui est dominante. Remarquez qu'en raison du couplage entre les deux articulations, ces matrices ne sont pas diagonales.

ANNEXE M

Dimensionnement des matrices du modèle d'Haedus

DIMENSIONNEMENT DES MATRICES DU MODÈLE D'HAEDUS

Symbole de la matrice	Dimension
M	11x11
N	11x1
τ	8x1
H	11x11
χ	11x1
q	8x1
Γ	11x1
B	11x8
F	11x1
u	11x1

ANNEXE N

Lecture des encodeurs

LECTURE DES ENCODEURS

Lecture des encodeurs

Les encodeurs retournent la position du moteur en impulsions. Pour traduire cette position en radian, il faut connaître le nombre d'impulsions par tour et la méthode de lecture de la carte. Généralement, la lecture d'un encodeur en quadrature avec décodage sur les fronts donne quatre fois plus d'impulsions par tour que le nombre de fente dont est conçu l'encodeur.

Notre encodeur compte 500 impulsions par tour et la lecture en quadrature multiplie ce nombre par 4. En rapportant en rad, :

2 pi / (500*4) = position en radians

Pour obtenir la vitesse de rotation, il suffit de dériver la position. Comme une dérivée exacte n'est pas réalisable dans un monde réel, celle-ci est remplacée par une approximation de la dérivée obtenue à l'aide d'un filtre. Il faut tenir compte du temps d'échantillonnage du système pour éviter les erreurs. La fréquence de coupure du filtre doit être plus élevée que celle du signal mais aussi plus faible que la fréquence d'échantillonnage. La fonction de transfert peut être de la forme suivante :

$$\frac{s\omega_c}{s+\omega_c},$$

ou encore sous la forme :

$$\frac{s}{\Gamma_f s+1}$$

où ω_c est la fréquence de coupure du filtre et Γ_f la période.

Pour respecter le théorème de Nyquist, Γ doit être 2.5 fois plus grand que la période d'échantillonnage. La période d'échantillonnage choisie pour le système est de 0.001 sec. En choisissant un Γ trois fois plus grand, la lecture est assurée. Ce qui donne une fréquence de coupure de 333 hertz. À cette fréquence, l'articulation de la patte devrait atteindre 5.75 tours par seconde pour atteindre la fréquence de coupure du filtre. La fonction de transfert suivante a donc été implantée dans le système pour estimer la vitesse à partir de la position.

$$\frac{s}{0.003s+1}$$

Ces positions et vitesses, une fois déterminées au niveau du moteur, sont transposées au niveau articulation en passant par la matrice Tv_JM. (Voir l'annexe L)

ANNEXE O

Photo d'Haedus

PHOTO D'HAEDUS