TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

COM	Center of mass
ZMP	Zéro moment point
ÉTS	École de technologie supérieure
SMC	Sliding mode Contrôle
TDE	Time delay estimation
CPU	Central processing Unit
DDL	Degré de liberté
DARPA	Defense Advanced Research Projects Agency
ROS	Robot operating system
IA	Intelligence artificial
SPL	Standard platform League
DH	Denavit–Hartenberg
3D LIPM	3 Dimensionnal Linear inverted Pendulum Mode
PID	Proportionnel-Integral-Dérivé

LISTE DES SYMBOLES ET UNITÉS DE MESURE

kg	Kilogramme
mm	Millimètre
cm	Centimètre
km/h	Kilomètre par heure
mm/s	millimètre par seconde

INTRODUCTION

En 2020, la robotique se retrouve tout autour de nous, elle peut prendre plusieurs formes et être utilisée dans plusieurs situations. Le domaine de la recherche pour la robotique est très populaire. Une communauté internationale se passionne pour des projets de robotique et se rassemble année après année pour tester ses recherches en situation réelle contre d'autres chercheurs et faire avancer leur recherche plus rapidement. La RoboCup[1] est une compétition qui accueille une partie de cette communauté internationale. Le soccer est le sport le plus joué durant cette compétition. Il y a cependant une petite variante par rapport aux athlètes qui forment l'équipe, les joueurs sont des robots humanoïdes autonomes présentés à la figure 0.1. Chaque équipe comporte 5 joueurs et une fois le sifflet de l'arbitre entendu, les robots sont laissés à eux-mêmes. Leur objectif est de gagner la partie.

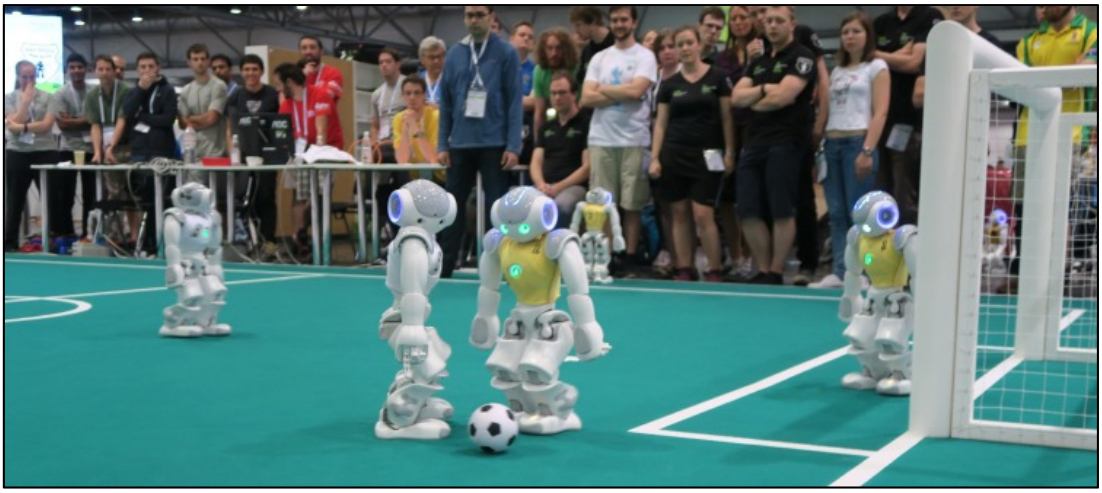

Figure 0.1 Division de soccer standard RoboCup
Tirée de RoboCup Fédération (2016)

Naova, une équipe d'étudiants dans différents programmes de génie de l'ÉTS, représente maintenant le Canada au Championnat du monde de la RoboCup. Naova participe avec une équipe de joueur de soccer robotique humanoïde autonome. La figure 0.2 présente l'équipe 2018 de Naova.

Figure 0.2 Équipe Naova de l'ÉTS

0.1 Motivation

Jouer au soccer est intuitif pour nous, les humains, grâce à nos habiletés favorisant le travail d'équipe. Il y a également beaucoup d'autres facteurs (communication, toucher, vue, course, stratégies) qui sont utilisés en même temps. En regroupant les recherches individuelles sur la robotique bipède, nous constatons que chacun des aspects est traité pour des cas précis. En considérant la marche, la stabilité, la vision, la collaboration et la communication, il devient possible, avec une bonne intégration, de faire une équipe constituée de robots bipèdes qui jouent au soccer. C'est la mission que le club scientifique de l'École de technologie supérieure Naova a réalisée.

Durant une partie de soccer, la tâche première est de trouver le ballon. Une fois localisé, le robot va commencer à marcher en direction du ballon. Il arrive souvent que deux robots arrivent en même temps sur le ballon. À ce moment, c'est le robot le plus solide sur ses

jambes qui va réussir à pousser l'autre joueur et botter le ballon vers le but adverse. Pour éviter cette confrontation, il est possible de miser sur la vitesse. Si le robot arrive avant l'adversaire sur le ballon, il est en mesure de la frapper dans le territoire adverse plus rapidement. Dans les deux scénarios, le meilleur robot est celui qui reste debout le plus longtemps. Dans ce mémoire, la solution aidera les robots de type Nao en situation de jeu. Une solution pour offrir une meilleure stabilité est proposée pour aider les robots à améliorer leur compétitivité de manière générale. Cette solution sera testée et mise en contexte contre d'autres méthodes que les autres équipes utilisent.

0.2 La marche du Nao est complexe

Lorsque nous pensons à la marche d'un robot, nous n'imaginons pas que ce soit un mécanisme complexe. Cependant beaucoup d'éléments doivent être pris en compte pour que le robot enchaine deux pas. Transposer ce mécanisme humain à un robot permet de décortiquer chacun de ses mouvements. Atlas[2], le robot de Boston Dynamics qui fait des acrobaties, le robot de Honda, Asimo[3] qui court de façon surprenante. De plus, il est possible de voir les centaines de robots humanoïdes qui participent à la RoboCup depuis 2004 et qui sont toutes capables de marcher. Ces robots sont tous différents par leur taille et leurs conceptions possèdent des méthodes de programmation différentes.

La méthode de déplacement d'un robot roulant et d'un robot bipède est très différente, car le robot à roues utilise un contact continu avec le sol, tandis que le robot avec deux jambes utilise la force de contact entre le pied et le sol. Il doit donc enchainer les pas pour pouvoir se déplacer. Il peut également avoir en parallèle un module pour la gestion du basculement en cas de chute.

Une fois la marche acquise par le robot, beaucoup de travail reste à faire. Il est possible que le robot évolue sur des terrains différents, avec des obstacles statiques et voir même avec des obstacles dynamiques qui bougent autour de lui. Ainsi, dans ce type de situation, la marche inclut des notions d'équilibre et par conséquent, la robustesse de la méthode choisie est plus

importante, car elle aura un impact sur les performances du robot. Imaginer un robot devant traverser un amas de roche pour aller porter des vivres à des survivants. La marche de ce robot en terrain accidenté va être critique et des vies vont dépendre de sa capacité à marcher en équilibre.

En dernier lieu, la linéarisation du mouvement de la marche rend la tâche difficile. Il est plus simple de travailler de manière linéaire. Grâce à une linéarisation du modèle, moins de puissance de calcul est requise. Ce compromis entre robustesse et complexité de calcul est satisfaisant pour le moment.

0.3 Problématique

La problématique étant large et ambitieuse, il sera donc également défini des bornes plus précises et réalistes. La phase 1 de la problématique est que Naova, un club scientifique de l'ÉTS, n'a pas son propre modèle mathématique représentant le robot Nao. Le modèle mathématique est utile pour envoyer des commandes aux robots ou faire des tests sur un simulateur. Actuellement, nous utilisons un modèle qui est fourni par une autre équipe. Il est donc difficile de bien maîtriser le modèle et de tester de nouveaux algorithmes pour se démarquer des autres équipes.

La phase deux de la problématique consiste en l'instabilité de nos robots et leur vitesse de marche lente sur la surface de compétition composée d'un tapis de soccer d'une épaisseur de 1,8 cm de haut. L'algorithme doit permettre au Nao de rester stable sur la surface de compétition. Il faut également que le robot reste debout lors des contacts avec les autres joueurs. Actuellement, le robot, en plus de manquer de stabilité, marche lentement. Pour ne pas tomber, la vitesse de la marche doit être inférieure à 350 mm/sec. La vitesse de prédilection pour une marche considérée stable par la communauté de la RoboCup est de 310 mm/s. Le tableau 0.1 représente l'erreur en degré et radian des joints des jambes de la marche du Nao avec la méthode populaire. L'erreur provient de la différence entre la valeur désirée

et la valeur réelle après l'action de marcher. Ces résultats ont été enregistrés avec une vitesse de 310 mm/s.

Tableau 0.1 Erreur des moteurs avec une vitesse de 310mm/s

Erreur des moteurs avec une vitesse de 310 mm/s		
Joints	Erreur radian	Erreur Degré
AnklePitch	0.078	4.49
AnkleRoll	0.010	0.62
HypPtich	0.291	16.69
HypRoll	0.010	0.60
HypYawPitch	0.011	0.68
KneePitch	0.088	5.05

0.4 Objectifs et contraintes

Pour répondre à la problématique soulevée à la section précédente, les deux objectifs suivants sont proposés :

- valider un modèle mathématique pour modéliser la marche du Nao;
- développer un algorithme de commande pour faire marcher le Nao;
  - baser l'algorithme sur un contrôle par mode de glissement;
  - inspirer par la méthode d'une estimation du délai de temps.

Pour atteindre ces objectifs, il faut respecter les contraintes suivantes :

- fonctionner avec le CPU du Nao;
- permettre une vitesse de plus de 350 mm/s sans faire de chute;
- avoir une erreur égale ou inférieure aux valeurs du tableau 0.1.

0.5 Méthodologie

Pour atteindre nos objectifs, la première étape consiste en la conception d'un environnement expérimental propice au projet. Un terrain de soccer règlementaire équipé d'ordinateurs et d'outils de contrôle sera créé. Les outils de contrôle sont :

1. Le « Game Controller » (un logiciel pour monitorer les parties).
2. SimRobot (Un logiciel de simulation).

Le « Game Gontroller » est utilisé pour virtualiser un arbitre et définir les règles que le robot doit suivre. Il a également une fonction importante, l'enregistrement des parties jouées.

SimRobot conçu par B-Human et adapté à nos besoins permet de calibrer les robots pour avoir à chaque partie les mêmes configurations. SimRobot nous permet de contrôler l'ensemble des paramètres nous donnant ainsi un contrôle sur notre environnement de test. Ces outils permettent de maximiser la compréhension des changements apportés, positifs ou négatifs.

Une fois cet environnement expérimental mis en place, l'aspect théorique de ce travail de recherche sera étudié, débutant avec la modélisation mathématique du Nao v5. Le modèle mathématique développé dans ce travail de recherche doit représenter le comportement réel de nos robots. Plus il est précis et plus il nous donnera les meilleurs résultats possible. Aussi le modèle de référence que nous utilisons sera validé et optimisé. Il sera question de calculer le modèle mathématique du Nao v5 pour nous permettre de le comparer avec les modèles des autres équipes.

La méthodologie de travail est rigoureuse, les résultats seront comparés avec d'autres approches ayant les mêmes buts. Comme l'objectif est d'apporter une amélioration à ce robot déterminé, les résultats seront comparés avec les algorithmes déjà implémentés dans le même robot choisi pour cette recherche. Cela permettra une meilleure validation du modèle présenté.

Ce projet de recherche d'une durée de deux ans est découpé en cinq phases :

1. Recherche et faisabilité;
2. Conception des algorithmes;
3. Tests en simulation;
4. Tests en laboratoire;
5. Participation à la RoboCup.

Ce processus est répété chaque année. Il est donc possible de développer de nouveaux algorithmes ou d'améliorer ce qui est déjà fait. Dans tous les cas, aux finales, ce qui est utilisé est validé, testé et comparé avec d'autres chercheurs pour avoir des commentaires ou des pistes de recherche lors d'impasse.

À la mi-année, un « White paper » ou un article scientifique est soumis au comité scientifique de la RoboCup pour valider l'avancement du projet ainsi que sa qualité scientifique. Ce fut donc une étape importante dans le processus de ce mémoire. Il en sera question dans les chapitres suivants.

0.6 Composition du mémoire

La composition de ce mémoire permet une compréhension logique. Il commence par le chapitre 1 qui va faire un survol des différentes notions de base qui seront utiles à la compréhension de la matière. Il sera suivi d'une revue de littérature, au chapitre 2 permettant d'avoir une vue d'ensemble des recherches faites autour du monde. Le chapitre 3 est la première étape de cette recherche, la détermination du modèle mathématique. Les chapitres 4 et 5 détailleront deux algorithmes développés qui seront utilisés pour répondre à l'objectif principal.

CHAPITRE 1

NOTIONS DE BASE

Avant de comprendre les sujets discutés dans le mémoire, il faut se familiariser avec certaines notions indissociables au domaine des robots bipèdes. Ce chapitre va aider à la compréhension générale des chapitres suivants. Il abordera des définitions sur les robots et des notions favorisant la compréhension globale du mémoire.

1.1 Le robot

Un robot est un système alimenté par une source d'énergie. Il évolue dans un environnement qui peut être défini ou indéfini. Il est constitué de plusieurs composantes, dont des capteurs, des microcontrôleurs et des moteurs. Il existe une grande variété pour chaque composante et, en fonction de la situation, il est possible de faire le meilleur choix possible pour le robot.

Le robot capte l'information de son environnement grâce aux capteurs qu'il dispose. Cette information enregistrée est par la suite traitée par son système, en fonction de son programme informatique qui lui transmet les instructions à suivre à la suite de l'interprétation des données. La machine est donc en mesure d'associer le résultat à une action à la fin du traitement. Ce processus répondant à des conditions de programmation envoie un signal et les parties du robot qui attendent ce signal pourront donc être activées pour accomplir une tâche précise grâce à ses actionneurs qui seront mis en marche. Le robot est fonctionnel grâce à son programme informatique qui peut être écrit dans différents langages selon le choix de son concepteur. Les langages les plus populaires sont C, C++, Python et bien d'autres. Le programme du robot est une suite de modules composés de lignes de commande qui représentent un algorithme de base ou complexe. L'information est traitée dans un microordinateur qui est l'élément électronique où les calculs et les décisions sont exécutés. Le microordinateur du robot est souvent associé au cerveau de la machine.

1.2 Les algorithmes

L'algorithme[4] est une méthode pour résoudre un type de problème, on mesure sa performance par sa durée de calcul, c'est-à-dire par sa consommation de mémoire vive et par la précision des résultats obtenus, etc. Les ordinateurs ou robots sur lesquels s'exécutent ces algorithmes ne sont pas infiniment rapides. Souvent ce sont des systèmes embarqués donc le temps de calcul d'une machine reste une ressource limitée, malgré une augmentation constante des performances des microordinateurs. Un algorithme sera performant s'il utilise les ressources dont il dispose avec une économie considérable, c'est-à-dire s'il limite la monopolisation du CPU ou optimise le travail coopératif. Des algorithmes peuvent être utilisés pour optimiser le temps du CPU, la mémoire vive et la consommation électrique. La consommation électrique est souvent un enjeu important lorsqu'un programmeur travaille sur un algorithme et qu'il désire tirer le meilleur du modèle. En effet, l'aspect d'économie électrique est en lien direct avec l'apport environnemental : la diminution ou l'optimisation de la gestion de l'énergie rend un projet plus favorable à la protection de l'environnement.

1.2.1 Le sliding mode

Le « sliding mode » est une méthode non linéaire de contrôle robuste utiliser fréquemment dans le domaine de la robotique. Cette approche est résumée à la figure 1.1 et illustre le principe de cette méthode. Le point bleu représente le système. De ce point le système fait un glissement suivant la flèche bleue, durant la phase qui se nomme « *sliding phase* », d'où le terme « *slide* » qui signifie glisse, pour l'apporter à la dynamique désirée. Une fois sur la ligne rouge, le point noir doit se rendre au point d'équilibre représenté par le point jaune. Cependant, entre le point noir et jaune, il y a un phénomène connu sous le nom de « *chattering* » ou en français de la réticence. Ce phénomène se produit, car il y a un signal qui passe de -1 à 1 dû à la fonction $sign(x)$ qui crée une discontinuation. Cette discontinuation ce produit rapidement sans arrêt et cause du tremblement sur les articulations du robot. Cette fonction est définie à l'équation 1.1 pour contrer ce tremblement ou tenter un

adoucissement, il faut donc ajouter une fonction continue, mais similaire à $sign(x)$ qui donnerait une meilleure convergence durant un temps fini et rendrait le système moins saccadé. En accord avec les auteurs Slotine et Li [5], la fonction de saturation serait une bonne approche. Elle consiste à remplacer le facteur de discontinuité de la commande $ksign(s)$ par la fonction plus souple. La fonction $ksat(S/\phi)$ présentée à l'équation 1.2 où S est la fonction de glissement, k le gain discontinue et ϕ, phi est la largeur de la couche limite. Il y a dans la littérature beaucoup de travaux sur le sujet et sur des méthodes pour diminuer le tremblement.

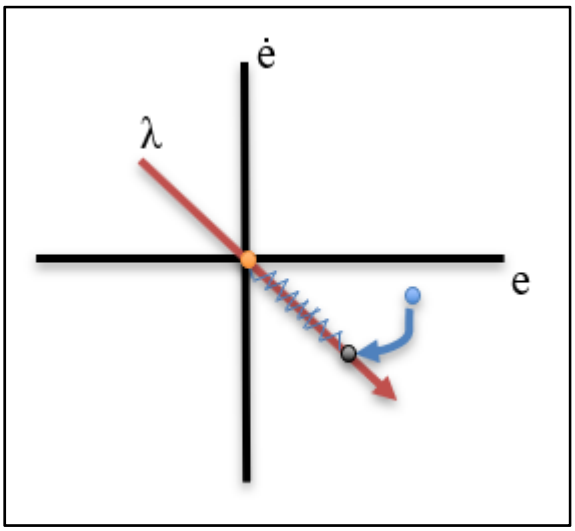

Figure 1.1 Illustration du sliding mode

$$sign(x) = \begin{cases} -1 & if\ x\ < 0 \\ 0 & if\ x\ = 0 \\ 1 & if\ x\ > 0 \end{cases} \tag{1.1}$$

$$sat(S/\phi) = \begin{cases} 1 & if \quad S/\phi > 1 \\ S/\phi & if \quad -1 \leq S/\phi \leq 1 \\ -1 & if \quad S/\phi > 1 \end{cases} \tag{1.2}$$

1.3 La marche bipède

La marche bipède est une action chez les humains, elle est simple et nous l'exécutons jour après jour sans y réfléchir. Pourtant, pour le robot bipède, la marche est plus complexe. Il doit être en mesure de commencer son parcours sur deux jambes et rester en équilibre. Une fois en position de départ, s'il est programmé pour une marche dynamique, alors il va lever une jambe pour commencer, atteindre une certaine hauteur, déplier la jambe et finalement la déposer devant lui à une certaine distance désirée. Explorons les prérequis pour cette action. La figure 1.2 présente deux plans d'un robot humanoïde, le plan A présente une vue de profil et le plan B présente une vue de haut montrant les emplacements des pieds du robot. La phase 1 représente les deux pieds (par des carrés noirs) stables au sol et le point rouge représente le centre de masse du robot qui bouge tout le long du mouvement afin de permettre un déplacement sans chute.

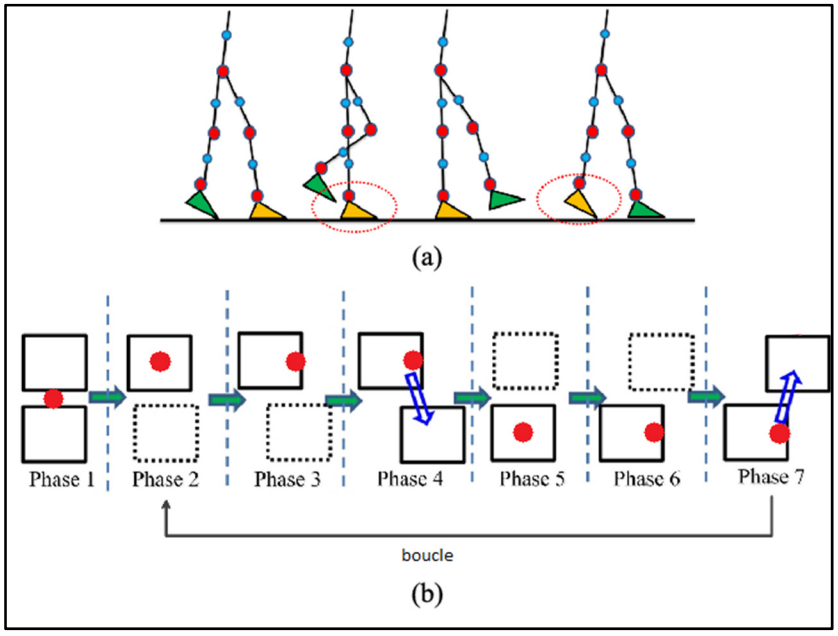

Figure 1.2 a) Image d'un robot bipède vue de profile
b) Représentation du COM
Tirée de Lin, Hwang, Jiang & Chen (2016, p.3)

Pour être en mesure de donner des consignes au robot, il faut connaitre son modèle mathématique. Pour cela, il faut calculer la cinématique directe et par la suite la cinématique inverse qui est expliquée dans les chapitres suivants.

Précédemment, la marche d'un robot programmé pour une marche dynamique a été expliquée. Ceci dit, il existe également des robots plus simples et programmés pour la marche statique. Dans le cas de la marche statique, les pas sont tellement petits, que le centre de masse du robot reste pratiquement toujours centré et donc en équilibre. Donc, comparativement à la marche statique, le point rouge de la Figure 1.2 ne bougerait pratiquement pas durant la marche.

Cela donne l'illusion que la marche statique est la solution simple pour garder un robot en équilibre. Cependant, cela ne règle rien, car la méthode statique est très peu robuste et donc au moindre changement dans l'environnement le robot est porté à faire une chute sans moyen de la contrer.

1.4 Degrés de liberté

L'espace qui nous entoure est tridimensionnel. Pour avoir la faculté de se déplacer dans l'espace, il faut avoir un degré de liberté (DDL) minimum. Il faut avoir six degrés de liberté pour faire un mouvement sans contrainte. Il y a deux types de liberté, la translation et la rotation. Chacun possède trois orientations connues avec les référentiels XYZ. Il faut les trois translations Tx, Ty, Tz et les trois rotations Rx, Ry, Rz pour former six degrés de liberté et offrir une liberté d'action dans l'espace qui nous entoure sans contrainte.

1.5 Conclusion

Le chapitre présent avait pour objectif de familiariser le lecteur et de rendre les chapitres suivants plus faciles à comprendre. Il y a encore des sujets sur la robotique non explorée mais pour cette recherche l'essentiel est couvert. La compréhension de la robotique est un atout qui gagne de la valeur aujourd'hui. Les recherches en robotique sont de plus en plus populaires et les sujets de plus en plus diversifiés. Dans le prochain chapitre, une revue de littérature montre une partie de ce phénomène.

CHAPITRE 2

REVUE DE LITTÉRATURE

La robotique comporte une grande variété de robots, souvent inspirés par la nature elle-même : il y en a avec six pattes, huit pattes, rampant et des robots à l'apparence humaine. Ce chapitre présente différents robots bipèdes populaires sur le marché et qui ont des traits communs avec le robot Nao utilisé dans ce mémoire. Il y aura également une brève introduction aux lois de commandes appliquées sur les robots bipèdes.

2.1 La robotique bipède existante

Dans le monde de la robotique bipède, il y a des incontournables. Il y a des robots qui sont commercialisés et d'autres encore exclusifs à la recherche.

2.1.1 Assimo

Assimo est le robot humanoïde conçu par Honda. Honda a commencé le projet avec une idée abstraite qui est devenue rapidement un concept réel. Toute l'énergie déployée pour Assimo, représenté à la figure 2.1, répond à rendre ce robot le plus mobile possible afin de l'aider dans des tâches et pouvoir vivre en harmonie avec les humains. Ce modèle humanoïde mesure 130 cm de haut et pèse 54 kg. Avec ses deux jambes, il marche à 2,7 km/h et peut courir à 6 km/h. Il a une autonomie d'environ 25 minutes.

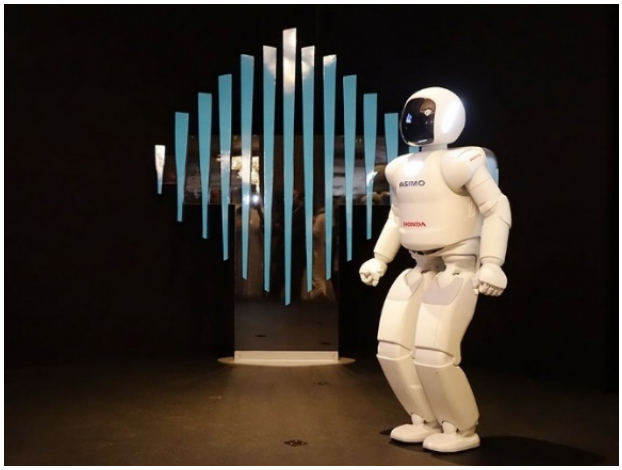

Figure 2.1 Assimo sur l'estrade à Dubai
Tirée de Honda (2015)

De façon plus concrète, l'équipe qui travaille avec Assimo est multidisciplinaire et leur recherche porte sur une grande variété de sujets [3]; nous retrouvons des algorithmes de marche, de vision, des prédictions de trajectoire, de la reconnaissance faciale et d'objets et de l'interaction avec les humains. Aujourd'hui, Honada a pour objectif de perfectionner Assimo en utilisant de l'intelligence artificielle (IA). Tous les modules comportent de l'IA, de l'apprentissage et de l'adaptation dans un environnement dynamique. Cela rend le projet Assimo très motivant pour l'équipe qui travaille sur ce robot. Ce robot est capable de faire de multiples tâches. Il cartographie son environnement en continu, il est en mesure de reconnaitre des voix et des objets.

Cependant, ce n'est pas un robot pour la vente, il est destiné la recherche et les ingénieurs de Honda veulent le développer davantage. [8]

2.1.2 Atlas

Atlas, présenté à la figure 2.2, est identifié comme étant « *the world's most dynamic humanoid* »[2] depuis son apparition dans une vidéo où il réalise un « back flip ». Pour accomplir cet exploit, il aura fallu plus de 100 000 heures de travail, car le robot n'est pas

doté d'intelligence artificielle, mais le tout est programmé de manière heuristique. Atlas est régi par des algorithmes très avancés, ce qui lui permet d'avoir une si grande robustesse. Il est le neuvième robot de la célèbre compagnie Boston Dynamics. Il est assez robuste pour avoir participé à la grande compétition de robotique, la DARPA. Du haut de ses cinq pieds et 330 livres, il accomplit des prouesses technologiques.

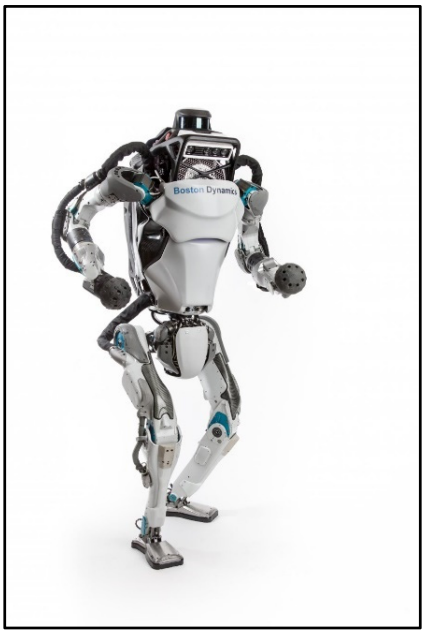

Figure 2.2 Robot Atlas
Tirée de Boston Dynamics (2018)

Cette machine possède 28 degrés de liberté [10] et est développée à partir de ROS (*Robot Operating System*), avec son environnement de simulation sur Gazebo pour faire le défi « simulation challenge » [11] de la compétition internationale, DARPA. Pour la cartographie, il utilise la technologie du point de « cloud », car elle permet de recueillir une grande quantité d'information sur l'environnement et cela en très peu de temps.

2.1.3 Roméo

Le robot Roméo [12] qui fait partie de la famille Nao a été développé par Aldebaran également, mais son but ultime est de venir en aide aux personnes avec des facultés affaiblies. Mesurant 146 cm et pesant 36 kg, il possède 37 degrés de liberté. Ce robot présenté à la figure 2.3 possède une apparence humaine et permet de faire avancer la recherche.

Figure 2.3 Roméo le robot
Tirée de Aldebaran (2013)

En comparaison avec Nao, il possède le même CPU, mais il en possède quatre, dont trois dans la tête et un dans le torse. Pour être à l'écoute de son environnement, il possède 16 micros qui lui permettent de bien déterminer la direction du son. Étant sur ROS, les quatre CPU ne sont pas inefficaces, car l'utilisation de ROS permet une bonne communication réseau.

2.1.4 Walker

Walker [14] est le robot de la compagnie UBTECH représenté en action à la figure 2.4, il a pour dimension une hauteur de 140 cm et un poids de 77 kg. Il totalise 36 degrés de liberté. La longueur de ses bras est de 540 mm et il peut prendre jusqu'à 1,5 kg pour chaque bras lors d'un étirement complet. Il communique par wifi 2, 4 ou 5G. Il possède une autonomie de deux heures pour une durée de chargement de deux heures. Le robot travaille avec un processeur Intel I7-7500U avec 2.7 GHz.

Le robot est récent et vient d'être annoncé sur le marché. Le but de ce robot est de faire partie des maisons pour rendre la vie dans les maisons plus intelligente.

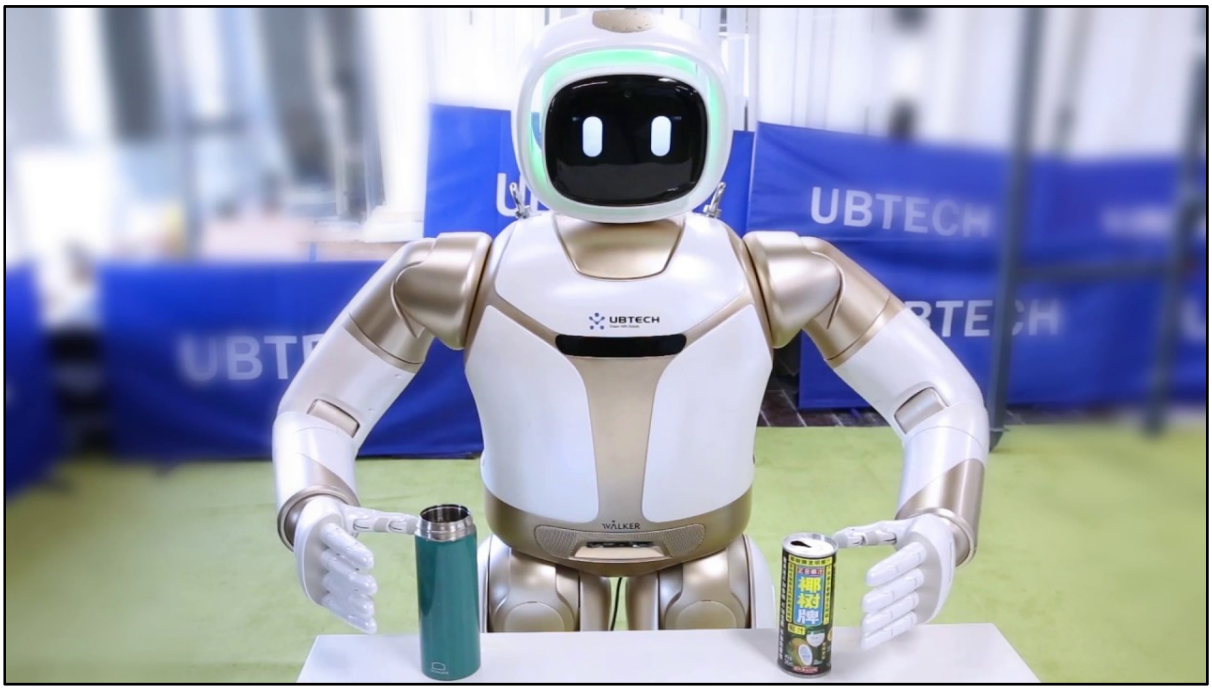

Figure 2.4 Walker utilisant les deux bras
Tirée de UBTECH (2012)

2.1.5 Darwin-OP

Darwin-OP [16] est un petit robot similaire au Nao. Il est humanoïde mesurant 45,5 cm de haut. Ce petit robot est doté de 20 degrés de liberté et pour bouger il va utiliser ses moteurs de type dynamixel MX-28T. Sa conception permet une multitude de mouvements telle que montrée à la figure 2.5. Darwin est une machine développée par Robotis, utilisant une plateforme "open source". Cela signifie qu'à la suite de l'acquisition de ce modèle, il est possible d'utiliser des algorithmes déjà fonctionnels avec ce robot. Il est en mesure de marcher, de danser et de parler. Ce robot se retrouve sur le marché et se vend à 12 000$ US.

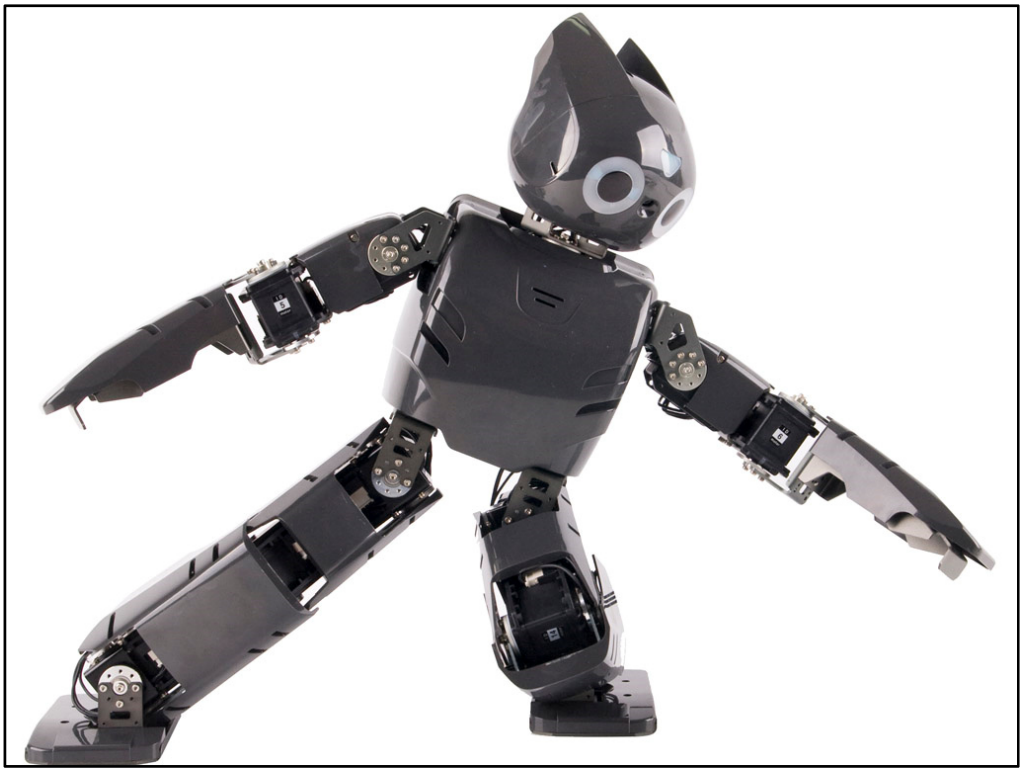

Figure 2.5 Darwin en mouvement
Tirée de IEEE Robots (2010)

2.1.6 Mercury

Mercury [18] présenté à la figure 2.6 est un robot bipède représentant les jambes d'un adulte. Il est conçu pour l'apprentissage de la dynamique de marche. Il a été développé par l'université UT Austin au Texas pour la somme de 150 000$ US. Ce modèle possède six degrés de liberté et est d'une hauteur de 150 cm. Pour se mouvoir, cette machine utilise un algorithme basé sur le modèle du « dynacore open-source whole-body locomotion controller ». Cet algorithme robuste permet à Mercury d'avoir une vitesse de marche de 4 km/h, se traduisant par trois pas par seconde.

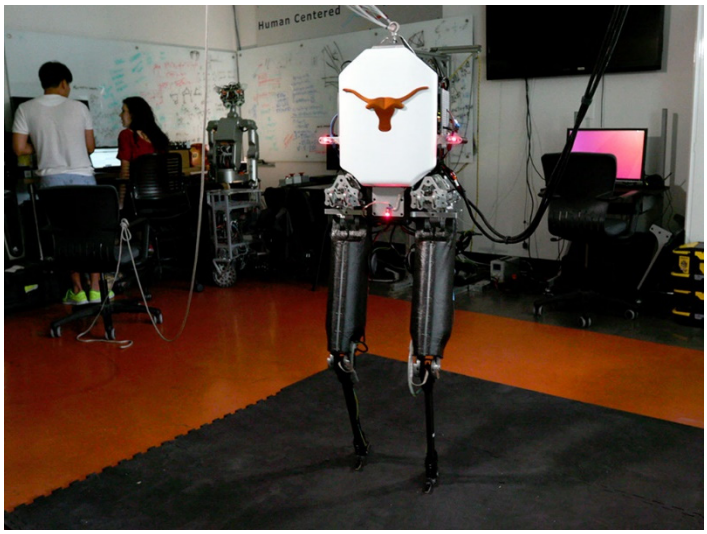

Figure 2.6 Mercury
Tirée de IEEE Robots (2018)

2.1.7 Valkyrie

Valkyrie [20] est le robot de la NASA. Avec ses six pieds et 300 livres, il possède une grande dextérité. Il possède un nombre important de 44 degrés de liberté. L'apparence de Valkyrie peut faire penser à Iron man, à cause du cercle lumineux sur sa poitrine, visible sur la figure 2.7. Cependant, cela n'a pas la fonctionnalité de l'alimentation, mais sert d'indicateur du statut du robot. Par exemple, la couleur bleue indique qu'un moteur est actif.

Figure 2.7 Valkyrie
Tirée de Young (2017)

En comparaison avec Atlas qui utilise un système hydraulique, Valkyrie possède un système moins puissant, mais, plus fiable pour les missions dans l'espace. Il est donc doté d'un système électrique pour sa mobilité. Pour cela, son alimentation lui permet de fonctionner pour une période d'une heure. Au total, cette machine hautement complexe a coûté deux millions US.

2.2 Algorithme de marche autonome

L’objectif d'une marche autonome est d’avoir un algorithme robuste pour permettre au robot de marcher rapidement de façon stable et sur des terrains de toute sorte. En somme, marcher comme un humain. C’est l’algorithme qui permet en partie cet exploit. L’algorithme de marche autonome est une série de modules qui tournent en boucle pour faire parvenir de l’information au robot. Il va avoir des modules qui donnent des ordres au système grâce aux données reçues. Il va avoir des modules de validation et de contrôle pour garantir la fiabilité du système. C’est une méthode pour régler une problématique. Les algorithmes se regroupent sous ce qui est appelé des lois de commandes ou de contrôle. Il y a les lois de contrôle linéaire et les lois de contrôle non linéaire. Dans le cas linéaire, nous retrouvons la loi du PID (proportionnel, intégral, dérivé) qui est le plus utilisé dans l’industrie, car ses corrections s’appliquent facilement à beaucoup de projets. Il y a également le contrôle linéaire quadratique ou encore le gain de Kalman (LQR). Cette méthode permet de calculer la matrice de gains d’une commande par retour d’état. Dans les commandes non linéaires, nous pouvons établir qu’une des plus utilisées est la commande du pendule inversé. Elle est très populaire pour beaucoup de robots humanoïdes bipèdes, mais manque encore de robustesse.

2.3 Conclusion

Il y a beaucoup de modèles différents de robot bipède. Grâce à cela, il est donc possible d’avoir un modèle de robot répondant au sujet de recherches de chaque chercheur. Chaque robot répond à un but différent, il devient donc important de bien comprendre les capacités et les limites du robot identifier dans la recherche, cela fera du robot un meilleur atout comme instrument de recherche.

CHAPITRE 3

MODÉLISATION MATHÉMATIQUE

La problématique de la cinématique directe et inverse est bien connue de chacune des équipes participantes à la RoboCup SPL. Lors de l'achat du robot Nao, la Compagnie Aldebaran offre un code contenant les informations sur la cinématique inverse. Elle donne directement un modèle permettant de travailler avec le robot. Cependant, il a été montré par expérimentation que la méthode d'Aldebaran n'était pas efficace et limitait la fonctionnalité de la marche. Elle ne permet pas de recevoir aucune entrée de données et ne fonctionne qu'avec la configuration des joints fournis par Aldebaran.

3.1 Les méthodes populaires

B-Human est une équipe participante à la RoboCup et qui annuellement rend son code public pour les autres équipes participantes. Elle a travaillé activement sur la problématique de la marche du Nao et sa méthode choisie est un algorithme qui utilise la cinématique inverse. C'est l'équipe qui a la méthodologie la plus similaire à la nôtre. Pour faire des comparatifs sur la méthodologie utilisée pour la marche du Nao, nous utiliserons également un morceau de code de l'équipe de Kourette de Grèce et un module de l'équipe rUNSWIFT d'Australie. Ces trois équipes sont des équipes populaires qui ont déjà gagné le Championnat du monde.

3.2 Présentation du Nao

Ce petit robot est populaire pour la recherche sur la planète entière. Il y a plusieurs universités qui l'utilisent pour la recherche avec les enfants autistes [22]. D'autres universités ont des équipes d'étudiants qui participent annuellement à la RoboCup, une compétition internationale de soccer robotique autonome, qui fait avancer la recherche rapidement, car l'objectif est de jouer en 2050 contre des équipes humaines. Chaque année, les équipes arrivent avec de nouveaux algorithmes de vision, de marche, de stabilité ou des stratégies et

doivent en faire des présentations aux autres équipes. La complexité du défi est dans la puissance du robot, car il ne possède qu'un tout petit CPU Atom 1.6 GHz et 1 Go de RAM. Le petit Nao communique par Wifi comme le schéma de la figure 3.1 le montre pour le déploiement des modules, il est également possible de le contrôler à distance via un logiciel nommé chorégraphe. Cependant, dans ce mémoire il est question de contrôle autonome donc tout comme les équipes, un environnement développé sur mesure est utilisé pour la simulation. La plateforme de l'entreprise ne sera donc pas utilisée, ce qui de toute façon limite le développement puisque la compagnie tient à garder certains éléments privés.

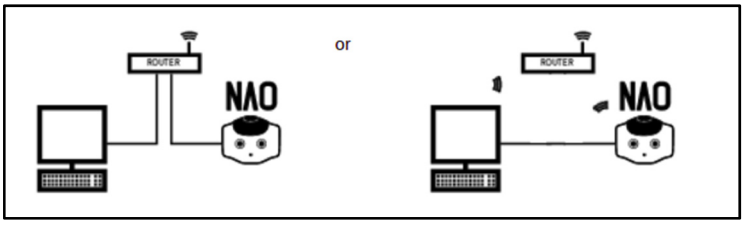

Figure 3.1 Communication avec le Nao
Tirée d'Aldébaran (2013)

Ce petit robot de 58 centimètres de haut est un modèle ayant 25 degrés de liberté, tel que la figure 3.2. Avec le module de marche d'origine, le robot marche à une vitesse de 100 mm/sec. Il est la version V5 de la compagnie Aldébaran bien que la version V6 est actuellement sur le marché. Ce mémoire ne parlera que de la version V5.

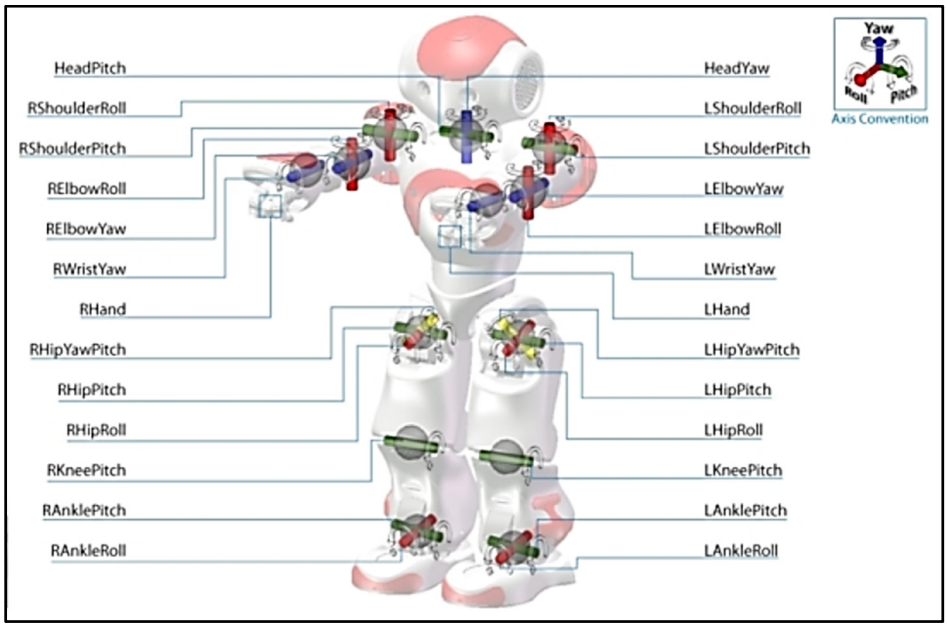

Figure 3.2 Model du Nao
Tirée de Aldebaran (2013)

Il y a 5 chaines différentes pour déterminer le modèle mathématique du robot. Une chaine est un assemblage de moteurs liés ensemble qui permettent le mouvement. Pour comprendre chaque élément de la chaine, il faut utiliser la figure 3.2 plus haut. Elle permet de voir une vue d'ensemble des liens sur le Nao avec le nom des joints. Le « Roll » est la rotation dans l'axe des X, le « Pitch » est la rotation dans l'axe des Y et le « Yaw » est la rotation dans l'axe des Z.

La première chaine est pour le mouvement de la tête du robot. Elle est constituée d'un seul moteur, mais possède deux axes de rotation. Ils sont appelés « HeadPitch » et « HeadYaw », et ensemble ils permettent le contrôle sur l'orientation des caméras. Le Nao possède deux caméras sur la tête : l'« UpperCamera » située au-dessus des yeux, elle permet une vision éloignée et la « LowerCamera » qui est située dans la bouche visant le sol et permet la vision rapprochée. La figure 3.3 montre la tête avec les différents angles atteignables pour les deux maillons de la chaine.

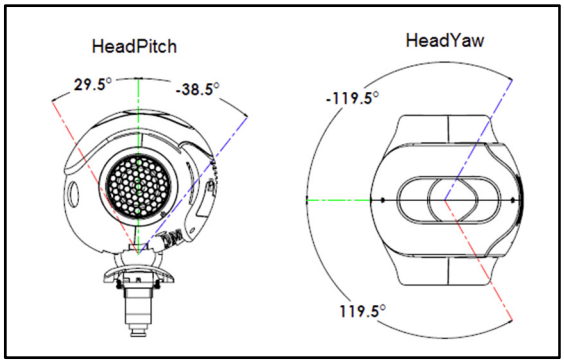

Figure 3.3 La tête du Nao
Tirée de Aldebaran (2013)

Pour la deuxième chaine et la troisième chaine, ils font référence au bras gauche et droit du robot. Il y a un total de trois moteurs dans chaque bras. Les deux bras sont identiques, mais il faut prendre en considération leur opposition. Cela va inverser le signe des angles possibles à atteindre pour les bras. Dans ces deux chaines, même si les doigts formant le maillon « Rhand » sont amovibles, ils sont exclus de la chaine et des équations. Les maillons qui forment la chaine du bras droit, représenté à la figure 3.4, sont identifiés par les noms « RShoulderRoll », « RshoulderPitch », « LelbowYaw », « LelbowRoll » et « LwristYaw ».

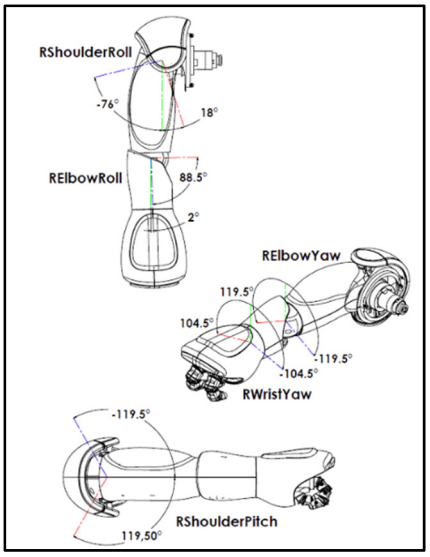

Figure 3.4 La chaine du bras droit
Tirée de Aldebaran (2013)

Pour le bras gauche, les maillons sont « RShoulderRoll », « RshoulderPitch », « LelbowYaw », « LelbowRoll » et « LwristYaw ». La figure 3.5 montre la chaine du bras gauche et en comparaison avec la chaine du bras droit, il est possible de voir l'inversion des signes.

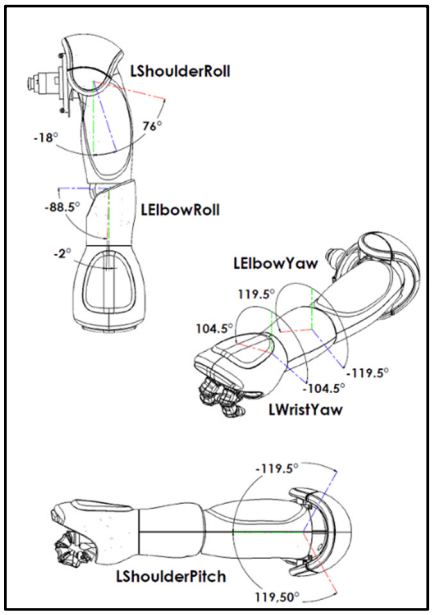

Figure 3.5 Bras gauche
Tirée de Aldebaran (2013)

Finalement, les chaines quatre et cinq représentent la jambe droite et gauche. Ces deux chaines sont liées par un joint unique, il est donc partagé par les deux jambes, mais possède cependant deux noms. Ce joint est nommé « RHipYawPitch » pour la jambe droite et « LHipYawPitch » pour la jambe gauche comme le montre la figure 3.6.

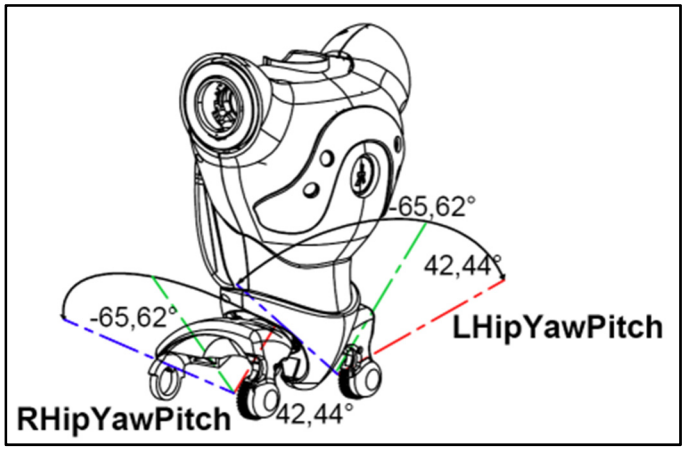

Figure 3.6 Articulation commune aux deux jambes
Tirée de Aldebaran (2013)

Pour la chaine de la jambe droite, qui est la chaine quatre, elle est composée des maillons « RHipYawPitch » présentés plus haut, suivis de « RHipPitch », « RhipRoll », « RkneePitch », « RanklePitch » et « RankleRoll » présentés à la figure 3.7.

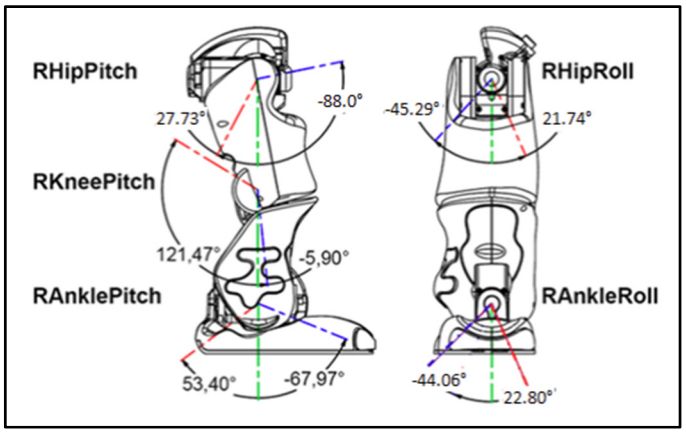

Figure 3.7 Chaine de la jambe droite
Tirée de Aldebaran (2013)

Comme pour les bras, les jambes sont identiques, mais inversées aussi. La dernière chaine est la cinq et comporte les éléments « LHipYawPitch », « LhipPitch », « LhipRoll », « LkneePitch », « LanklePitch » et « LankleRoll ». La figure 3.8 montre les maillons et les angles possibles pour chaque articulation.

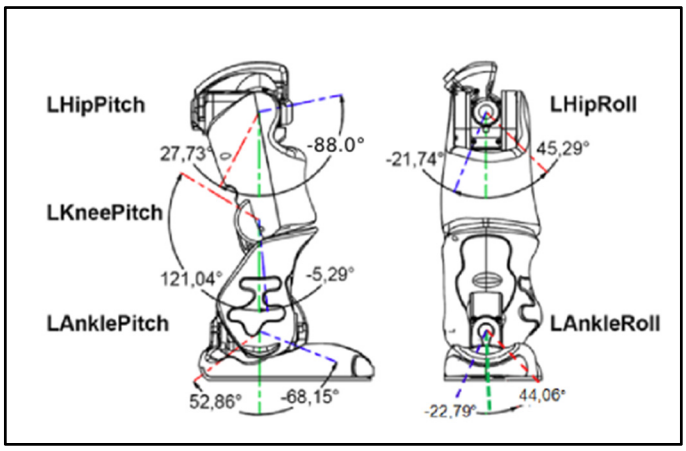

Figure 3.8 Chaine de la jambe gauche
Tirée de Aldebaran (2013)

Pour compléter le modèle du Nao, il nous faut connaitre la distance entre les joints et grâce à la figure 3.9 et la figure 3.10, il est possible de compléter notre modèle mathématique. Pour les tableaux des paramètres DH du chapitre 3, il va falloir utiliser les mesures du tableau 3.1 pour compléter les équations et développer les matrices de transformation de chaque chaine du Nao.

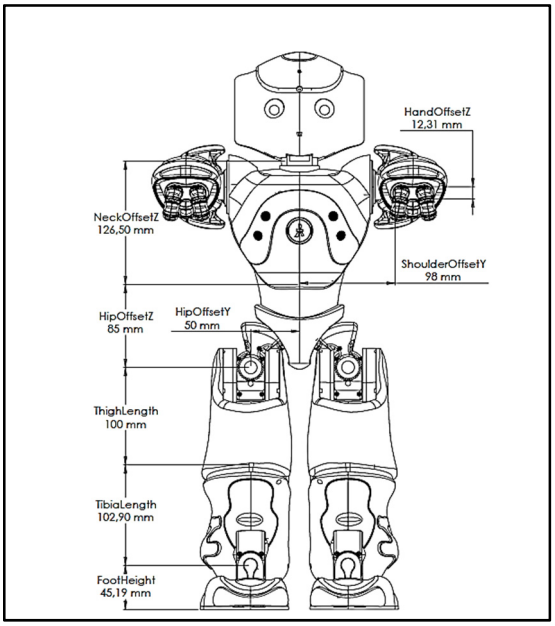

Figure 3.9 Distances entre
les pièces du robot Nao
Tirée de Aldebaran (2013)

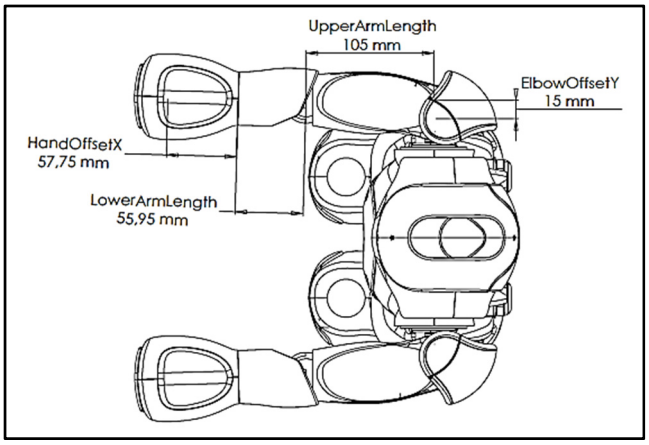

Figure 3.10 Distances des articulations
vue de haut
Tirée de Aldebaran (2013)

Tableau 3.1 Distance entre les articulations
Tirée de Aldebaran (2013)

Nom	**Distance (mm)**
NeckOffsetZ	126.50
ShoulderOffsetY	98.00
ElbowOffsetY	15.00
UpperArmlength	105.00
LowerArmLength	55.95
ShoulderOffsetZ	100.00
HandOffsetX	57.75
HipOffsetZ	85.00
HipOffsetY	50.00
ThighLength	100.00
TibiaLength	102.90
FootHeight	45.19
HandOffsetZ	12.31

Les informations reliées au centre de masse du Nao sont toutes fournies sur le site web d'Aldebaran. Pour chaque pièce, le fournisseur nous donne la masse en kilogramme (kg), le Centre de masse, l'équation 3.1 et la matrice d'inertie I_o, sous la forme matricielle présentée par l'équation 3.2

$$CoM(S) = \begin{bmatrix} X_G \\ Y_G \\ Z_G \end{bmatrix}_{(o,R)} (m) \quad (3.1)$$

Tirée de Aldebaran (2013)

$$[I_o(S)]_R = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}_R (kg * m^2) \quad (3.2)$$

Tirée de Aldebaran (2013)

Dans l'équation précédente, S est un solide du robot, O et R sont les coordonnées locales du solide S. Ces mesures sont relatives à la position zéro du robot. Cela implique qu'il est debout avec les bras levés comme sur la figure 3.9.

Pour trouver le Centre de masse (COM) total du robot qui regroupe toutes les masses individuelles de chaque pièce (S), il a été déterminé que le point principal était là où toutes les mesures convergent, puisqu'elles sont fournies par la compagnie. Il a donc été nécessaire de le faire refléter à cette position (X, Y, Z) chacune des matrices de chaque pièce de la chaine.

3.3 Convention Denavit-Hartenberg (DH)

Pour arriver à une modélisation mathématique du modèle du robot Nao, il faut suivre une convention. La convention de Denavit-Hartenberd (DH) est utilisée pour les prochains calculs.

La convention DH possède quatre paramètres et ils ont tous un rôle précis, ils donnent des informations entre deux coordonnées du robot à l'étude. Il y a deux paramètres de rotation et deux pour la translation.

Pour arriver à trouver la cinématique directe, il faut passer par certaines étapes en utilisant la convention DH.

Étape 1 : Localiser et identifier les axes d'un joint z_0, …,z_{n-1}.

Étape 2 : Établir les bases de l'environnement. Mettre l'origine n'importe où sur l'axe z_0. Une fois choisie, les x_0 et y_0 vont s'identifier simplement avec la convention de la main droite.

Étape 3 : Localiser l'origine de o_i où la normale de z_i et z_{i-1} croise z_1. Dans le cas ou z_1 croise z_{1-1}, localiser o_i au croisement. Cependant, si z_i et z_{i-1} sont parallèles, alors positionner o_i à n'importe quel point suivant z_i.

Étape 4 : Fixer x_i à un endroit, le long de la normale entre z_{i-1} et z_i où sur le plan $z_{i-1} - z_i$ s'il y a intersection.

Étape 5 : Avec la règle de la main droite, il est possible de finir en plaçant y_i.

Étape 6 : Définir le dernier joint de la chaine $o_n x_n y_n z_n$. Dans la mesure ou le dernier joint n'est pas rotatif, placer x_n et y_n avec la méthode de la main droite. z_n est égale à a dans la direction de z_{n-1}. Définir o_n par convention, le long de z_n, idéalement au milieu.

Étape 7 : Créer le tableau des paramètres DH au tableau 3.2 (en exemple) contenant $a_i, \alpha_i, d_i, \theta_i$

Tableau 3.2 Tableaux des paramètres DH

Link	$\boldsymbol{a_{i-1}}$	$\boldsymbol{\alpha_{i-1}}$	$\boldsymbol{d_i}$	$\boldsymbol{\theta_i}$
1	a_1	0	0	θ_1
2	a_2	0	0	θ_2

a_{i-1} = distance sur x_{i-1}, de o_{i-1} jusqu'à l'intersection de x_i et l'axe z_{i-1}.
α_i = l'angle entre z_{i-1} et z_i mesuré à partir de x_i.
d_i = distance sur z_{i-1}, de o_{i-1} jusqu'à l'intersection de x_i et l'axe z_{i-1}. Il est important de noter que dans le cas ou d_i est un joint prismatique, donc sans rotation, d_i va être variable.
θ_i = l'angle entre x_{i-1} et x_i mesuré à partir de z_i.

Étape 8 : Former la matrice de transformation homogène T_i.

Étape 9 : Former $T_n^0 = A_1 \dots A_n$. Cela va donner la position et l'orientation de la chaine sous la forme de coordonnée.

À la suite de cela, il est possible d'analyser une chaine à partir de son premier joint jusqu'à son dernier et d'avoir la matrice T grâce à la formule 3.3 suivante.

$$T_{DH} = \begin{bmatrix} cos\,\theta & -sin\,\theta & 0 & \alpha \\ sin\,\theta\,cos\,\alpha & cos\,\theta\,cos\,\alpha & -sin\,\alpha & -d\,sin\,\alpha \\ sin\,\theta\,cos\,\alpha & cos\,\theta\,sin\,\alpha & cos\,\alpha & d\,cos\,\alpha \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

3.4 Cinématique directe

Afin de bien commencer la cinématique directe, il faut définir un point central et un point de référence pour toutes les chaines. Dans cette optique, ce point va être le torse du Nao et la position initiale va être debout avec les bras levés. La raison est que cela permet l'utilisation des paramètres fournis par Aldébaran. Le point est donc montré à la figure 3.11. L'objectif ici est de comparer le modèle du Nao v4 calculé par Nikos Kofinas [28] avec le modèle du Nao v5 afin de déterminer si les résultats sont identiques.

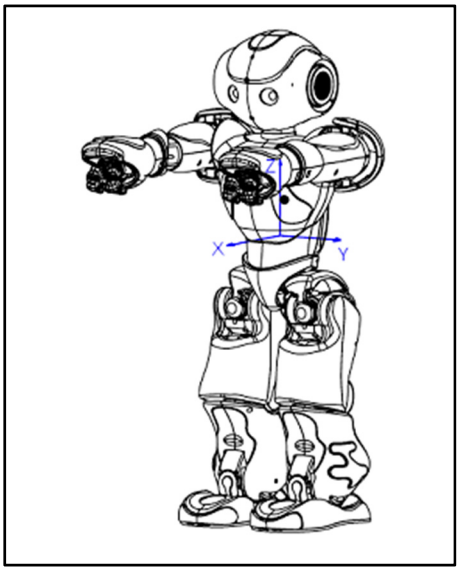

Figure 3.11 Système de coordonnée
et point initial
Tirée de Aldebaran (2013)

Chacune des cinq chaines de moteur pour calculer la cinématique directe est présentée sous forme de tableau. Une chaine pour la tête qui est la plus simple du groupe, comme le montre le tableau 3.3. Une chaine pour représenter chaque bras, le tableau 3.5 et le tableau 3.6. Pour finir, les deux jambes le tableau 3.7 et le tableau 3.8. Chaque chaine a pour point de départ une base qui est le torse et se termine à l'extrémité de la chaine. En se basant sur la figure 3.11 pour avoir le point de référence et l'orientation du système d'axe pour les calculs des tableaux de chaque chaine.

Tableau 3.3 Chaine qui représente la tête pour les paramètres DH.
Adapté de Kofinas (2012, p.30)

Tête	**a**	**α**	**d**	**Θ**
Base	A(0,0,NeckOffsetZ)			
HeadYaw	0	0	0	$\theta 1$
HeadPitch	0	$-\frac{\pi}{2}$	0	$\theta 2 - \pi 2$
Top Camera	A (topCameraX, 0, topCameraZ)			
Bottom Camera	A (bottomCameraX, 0, bottomCameraZ)			

Pour les lignes topCamera en X, Z et bottomCamera en X, Z il faut comprendre que ce sont des mesures fournies par la compagnie. Pour avoir les mesures liées aux distances, il faut se fier au tableau 3.4.

Tableau 3.4 Mesures des distances
Tiré de Aldebaran (2013)

Nom	**Distance**
TopCameraX	53.9 mm
TopCameraZ	67.9 mm
BottomCameraX	48.8 mm
BottomCameraZ	23.8 mm

TopCamera et Bottom Camera correspondent aux deux effecteurs finaux de la chaine de la tête. A_2^{End} est une des deux matrices de transformation pour les deux caméras. Grâce à l'équation 3.4, il est possible de calculer la matrice de transformation T_{Base}^{End}.

$$T_{Base}^{End} = A_{Base}^{0} T_0^1 T_1^2 A_2^{End} \tag{3.4}$$

Tirée de Kofinas (2012, p.30)

Maintenant que la tête est calculée, il faut calculer les deux bras. Cette étape est importante pour la dynamique complète du robot, donc dans le cas d'une estimation ou d'une dynamique

partielle, l'importance des deux bras est donc diminuée. Encore une fois, les bras comportent une petite formalité puisque les deux bras sont symétriques, les matrices seront très similaires. Cependant, la différence va être portée sur le signe qui va changer, car les bras subissent seulement une rotation de 180 degrés l'un par rapport à l'autre. Il est important ici de mentionner que le point initial des bras est à la verticale devant le corps du Nao.

Tableau 3.5 Chaine qui représente le bras gauche pour les paramètres DH
Adapté de Kofinas (2012 p.31)

Bras gauche	**a**	**α**	**d**	**Θ**
Base	A (0, ShoulderOffsetY + ElbowOffsetY, ShoulderOffsetZ)			
LShoulderPitch	0	$-\frac{\pi}{2}$	0	$\theta 1$
LSoulderRoll	0	$\frac{\pi}{2}$	0	$\theta 2 + \frac{\pi}{2}$
LElbowYaw	0	$-\frac{\pi}{2}$	UpperArmLength	$\theta 3$
LElbowRoll	0	$\frac{\pi}{2}$	0	$\theta 4$
Rotation	$R_z(\frac{\pi}{2})$			
End effector	A (HandOffsetX + LowerArmLength, 0, 0)			

$$T_{Base}^{End} = A_{Base}^{0} T_0^1 T_1^2 T_2^3 T_3^4 R_z(\tfrac{\pi}{2}) A_4^{End} \quad (3.5)$$

Tirée de Kofinas (2012, p.31)

Tableau 3.6 Chaine qui représente le bras droit pour les paramètres DH
Adapté de Kofinas (2012 p.32)

Bras Droit	**a**	**α**	**d**	**Θ**
Base	A (0, -ShoulderOffsetY - ElbowOffsetY, ShoulderOffsetZ)			
RShoulderPitch	0	$-\frac{\pi}{2}$	0	$\theta 1$
RSoulderRoll	0	$\frac{\pi}{2}$	0	$\theta 2 + \frac{\pi}{2}$
RElbowYaw	0	$-\frac{\pi}{2}$	- UpperArmLength	$\theta 3$
RElbowRoll	0	$\frac{\pi}{2}$	0	$\theta 4$
Rotation	$R_z(\frac{\pi}{2})$			
End effector	A (HandOffsetX + LowerArmLength, 0, 0)			
Rotation 2	$R_z(-\pi)$			

Comme il a été dit plus tôt, il faut rajouter une rotation de 180 degrés, que nous appliquons sur le bras droit, car le bras est à l'inverse de l'autre. Il aurait été possible ici de l'ajouter sur le bras gauche au lieu du droit et le résultat aurait été pareil.

$$T_{Base}^{End} = A_{Base}^{0} T_0^1 T_1^2 T_2^3 T_3^4 R_z(\tfrac{\pi}{2}) A_4^{End} R_z(-\pi) \tag{3.6}$$

Tirée de Kofinas (2012, p.32)

Pour les chaines des jambes, il y a une grande difficulté, car il y a un angle de 45 degrés qui lie la jambe droite et la jambe gauche au corps du Nao. Donc, pour que le modèle soit précis, il est essentiel de faire une rotation. Il faut aussi ajouter les distances appropriées pour la base et à la fin du pied, car sinon la matrice ne sera pas représentée au point désiré.

Tableau 3.7 Chaine qui représente la jambe gauche pour les paramètres DH
Adapté de Kofinas (2012 p. 33)

Jambe gauche	**a**	**α**	**d**	**Θ**
Base	A(0,HipOffsetY, -HipOffsetZ)			
LHipYawPitch	0	$-\frac{3\pi}{4}$	0	$\theta 1 - \frac{\pi}{2}$
LhipPitch	0	$-\frac{\pi}{2}$	0	$\theta 2 + \frac{\pi}{4}$
LhipRoll	0	$\frac{\pi}{2}$	0	$\theta 3$
LkneePitch	-ThighLength	0	0	$\theta 4$
LanklePitch	-TibiaLength	0	0	$\theta 5$
LankleRoll	0	$-\frac{\pi}{2}$	0	$\theta 6$
Rotation	$R_z(\pi)R_Y(-\frac{\pi}{2})$			
End effector	A(0,0,-footHeight)			

La jambe gauche est donc représentée par l'équation 3.7:

$$T_{Base}^{End} = A_{Base}^{0} T_0^1 T_1^2 T_2^3 T_3^4 T_4^5 T_5^6 R_z(\pi) R_Y(-\tfrac{\pi}{2}) A_6^{End} \quad (3.7)$$

Tirée de Kofinas (2012, p.33)

Tableau 3.8 Chaine qui représente la jambe droite pour les paramètres DH
Adapté de Kofinas (2012 p.33)

Jambe droite	**a**	**α**	**d**	**Θ**
Base	A(0, -HipOffsetY, -HipOffsetZ)			
RHipYawPitch	0	$-\frac{\pi}{4}$	0	$\theta 1 - \frac{\pi}{2}$
RhipPitch	0	$-\frac{\pi}{2}$	0	$\theta 2 - \frac{\pi}{4}$
RhipRoll	0	$\frac{\pi}{2}$	0	$\theta 3$
RkneePitch	-ThighLength	0	0	$\theta 4$
RanklePitch	-TibiaLength	0	0	$\theta 5$
RankleRoll	0	$-\frac{\pi}{2}$	0	$\theta 6$
Rotation	$R_z(\pi)R_Y(-\frac{\pi}{2})$			
End effector	A(0,0,-footHeight)			

Donc, le tableau 3.8 si dessus, nous donne l'équation 3.8 de la matrice de transformation pour la jambe droite de notre Nao.

$$T_{Base}^{End} = A_{Base}^{0} T_0^1 T_1^2 T_2^3 T_3^4 T_4^5 T_5^6 R_z(\pi) R_Y(-\tfrac{\pi}{2}) A_6^{End} \quad (3.8)$$

Tirée de Kofinas (2012, p.33)

3.5 Cinématique inverse

Pour la cinématique inverse, nous prenons le problème à l'inverse. Effectivement, comparativement à la cinématique directe, maintenant nous cherchons à déterminer la variation des joints en fonction de la position et l'orientation de joint final de la chaine.

Un point pour le joint final correspond dans un espace à trois dimensions à une position (Px,Py,Pz) et une orientation (ax, ay, az). Voilà pourquoi il est important de faire la cinématique directe avant l'inverse. Son résultat est une matrice de transformation T (formule) qui inclut, une matrice de rotation, le carré en bleu et une matrice de translation le carré en rouge. La matrice T est la matrice de transformation finale pour le Nao.

$$T = \begin{matrix} \cos a_y \cos a_z & -\cos a_x \sin a_z + \sin a_x \sin a_z \cos a_z & \sin a_x \sin a_z + \cos a_x \sin a_y \cos a_z & p_x \\ \cos a_y \sin a_z & \cos a_x \cos a_z + \sin a_x \sin a_y \sin a_z & -\sin a_x \cos a_x + \cos a_x \sin a_y \sin a_z & p_y \\ -\sin a_y & \sin a_x \cos a_y & \cos a_x \cos a_y & p_z \\ 0 & 0 & 0 & 1 \end{matrix} \quad (3.9)$$

Donc pour la tête,

$$T_{tete}^{torse} = \begin{bmatrix} -\cos\theta 1 \sin\theta_b 2 & -\sin\theta 1 & \cos\theta 1 \cos\theta_b 2 & l2 \cos\theta 1 \cos\theta_b 2 - l1 \cos\theta 1 \sin\theta_b 2 \\ -\sin\theta 1 \sin\theta_b 2 & \cos\theta 1 & \cos\theta_b 2 \sin\theta 1 & l2 \cos\theta_b 2 \sin\theta 1 - l1 \sin\theta 1 \sin\theta_b 2 \\ -\cos\theta_b 2 & 0 & -\sin\theta_b 2 & l3 - l2 \sin\theta_b 2 - l1 \cos\theta_b 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.10)$$

Avec pour l1 = camera X , l2 = camera z et l3 = NeckOffsetZ

La matrice pour le bras gauche est identifiée par l'équation 3.11 et leur valeur par les équations qui suivent.

$$T_{main}^{torse} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.11)$$

$$r_{11} = sin\,\theta4\big(sin\,\theta1\,sin\,\theta3 - cos\,\theta1\,cos\,\hat{\theta}2\,cos\,\theta3\big) - cos\,\theta1\,cos\,\theta4\,sin\,\hat{\theta}2 \quad (3.12)$$

$$r_{12} = sin\,\theta4\big(sin\,\theta1\,sin\,\theta3 - cos\,\theta1\,cos\,\hat{\theta}2\,cos\,\theta3\big) + cos\,\theta1\,sin\,\hat{\theta}2\,sin\,\theta4 \quad (3.13)$$

$$r_{13} = cos\,\theta3\,sin\,\theta1 + cos\,\theta1\,cos\,\hat{\theta}2\,sin\,\theta3 \quad (3.14)$$

$$r_{14} = l4\,(sin\,\theta4\big(sin\,\theta1\,sin\,\theta3 - cos\,\theta1\,cos\,\hat{\theta}2\,cos\,\theta3\big) - l3\,cos\,\theta1\,sin\,\hat{\theta}2 \quad (3.15)$$

$$r_{21} = cos\,\hat{\theta}2\,cos\,\theta4 - cos\,\theta3\,sin\,\hat{\theta}2\,sin\,\theta4 \quad (3.16)$$

$$r_{22} = cos\,\hat{\theta}2\,sin\,\theta4 - cos\,\theta3\,cos\,\theta4\,sin\,\hat{\theta}2 \quad (3.17)$$

$$r_{23} = sin\,\hat{\theta}2\,sin\,\theta3 \quad (3.18)$$

$$r_{24} = l1 + l3cos\hat{\theta}2 + l4(cos\,\hat{\theta}2\,cos\,\theta4 - cos\,\hat{\theta}3\,sin\,\hat{\theta}2\,sin\,\theta4) \quad (3.19)$$

$$r_{31} = sin\,\theta4\big(cos\,\theta1\,sin\,\hat{\theta}3 - cos\,\theta1\,cos\,\hat{\theta}2\,cos\,\theta3\big) + cos\,\theta3\,sin\,\theta2\,sin\,\hat{\theta}4 \quad (3.20)$$

$$r_{32} = sin\,\theta4\big(cos\,\theta1\,sin\,\hat{\theta}3 - cos\,\theta1\,cos\,\hat{\theta}2\,cos\,\theta3\big) - sin\,\theta_1\,sin\,\hat{\theta}_2\,sin\,\theta_4 \quad (3.21)$$

$$r_{33} = cos\,\theta_1\,cos\,\hat{\theta}_3 - cos\,\hat{\theta}_2\,sin\,\theta_1\,sin\,\hat{\theta}_3 \quad (3.22)$$

$$r_{34} = l_2 + l_4\big(sin\,\theta_4\,\big(cos\,\theta_1\,sin\,\hat{\theta}_3 + cos\,\hat{\theta}_2\,cos\,\hat{\theta}_3\,sin\,\theta_1\big) + cos\,\theta_4\,sin\,\theta_1\,sin\,\hat{\theta}_2\big) + l_3\,sin\,\theta_1\,sin\,\hat{\theta}_2 \quad (3.23)$$

Pour le bras droit, il est symétrique au bras gauche, mais il y a une différence qui se reflète dans le symbole de la partie translation.

Donc la translation est représentée par r_{14}, r_{24} et r_{34} comme présenté plus bas.

$$r_{14} = l4\,(sin\,\theta4\big(sin\,\theta1\,sin\,\theta3 - cos\,\theta1\,cos\,\hat{\theta}2\,cos\,\theta3\big) + l3\,cos\,\theta1\,sin\,\hat{\theta}2 \quad (3.24)$$

$$r_{24} = -l1 - l3cos\hat{\theta}2 - l4(cos\,\hat{\theta}2\,cos\,\theta4 - cos\,\hat{\theta}3\,sin\,\hat{\theta}2\,sin\,\theta4) \quad (3.25)$$

$$r_{34} = l_2 - l_4\big(sin\,\theta_4\,\big(cos\,\theta_1\,sin\,\hat{\theta}_3 + cos\,\hat{\theta}_2\,cos\,\hat{\theta}_3\,sin\,\theta_1\big) + cos\,\theta_4\,sin\,\theta_1\,sin\,\hat{\theta}_2\big) - l_3\,sin\,\theta_1\,sin\,\hat{\theta}_2 \quad (3.26)$$

Les formules suivantes, de 3.27 à 3.36, sont ce que les calculs des matrices des jambes devraient donner, la raison pour qu'il y ait seulement le résultat c'est dû à la complexité des calculs et à leur grandeur. Donc pour le détail des jambes, il faut se référer aux travaux de Nikos Kofinas [28], un collègue de la RoboCup originaire de Grèce.

$$T' = \left(R_x\left(\frac{\pi}{2}\right)((A^0_{Base})^{-1}T(A^0_{Base})^{-1})\right)^{-1} \tag{3.27}$$

$$\theta 4 = \pm\left(\pi - cos^{-1}\left(\frac{l1^2+l2^2-\sqrt{\left(0-T'_{(1,4)}\right)^2+\left(0-T'_{(2,4)}\right)^2+\left(0-T'_{(3,4)}\right)^2}}{2*l1*l2}\right)\right) \tag{3.28}$$

$$\theta 6 = tan^{-1}\left(\frac{T'_{(2,4)}}{T'_{(3,4)}}\right) \rightarrow if\ (l2\ cos(\theta 5) + l1\ cos(\theta 4 + \theta 5))\ \neq 0 \tag{3.29}$$

$$\theta 6 = 0 \rightarrow if\ (l2\ cos(\theta 5) + l1\ cos(\theta 4 + \theta 5)) = 0 \tag{3.30}$$

$$T'' = \left((T')^{-1}\left(T_5^6 * R_z(\pi)R_y\left(-\frac{\pi}{2}\right)\right)^{-1}\right)^{-1} \tag{3.31}$$

$$\theta 5 = \pi - sin^{-1}\left(-\frac{T''_{(2,4)}(l2+l1*cos(\theta 4))+l1*T''_{(1,4)}\,sin(\theta 4)}{l1^2\,sin(\theta 4)^2+(l2+l1*cos(\theta 4))}\right) \tag{3.32}$$

$$T''' = (T'')^{-1}(T_3^4T_4^5)^{-1} \tag{3.33}$$

$$\theta 2 = \pm\, cos^{-1}\left(T'''_{(2,3)}\right) - \frac{\pi}{4} \tag{3.34}$$

$$\theta 3 = \pi - sin^{-1}\left(\frac{T'''_{(2,2)}}{sin\left(\theta 2+\frac{\pi}{4}\right)}\right) \tag{3.35}$$

$$\theta 1 = \pm\, cos^{-1}\left(\frac{T'''_{(1,3)}}{sin\left(\theta 2+\frac{\pi}{4}\right)}\right) + \frac{\pi}{2} \tag{3.36}$$

3.6 Conclusion

Grâce à la cinématique directe, il est possible de trouver la cinématique inverse et maintenant que le modèle est complet et validé il est possible de travailler avec une position. En donnant une position au robot, il va être en mesure de calculer l'angle que les joints vont prendre pour se rendre à cette position. Bien sûr, il y a plusieurs solutions possibles et pour cela, lors de l'intégration il faut imposer des limites au système pour que les joints ne prennent pas des positions impossibles et brisent ses moteurs durant la manœuvre.

Dans ce chapitre, les résultats sont les mêmes que Nikos et c'était ce qui devait être validé. Nikos présente l'approche pour calculer la dynamique sur le V4 et avec son approche, nous voulions garantir que les résultats de la dynamique utilisés pour le V5 soient les mêmes que le V4. Cela a été possible grâce aux détails fournis par le fournisseur.

CHAPITRE 4

ALGORITHME DU ZÉRO MOMENT POINT (ZMP)

L'expression ZMP, signifiant « Zéro moment point » est utilisée fréquemment en robotique. Cette expression identifie une méthode pour stabiliser un système en déplacement. Depuis que Miomir K. Vukobratović a présenté ce concept dans son article [29] en 1972, ce sujet aujourd'hui a été travaillé et utilisé par plusieurs chercheurs dans le monde.

4.1 Zéro moment point (ZMP)

L'objectif du ZMP est de garder le robot en équilibre. De ce principe, plusieurs recherches ont été effectuées sur le ZMP. La méthode du ZMP se base sur les équations de Newton-Euler[30] qui régissent tout système en mouvement dans l'espace. Plusieurs articles scientifiques dépendent de l'algorithme du ZMP qui remplace un PID et permet une meilleure robustesse sur la stabilité du robot. Lors de la marche du robot, le concept du ZMP permet d'avoir un contrôle sur le balancement du robot bipède en lui calculant une zone de sécurité lors de son balancement et donc pouvant effectuer des modifications. Le ZMP est un outil efficace de contrôle et la fonction de compensation de la gravité permet une marche et une manipulation d'objet stable en toute situation. Ce principe est illustré à la figure 4.1 où on voit l'évolution du concept et le résultat final dans la partie c. Les carrés noirs dans la figure représentent les pieds du robot. L'ensemble des points sont des candidats potentiels résultant du ZMP. Un d'entre eux sera la cible du « *Center of Mass* » (COM) pour arriver à un ZMP. La ligne bleue qui passe d'un carré (le pied du robot) à l'autre permet de comprendre le balancement du centre de masse, essayant d'atteindre la petite croix verte dans chacun des carrés.

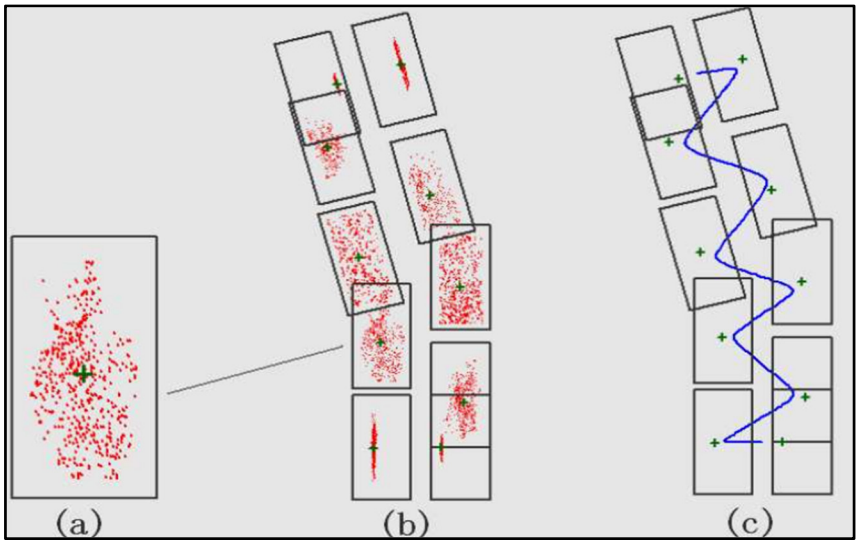

Figure 4.1 Représentation du ZMP et la trajectoire du COM
Tirée de Jinsu & Manuela (2008, p.5)

La figure 4.2 nous illustre le balancement du robot en situation réelle, la figure représente un déplacement du robot en ligne droite et se lit de droite à gauche. Cependant, il reste à comprendre comment et pourquoi le ZMP est important pour améliorer la marche du robot.

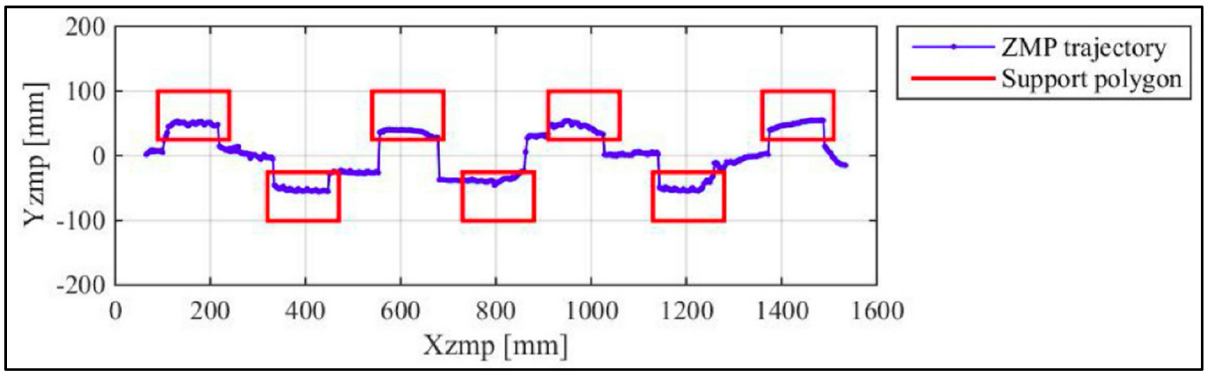

Figure 4.2 Trajectoire du ZMP
Tirée de Zang, Liu, Li, Lin &Zhao (2017, p. 13)

À la base, il faut prendre un point de référence situé sur le robot. Habituellement, ce point est mathématiquement défini grâce au COM du robot. Pour arriver à trouver ce point, il faut identifier une position qui sera la position du robot au repos. De façon générale, le point sera

en haut des hanches, car pour un robot humanoïde c'est habituellement un point central et dans la figure 4.1 comme dans la figure 4.2, il est représenté par la ligne bleue. L'objectif est de garder le COM identifier en tout temps dans une zone qui est définie sécuritaire. Cette zone est identifiée à l'intérieur des carrés rouges dans la figure 4.2, soit ici les pieds du robot. Cette zone est définie sécuritaire si elle répond aux critères imposés par le ZMP. Le plus important est que dans cette zone, le COM a un moment inertiel qui est égal à zéro, ou il en est très prêt. Donc si le point de référence sort de la zone de sécurité alors le robot va tomber au sol lors de son mouvement, car le COM aura une force trop grande, il va perde l'équilibre. Notre objectif est que le robot ne tombe pas, alors il est important de calculer de façon précise notre point de référence et de délimiter notre zone de sécurité. La zone de sécurité se calcule avec les deux formules mathématiques présenté en 4.1 et 4.2 si dessous: [30].

$$x_{zmp} = \frac{\sum_{i=1}^{n} m_i(\ddot{p}_{iz}+g)p_{ix}+\sum_{i=1}^{n} m_i\ddot{p}_{ix}p_{iz}}{\sum_{i=1}^{n} m_i(\ddot{p}_{iz}+g)} \quad (4.1)$$

x_{zmp} est le vecteur face au Nao. Il est un point en avant ou en arrière du robot.

$$y_{zmp} = \frac{\sum_{i=1}^{n} m_i(\ddot{p}_{iz}+g)p_{iy}+\sum_{i=1}^{n} m_i\ddot{p}_{iy}p_{iz}}{\sum_{i=1}^{n} m_i(\ddot{p}_{iz}+g)} \quad (4.2)$$

y_{zmp} est un vecteur perpendiculaire, donc à droite ou à gauche du Nao. Avec un p_{ix}, p_{iy} et p_{iz} représentant le COM des liens, m_i est la masse des mêmes liens, et le g est la constante gravitationnelle.

Lors d'un mouvement de la jambe, un déplacement mesuré par les angles des moteurs permettra de générer une erreur lorsque comparé avec la mesure désirée, cette erreur est visible sur la figure 4.2. La ligne bleue de la figure 4,2 représente cette erreur, car la ligne n'est pas constante et oscille. Cette erreur servira de mesure d'ajustement dans le système afin de garder notre point de référence dans la nouvelle zone de sécurité. Pour comprendre ce concept, il faut regarder la ligne bleue de la figure 4.2. Au point initial le robot est stable et

en équilibre. Le robot commence sa marche et donc il lève un pied, la ligne bleue va donc bouger vers le pied qui supporte encore le robot. Cette opération, mathématiquement, est très couteuse et demanderait beaucoup de temps de calcul pour arriver à un résultat d'une grande précision. La raison est qu'entre le point de départ et le point d'arrivée, il y a une multitude de points pour garder l'exactitude. Pour simplifier le tout, il est recommandé d'utiliser une méthode d'estimation. La méthode suggérée ici est la régression linéaire [33] pour arriver à une approximation de la trajectoire complète. L'équation 4.3 est la forme générale de la formule de linéarisation.

$$y = a_n x^n + a_n - 1x^{n-1} + \cdots + a_1 x + a_0 \tag{4.3}$$

Cette méthode robuste qui a fait ses preuves est l'outil de contrôle utilisé également par le club Naova. Cet algorithme de contrôle permet à nos robots d'être plus stables sur la surface de jeu. La surface de jeu est un tapis de gazon artificiel qui est très différent d'un plancher de bois conventionnel. Elle permet une marche plus rapide et également une meilleure stabilité lors des coups de pied sur le ballon. N'étant jamais seul sur le terrain, mais avec une autre équipe qui a pour but aussi de gagner, il est fréquent que les robots entrent en contact, ce qui occasionne un déséquilibre du système. Avec ce facteur non prévisible, si les robots n'ont pas des algorithmes robustes pour la stabilité, la probabilité de chute devient significative. C'est à ce moment-là que le ZMP permet à nos robots d'avoir une meilleure chance de rester debout comparativement à un adversaire sans cet atout. Une méthode robuste est essentielle pour permettre à nos robots de jouer au soccer. Car chaque chute évitée protège nos robots, mais aussi fait sauver d'importante secondes dans l'action. Une partie de soccer est très dynamique et donc la marche de notre joueur est omnidirectionnelle pour répondre au maximum à l'environnement très changeant. Pour répondre à ces besoins, il faut déplacer le centre de masse et donner aux joints des jambes les bons angles pour pouvoir avancer. Pour calculer et générer la trajectoire du robot, nous utilisons un module de prédiction de trajectoire. Pour simplifier le processus de la cinématique inverse la méthode du pendule inversé est la stratégie de linéarisation du modèle qui est utilisée sur le système comme

l'illustre la figure 4.3. Ayant un centre de masse fixe, qui se trouve directement sur le torse du Nao, l'application du pendule inversé se visualise bien.

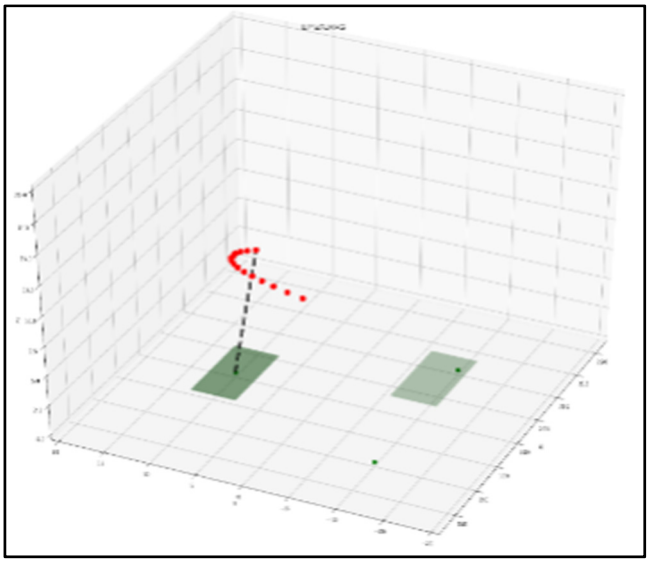

Figure 4.3 Le pendule inversé
appliquer pour le Nao

4.2 Le contrôle

La force des modules vient de son interaction rapide avec les autres modules pour arriver à un résultat désiré. Le schéma de contrôle de la figure 4.4 montre de façon simplifiée l'interaction dans une boucle fermée des différents modules impliqués pour le ZMP dans un Nao.

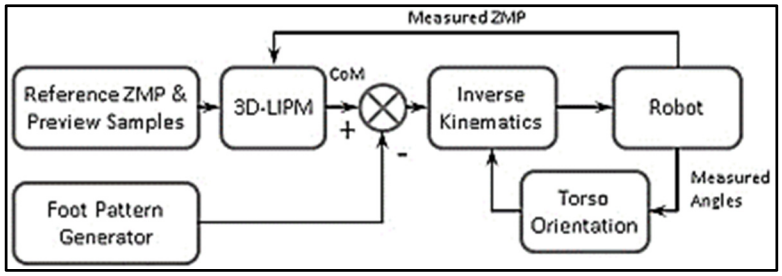

Figure 4.4 Schéma de contrôle du Nao avec le ZMP.
Tirée de Hashemi & Khajepour (2016, p. 61)

Tout commence avec des valeurs d'entrées dans le système. Un point de référence et un module de marche sont un bon point de départ. Il faut avoir la position actuelle et savoir comment marcher pour avoir une chance de bouger. Pour le mouvement, le « *3 Dimensionnal Linear inverted Pendulum Mode* » (3D LIPM) est utilisé, car il donne une estimation de la représentation physique du centre de masse (COM). Par addition, selon l'équipe de B-Human [35] la position et la vitesse du centre de masse sont calculables par les équations présenté en 4.4 et 4.5 :

$$x(t) = x_0 * cosh(k * t) + \dot{x}_0 * \frac{1}{k} * sinh(k * t) \quad (4.4)$$

$$\dot{x}(t) = x_0 * k * sinh(k * t) + \dot{x}_0 * cosh(k * t) \quad (4.5)$$

Avec $k = \sqrt{\frac{g}{h}}$, g est l'accélération gravitationnelle ($\approx 9{,}81\frac{m}{s^2}$), h étant la hauteur parallèle au sol, $x_0 \in \mathbb{R}^2$ est la position du centre de masse relative à l'origine du pendule inversé à t=0. $\dot{x}_0 \in \mathbb{R}^2$ est la vitesse du centre de masse à t=0.

En combinaison avec la cinématique inverse, la trajectoire peut être générée, cependant pour une diminution de l'erreur et en prenant en considération un environnement dynamique, un retour d'information est requis du ZMP permettant les corrections adéquates pour une meilleure stabilité. Une mesure des angles est déterminée et le torse se met donc en mouvement. Celui-ci est en accord avec le ZMP qui lui est dans le 3D LIPM afin de générer la trajectoire du COM comme représenter à la figure 4.4.

La stabilité du robot passe avant tout par sa compétence à bien gérer son centre de gravité. Pour ce faire, le robot utilise différentes méthodes de contrôle pour recevoir des retours d'informations sur la situation actuelle. Ces informations sont analysées et le robot applique le résultat mathématique en effectuant les changements appropriés. L'une de ces méthodes est le « Zéro moment point » (ZMP) qui est utilisé dans la chaine de commande. Le module associé au ZMP se nomme « ZMPBalance », il permet le balancement basé sur la méthode

robuste du ZMP et l'évitement des chutes. De façon plus précise, le fonctionnement du module est identifié par la figure 4.5. Ce module permet, en recevant les mesures des données en sortie de calculer les angles désirés pour la marche.

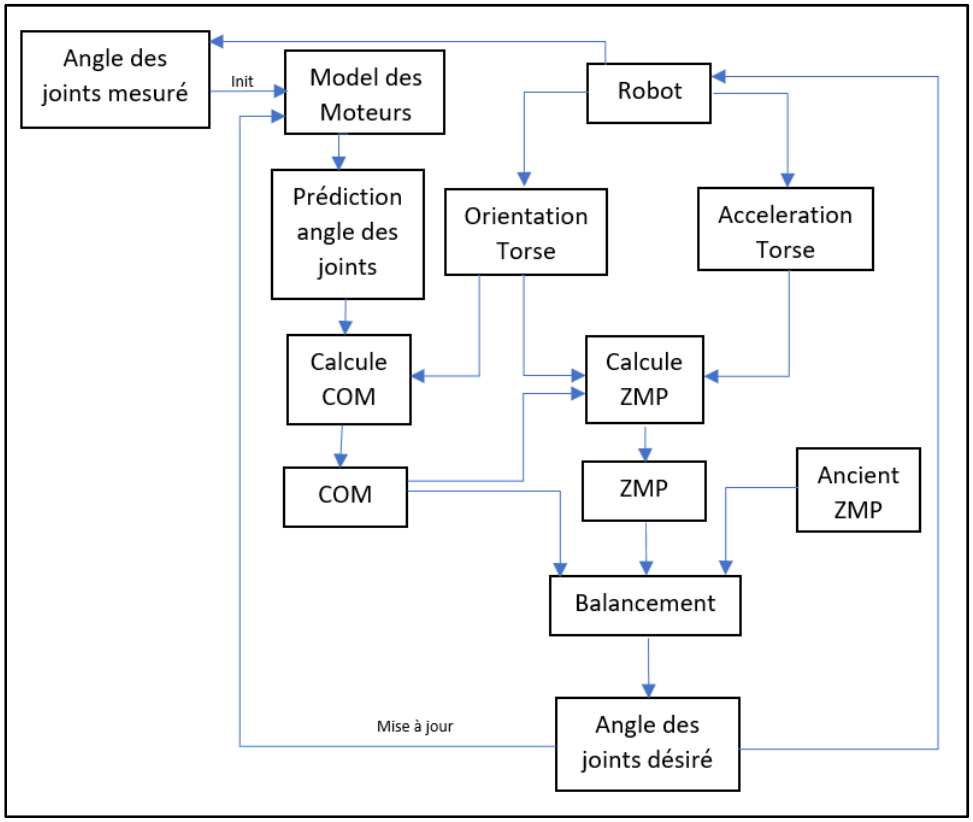

Figure 4.5 Fonctionnement du module ZMPBalance

4.3 Génération d'un pas

La dynamique commence avec l'objectif d'atteindre une position B partant d'un point de départ A. Le robot commence avec les deux pieds sur le sol, il a donc deux supports et le COM est centré entre les deux pour rester stable. Pour quitter le sol, une jambe va entrer dans la phase portant le nom « préswing phase ». C'est la phase de propulsion de la jambe vers le haut pour atteindre un point identifier au préalable. Lorsque le pied quitte le sol, c'est la phase « single support phase » qui commence. Il y a un seul support et donc le COM va se

déplacer pour permettre au robot de rester stable. Il va donc se rapprocher du point de support. Une fois stable, la jambe amorce la troisième phase « postswing phase ». La jambe va redescendre vers le sol et le COM va se déplacer jusqu'à ce que le robot ait deux points d'appui au sol et le cycle recommence. Ce processus va se répéter pour l'autre jambe et les phases vont se succéder jusqu'à la destination.

La partie importante est de générer la référence pour le COM grâce au ZMP, pour chaque pas désiré il faut convertir la séquence en valeur. La séquence de marche est divisée en quatre phases [30]. Quand le robot est en phase 1, appelé aussi la phase double, le ZMP va rester dans le centre des pieds et quand il passe en phase simple, qui fait référence à la phase 3, il faut que le ZMP ait communiqué la position au COM et qu'il se déplace au point du prochain pas avant que le pied quitte le sol et commence à avancer sa jambe. La figure 4.6 montre la phase initiale avec le COM, le X vert, au centre des deux pieds. La figure 4.7 a et b montre le X du COM, pendant les phases de la marche, qui tente de rejoindre, au maximum, le X droit ou gauche à l'intérieur du pied. Pour faciliter ce changement, une matrice de transformation est utilisée pour la mise à jour des prochains pas. Les phases 2 et 4 sont des phases de transition, la 2 est quand le pied commence à quitter le sol et la phase 4 est quand le pied commence à descendre.

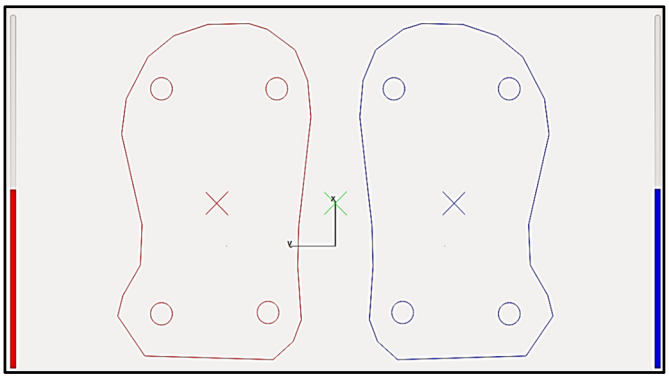

Figure 4.6 Représentation de la phase 1

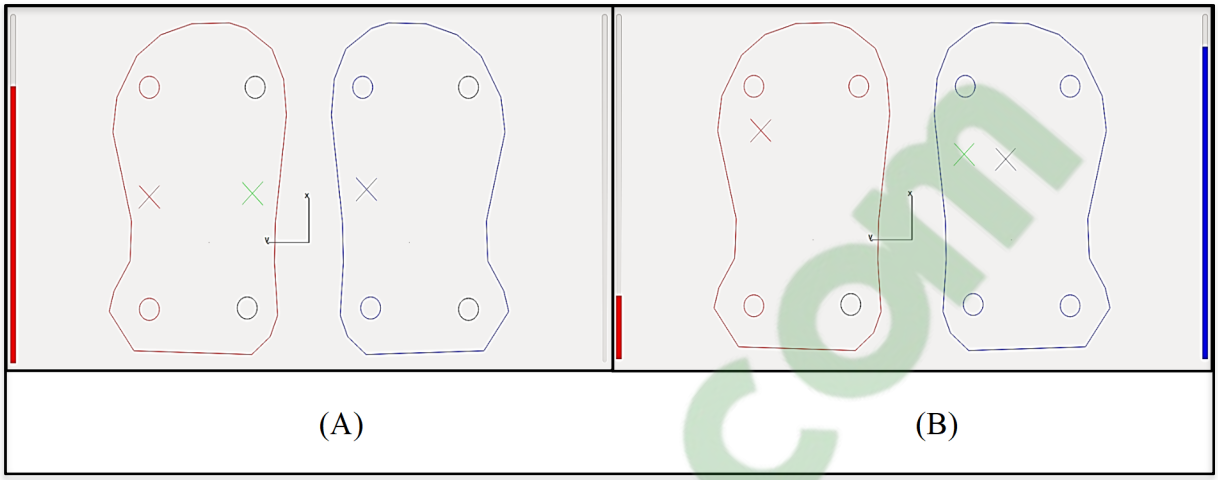

Figure 4.7 A) et B) Représentation du déplacement du COM pendant la marche

La figure 4.8 illustre le déplacement en y du centre de masse (COM) durant une période de 15 secondes. Cette séquence montre le robot se positionner vers la balle et tente un gros dégagement pour essayer de repousser la balle le plus loin possible. Le déplacement n'est pas constant, car le robot tourne sur lui-même et se place lentement pour être en mesure de frapper la balle dans la direction désirée.

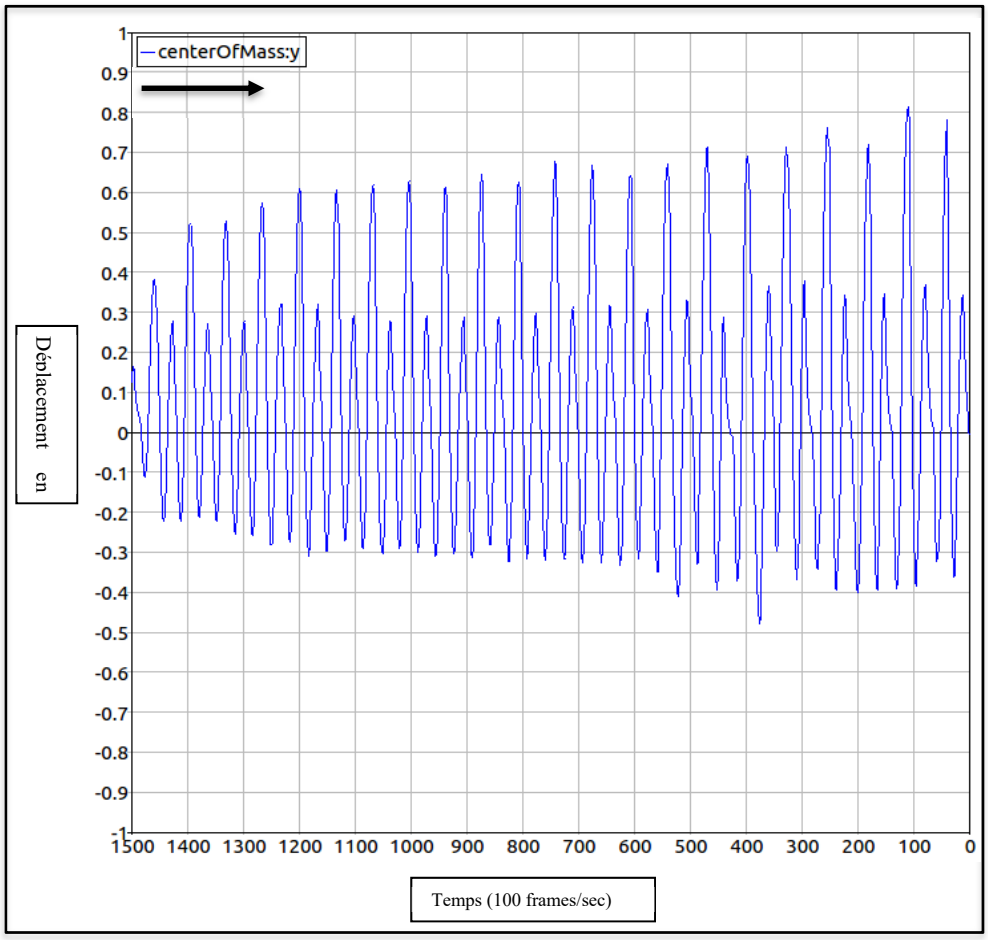

Figure 4.8 Déplacement du centre de masse du Nao

4.4 Conclusion

Nos joueurs, sans le ZMP [6] et en utilisant la méthode proposée par la compagnie de fabrication, peuvent atteindre une vitesse de 10,5 cm/s mais le robot semblait instable et tombait souvent sur une surface différente que le plancher rigide et plat même en l'absence de pente et d'obstacle. Pour cela, la vitesse optimale était de 7 cm/s, qui reste une vitesse trop lente pour rivaliser. Maintenant avec le ZMP intégrer, il est possible d'avoir une vitesse de 20 cm/s à 25 cm/s avec le robot. En simulation, la vitesse peut atteindre 31 cm/s. Certes le robot semble plus stable, mais bien que nous approchions de l'objectif d'atteindre une vitesse de 350 mm/s cela reste insuffisant. Cela n'ait que le début, car au prochain chapitre, un autre algorithme sera ajouté pour voir si on peut allez encore plus vite.

CHAPITRE 5

ALGORITHME DE MARCHE ROBUSTE ET STABLE

La plupart des robots commerciaux utilisent des servomoteurs DC avec un contrôleur intégrés. Généralement, le contrôleur est conçu pour une approche linéaire bien connue, le PID (Proportionnel-Integral-Dérivé). Cela permet alors de travailler en communiquant seulement une position aux moteurs. Cela ne serait pas un problème avec un model linéaire, mais l'objectif est de donner plus de robustesse à la marche du Nao. Une approche permettant d'envoyer un couple (torque) ou un courant est une option pour atteindre l'objectif principal. Pour cette raison, une toute autre approche de contrôle dynamique non linéaire basée sur un modèle robuste sera présentée dans ce chapitre. L'algorithme proposé répond à l'objectif de développer une meilleure approche pour la marche du robot bipède Nao.

5.1 La méthode

Pour avoir une marche omnidirectionnelle, la méthode actuelle est basée sur un ZMP classique et un contrôleur basé sur la méthode du « three-dimensional Linear inverted pendulum » (3D-LIPM) qui génère la position des joints désirés pour les jambes. Cette position désirée est reçue par le PID des servomoteurs. Cet algorithme est donc stable, mais n'offre pas la robustesse requise si l'environnement est changeant ou que le robot reçoit des forces externes sur lui.

Comme solution, il est proposé d'utiliser une méthode non linéaire capable de gérer cette problématique. Cette méthode est basée sur l'estimation du délai de temps (EDT), « Time Delay estimation » (TDE) en anglais, et sur la méthode du « Sliding mode control » (SMC). Ces méthodes donnent la possibilité d'envoyer au moteur un couple au lieu d'un position et ainsi gagner en efficacité. Pour cela, un module a été implémenté pour recevoir en entrée un couple et d'en sortir une position pour que le PID associé au servomoteur soit en mesure de

fonctionner correctement. Cette méthode créera l'équivalent d'un anti-PID pour nous permettre de travailler en torque. L'anti-PID sera tout simplement une image inverse de la dynamique du Nao. Le module présenté à la figure 5.1 montre comment en entrée il va recevoir la trajectoire désirée, il va la convertir en couple et la retransformer en commande de position de contrôle avant de l'envoyer au moteur.

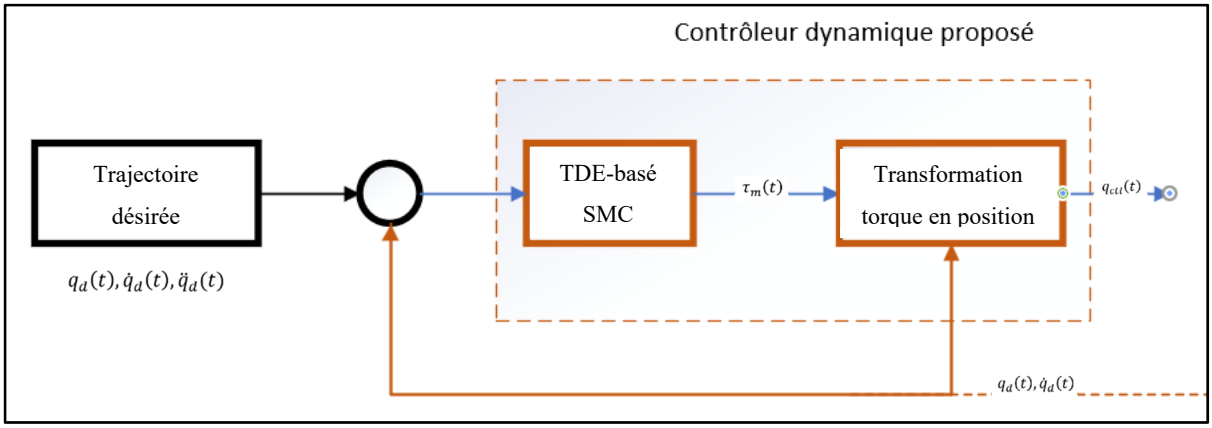

Figure 5.1 Proposition du module de contrôle dynamique

Dans la figure 5.1, les variables sont représenté comme suit :
$\ddot{q}(t), \dot{q}(t), q(t) \in \mathbb{R}^{12}$ sont respectivement le vecteur d'accélération des joints, la vitesse et la position; $\tau_m(t) \in \mathbb{R}^{12}$ est le vecteur de couple (torque) des moteurs et $\mathrm{q_{ctl}(t)}$ en sortie sous la forme d'une position désirée. Sans le module proposé, une commande de position désirée est directement envoyée au moteur du robot comme le montre la figure 5.2. Le module proposé implique donc une étape de plus, mais également une meilleure précision sans pour autant nuire aux performances du CPU.

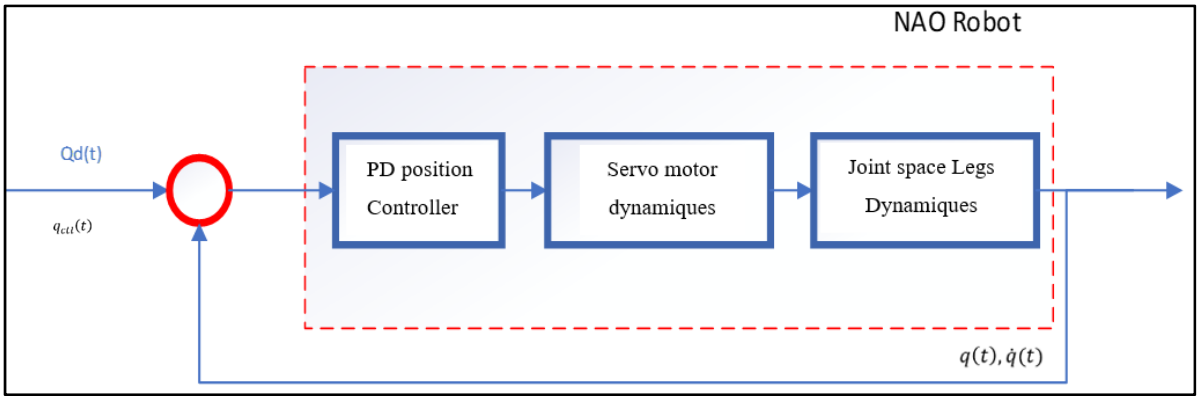

Figure 5.2 Module actuel pour fonctionnement des moteurs dans le robot Nao

Le module proposé est intégré directement dans l'architecture actuelle du Nao. Comme sur la figure 5.3, il va être la dernière étape juste avant de transmettre au robot.

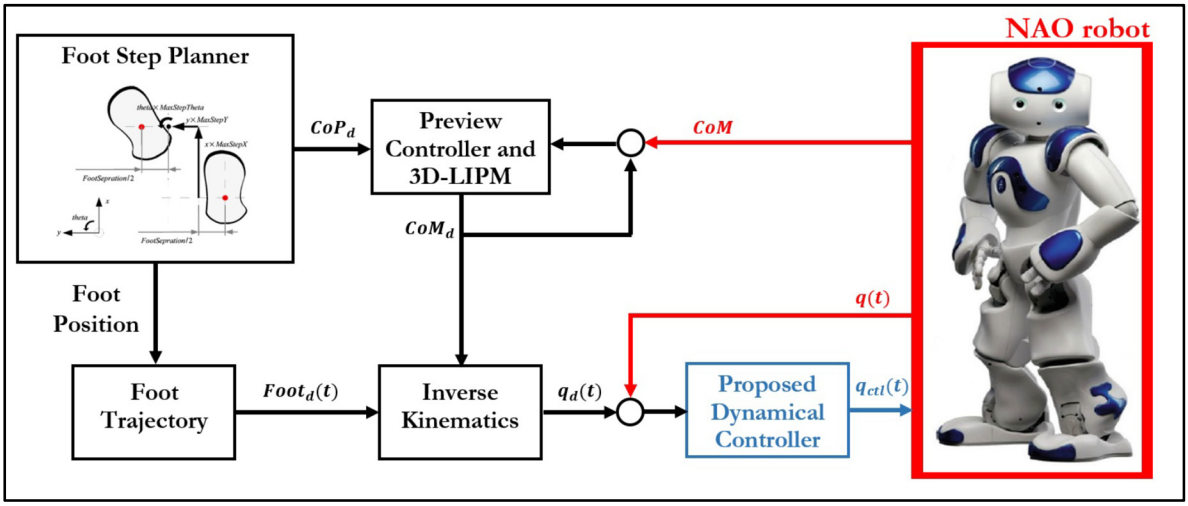

Figure 5.3 Schéma complet de l'architecture

5.2 La solution préliminaire

Pour la solution, le modèle prendra seulement les jambes du Nao, la partie inférieure jusqu'au joint du bassin avant le torse. Cela correspond à une combinaison de 12 degrés de liberté. La dynamique du Nao peut être représentée par l'équation suivante :

$$A\big(q(t)\big)\ddot{q}(t) + B\big(q(t),\dot{q}(t)\big)\dot{q}(t) + G\big(q(t)\big) = \tau(T) \tag{5.1}$$

Où :

$\ddot{q}(t), \dot{q}(t), q(t) \in \mathbb{R}^{12}$ sont respectivement le vecteur d'accélération des joints, la vitesse et la position;

$A\big(q(t)\big) \in \mathbb{R}^{12\times12}$ est la matrice d'inertie;

$B\big(q(t),\dot{q}(t)\big) \in \mathbb{R}^{12\times12}$ est la matrice de centrifuge et la force de Coriolis;

$G\big(q(t)\big) \in \mathbb{R}^{12}$ est le vecteur de force gravitationnelle;

$\tau(T) \in \mathbb{R}^{12}$ est le vecteur de torque.

Tous les actuateurs étant actionnés par un servo moteur dc, la dynamique des actuateurs peut être décrite ainsi :

$$I_M\ddot{q}_M(t) + f_M\dot{q}_M(t) = \tau_m(t) - R_\tau(t) \tag{5.2}$$

$\ddot{q}_M(t) = \ddot{q}(t), \dot{q}_M(t) = \dot{q}(t) \in \mathbb{R}^{12}$ sont respectivement les vecteurs de l'accélération et de la vitesse des moteurs des joints;

$I_M = diag(I_{M1}, \dots, I_{M12})\ et\ f_M = diag(f_{M1}, \dots, f_{M12})$ sont la matrice du moment d'inertie et la matrice du coefficient de friction;

$\tau_m(t) \in \mathbb{R}^{12}$ est le vecteur de torque des moteurs;

$R = diag(R_1, \dots, R_{12})$ c'est le ratio de réduction des engrenages

Une fois combinée l'équation 5.1 avec l'équation 5.2 on obtient :

$$A0\big(q(t)\big)\ddot{q}(t) + F\big(q(t),\dot{q}(t)\big) = \tau_M(t) \tag{5.3}$$

Où :

$\mathrm{A0}\big(\mathrm{q(t)}\big) = RA\big(\mathrm{q(t)}\big) + I_M$ et $\mathrm{F}\big(\mathrm{q(t)},\dot{q}(\mathrm{t})\big) = RB\big(\mathrm{q(t)},\dot{q}(\mathrm{t})\big)\dot{q}(\mathrm{t}) + RG\big(\mathrm{q(t)}\big) + f_M\dot{q(t)}$.

En introduisant la matrice diagonale $\bar{\mathrm{A}} \in \mathbb{R}^{12\times12}$ avec des constantes strictement positives, il est possible de réécrire la dynamique des moteurs combinée du robot Nao sous la forme suivante :

$$\bar{A}\ddot{q}(t) \; + \; H(q(t), \dot{q}(t), \ddot{q}(t)) \; = \; \tau_M(t) \tag{5.4}$$

Où $\mathrm{H}\big(\mathrm{q(t)}, \dot{\mathrm{q}}\mathrm{(t)}, \ddot{\mathrm{q}}\mathrm{(t)}\big) = \; \big[\mathrm{A0}\big(\mathrm{q(t)}\big) - \bar{\mathrm{A}}\big]\ddot{\mathrm{q}}\mathrm{(t)} + \; \mathrm{F}\big(\mathrm{q(t)}, \dot{q}\mathrm{(t)}\big)$.

Laissant $q_d(t) \in \mathbb{R}^{12}$ être le vecteur de la position désirée pour une trajectoire et laissant $\tilde{q}(t) = q(t) - q_t(t) \in \; \mathbb{R}^{12}$ être le vecteur utilisé pour repérer l'erreur.
Le but du contrôle est de développer un contrôleur robuste et stable utilisant la méthode « TDE » basée sur la méthode « SMC ». Par après, il faut transformer le torque calculé en commande de position par la création d'une image (inverse) du contrôleur PD existant actuellement dans le robot Nao.

5.3 Le design du Contrôleur

À partir de maintenant, le contrôleur va commencer à être de plus en plus précis. Il va tranquillement prendre forme.

5.3.1 Estimation de la dynamique

Dans cette section, la méthode TDE va être utilisée pour faire une approximation de la dynamique en assumant qu'elle est inconnue. De façon simple, l'idée est d'éviter d'avoir à faire de gros calcul, car la dynamique du Nao est très complexe. Pour ce faire, il faut assumer que les éléments du vecteur $H\big(q(t), \dot{q}(t), \ddot{q}(t)\big)$ sont la fonction de Lipschitz. Maintenant, le vecteur peut être estimé utilisant la méthode TDE [36],[37] comme suit :

$$\begin{aligned} \widehat{H}\big(q(t), \dot{q}(t), \ddot{q}(t)\big) &\cong H\big(q(t - \Delta t), \dot{q}(t - \Delta t), \ddot{q}(t - \Delta t)\big) \\ &= \; \tau_M(t - \Delta t) - \; \bar{A}\ddot{q}(t - \Delta t) \end{aligned} \tag{5.5}$$

Où :

Δt représente le délai, choisi la plupart du temps pour être égal à la période d'échantillonnage.

Pour faire l'estimation du vecteur $H\big(q(t), \dot{q}(t), \ddot{q}(t)\big)$ la mesure de l'accélération des joints est requise. Malheureusement, seulement la mesure de la position est fournie. Cependant, l'utilisateur est en mesure d'avoir accès à la position et la vitesse en retour d'information. Il devient donc possible de calculer une accélération retardée avec une des deux équations suivantes :

$$\dot{q}(t-\Delta t) = \begin{cases} 0, & if\ t < \Delta t \\ \frac{1}{\Delta t}[\dot{q}(t-\Delta t) - \dot{q}(t-2\Delta t)], & if\ t \geq \Delta t \end{cases} \tag{5.6}$$

$$\ddot{q}(t-\Delta t) = \begin{cases} 0, & if\ t < \Delta t \\ \frac{1}{\Delta t^2}[q(t-\Delta t) - q(t-2\Delta t) + q(t-3\Delta t)], & if\ t \geq \Delta t \end{cases} \tag{5.7}$$

5.3.2 Sliding mode

Pour voir une étude complète sur la théorie du SMC avec le TDC, il est possible de consulter l'article de Kali et al.[37]. La première étape concerne l'identification de la surface de glissement, le choix conventionnel pour un système de second ordre est donné par l'équation suivante :

$$\sigma(t) = \dot{\tilde{q}}(t) + \beta\tilde{q}(t) \tag{5.8}$$

Où :

β peut être choisi comme étant une constante positive.

En second lieu, la loi de contrôle est dérivée pour garantir une robustesse dans un temps fini par la résolution de l'équation suivante :

$$\dot{\sigma}(t) = -K_1\big(\sigma(t)\big)\sigma(t) - K_2 D\big(\sigma(t)\big) sign(\sigma(t)) \tag{5.9}$$

Où :

$K_1\big(\sigma(t)\big) = diag\ (K_{1i}\frac{\sigma_0}{\sigma_0+|\sigma_i(t)|})$ avec $\sigma_{0>0}$ et $K_{1i} > 0$ pour i = 1, … , 12;

$K_2 = diag(K_{2i})$ et $K_{2i} > 0$ pour i = 1, … , 12;

$D\big(\sigma(t)\big) = diag(\frac{1}{d_i+(1-d_i)exp^{-|\sigma_i|^{l_i}}})$ avec $0 < d_i < 1\ et\ l_i > 0$ pour i = 1, … , 12;

$sign\big(\sigma_i(t)\big) = \big[sign\big(\sigma_1(t)\big), \dots, sign\big(\sigma_{12}(t)\big)\big]^T$ avec $sign\big(\sigma_i(t)\big)$ défini par :

$$sign\big(\sigma_i(t)\big) = \begin{cases} -1, & if\ \sigma_i(t) < 0 \\ 0, & if\ \sigma_i(t) = 0 \\ 1, & if\ \sigma_i(t) > 0 \end{cases} \tag{5.10}$$

Considérant la dynamique du robot Nao qui consiste en une suite de moteur combinez dans l'équation 5.4, la loi de contrôle de la proposition TDE basée sur SMC est donc obtenue par la résolution de l'équation suivante :

$$\tau_M(t) =$$
$$\bar{A}\big[\ddot{q}_d(t) - \beta\dot{\tilde{q}}(t) - K_1\big(\sigma(t)\big)\sigma(t) - K_2 D\big(\sigma(t)\big) sign(\sigma(t))\big] + \hat{H}(q(t), \dot{q}(t), \ddot{q}(t)) \tag{5.11}$$

Avec une garantie de convergence de la position de référence des joints si les gains K_{2i} positifs pour i = 1, … ,12 sont sélectionnés comme le montre la formule suivante :

$$K_{2i} > \frac{1}{\bar{A}_{ii}}|\Delta H_i| \tag{5.12}$$

Preuve. Pour analyser la stabilité du système en boucle fermée, la fonction suivante de Lyapunov est sélectionnée, en utilisant l'équation 5.13 nous cherchons à montrer que la

dérivée est décroissante et tant vers 0 et cella sera la preuve que notre système est stable mathématiquement.

$$V(t) = 0.5\sigma^T(t)\sigma(t) \tag{5.13}$$

En prenant la dérivée temporelle de la fonction positive ci-dessus, on obtient l'équation 5.14.

$$\begin{aligned}\dot{V}(t) &= \sigma^T(t)\dot{\sigma}(t) \\ &= \sigma^T(t)\left[\ddot{q}(t) - \ddot{q}_d(t) + \beta\dot{\tilde{q}}(t)\right] \\ &= \sigma^T(t)\left[\bar{A}^{-1}\left[\tau_M(t) - H(q(t), \dot{q}(t), q(\ddot{t}))\right] - \ddot{q}_d(t) + \beta\dot{\tilde{q}}(t)\right]\end{aligned} \tag{5.14}$$

$$\begin{aligned}\dot{V}(t) &= \sigma^T(t)\left[\bar{A}^{-1}\Delta H - K_1\big(\sigma(t)\big)\sigma(t) - K_2 D\big(\sigma(t)\big)sign(\sigma(t))\right] \\ &= \sum_{i=1}^{12}\frac{1}{\bar{A}_{ii}}\sigma_i(t)\Delta H_i - K_{1i}\frac{\sigma_0\sigma_i^2(t)}{\sigma_0+|\sigma_i(t)|} - K_{2i}\frac{|\sigma_i(t)|}{d_i+(1-d_i)exp^{-|\sigma_i|^{l_i}}} \\ &\leq \sum_{i=1}^{12}\frac{1}{\bar{A}_{ii}}|\sigma_i(t)||\Delta H_i| - K_{1i}\frac{\sigma_0\sigma_i^2(t)}{\sigma_0+|\sigma_i(t)|} - K_{2i}\frac{|\sigma_i(t)|}{d_i+(1-d_i)exp^{-|\sigma_i|^{l_i}}}\end{aligned} \tag{5.15}$$

5.3.3 Transformation du couple en position

Dans cette section, nous répondons à un critère de base pour que tout fonctionne. Même si nous calculons le torque qu'il faut pour nos moteurs, il faut néanmoins refaire la conversion en position pour arriver avec $q_{ctl}(t)$ en sortie sous la forme d'une position. C'est ce que les équations 5.16 et 5.17 représentent.

$$\tau_M(t) = K_p\big(q(t) - q_{ctl}(t)\big) + K_d\dot{q}(t) \tag{5.16}$$

$$q_{ctl}(t) = q(t) + K_p^{-1}(K_d\dot{q}(t) - \tau_M(t)) \tag{5.17}$$

5.4 L'expérimentation

L'objectif de l'expérimentation est de rendre la marche d'un robot Nao plus robuste avec l'ajout d'un module. La performance ici est évaluée par la robustesse lors d'impact en évitant les chutes et par la vitesse de marche sans faire de chute. Le module, nommé « DynamicControle », est le résultat de la nouvelle approche pour une marche plus robuste.

5.4.1 DynamicControl

Le module « DynamicControl » est le module qui permet de répondre à l'objectif de développer un algorithme plus robuste pour la marche du robot bipède Nao. Ce module permet de calculer un couple qu'il va ensuite transformer en position et envoyer la commande au moteur. En travaillant avec des angles, le « DynamicControl » calcule l'accélération avec la fonction « calcAcceleration », on calcule l'erreur de position en utilisant la fonction « calcErrorPosition » et on arrive à calculer l'erreur de vitesse avec la fonction « calcErrorVelocity ». À partir de cela, on peut utiliser une fonction qui nous permet de calculer la surface de glissement, « calcSlidingSurface ». Ces étapes permettent d'arriver à calculer le torque, « calcTorque » que sera bonifié par un calcul basé sur la méthode d'une estimation d'un délai de temps (time delay estimation, TDE), « calcTDE ». Une fois cela calculé, pour que les moteurs soient en mesure de recevoir la commande il faut le transformer en position.

Pour rendre le modèle plus doux lors de son exécution dans le robot et éviter au maximum les risques que le robot se brise pendant un test. Il a été ajouter un module qui calcule la saturation. Grâce à ce module sigmoïde, l'algorithme est moins brusque et permet de diminuer le « chattering » du robot.

Il est possible de consulter le module « DynamicControl » à l'annexe I pour comprendre davantage les fonctions présentées plus haut.

5.4.2 Phase 1 : L'intégration

Pour l'intégration dans l'architecture du projet de Naova, le module « DynamicControl » a été ajouté dans le code. Il ne remplace pas leur module « Walk2014Generator » qui est actuellement utilisé, mais il vient le compléter en lui ajoutant davantage de robustesse. Pour le moment, si on retire notre module du code, la marche devient moins performante. L'intégration est complexe, mais la figure 5.4 illustre l'interaction entre les modules déjà existants et le module « DynamicControl ».

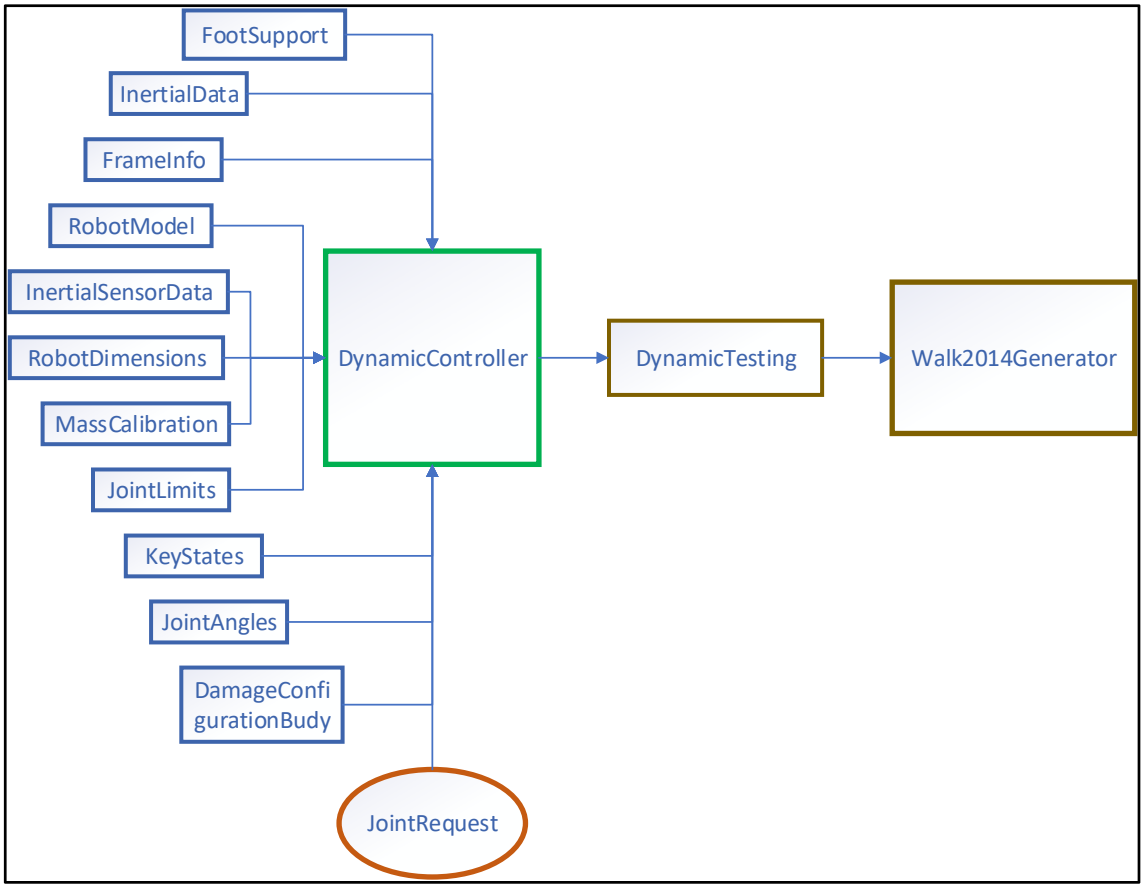

Figure 5.4 Intégration du fichier DynamicController. Rectangle bleu signifie requis, ellipse orange signifie utiliser, rectangle rouge reçoit

On retrouve en entrée dans le module « DynamicControl » les variables suivantes : Beta, il y a Kp et Kd qui forment les gains proportionnel et dérivé, le K1, K2 sont des variables, A est une matrice diagonale, theta0 et finalement *di*. Ces variables sont fournies par un fichier de configuration du même nom. Ce qui nous permet au besoin de faire des ajustements rapidement directement dans la simulation ou sur le robot en pleine action. La rapidité d'exécution nous permet de voir l'impact directement sur le robot comme le montre la figure 5.5.

5.4.3 Phase 2 : La simulation

La simulation actuelle, nommée SimRobot, nous permet de faire des parties de soccer configurées à l'avance. Cependant, comme la simulation n'offre pas tout, il a donc été important de bonifier la simulation avec des outils qui ont permis de voir l'évolution des ajouts dans l'algorithme. Cela a permis de mesurer les résultats du module, de tester différentes combinaisons de variables et d'interpréter les résultats.

Grâce à des modules de configuration, cela permet d'apporter des modifications dans la simulation, de changer des paramètres en temps réel sans avoir besoin de mettre fin à la simulation et recommencer. La figure 5.5 représente le simulateur SimRobot qui est utilisé. Il y a la vue 3D du robot, des graphiques avec l'option de les générer et dans deux consoles il est possible de faire des commandes qui affectent directement les graphiques et la simulation 3D.

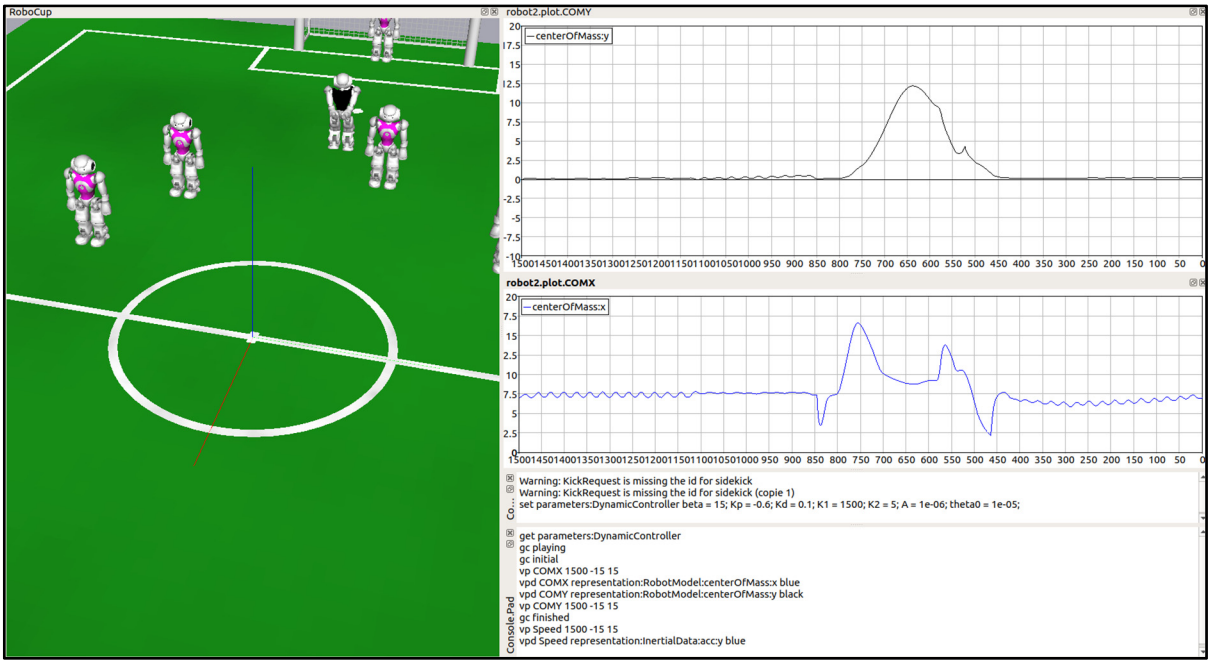

Figure 5.5 Simulation 3D avec SimRobot

La phase de simulation est importante pour déterminer ce qu'il est possible de mettre comme paramètre dans le robot, car grâce à cette phase il est possible d'éviter une grande partie des accidents sur le robot.

5.4.4 Phase3 : Test sur le robot

En utilisant l'interface de SimRobot, il est possible de se connecter directement sur un robot via un fil internet, de voir ce que le robot voit, de monitorer certaines données reçues des capteurs et d'y modifier des paramètres en temps réel, le tout de la même façon que lors de la partie en simulation. La figure 5.6 montre une connexion avec un robot et quelques options possibles pour le monitorage, dont la vision en temps réel du robot. Cette phase est présente pour fermer la boucle de test, car elle est la plus précise.

Figure 5.6 Monitorage en temps réel

Les tests pour valider le modèle et tester les performances de l'algorithme ont été faits en Australie pendant le Championnat du monde de robotique ou dans le laboratoire à l'école de technologie supérieure. Une équipe constituée de cinq robots avait dans leur code durant les matchs le module « DynamicControl ». Un total de dix parties contre des équipes de différents pays a été faites. Par observation qualitative, les robots, avec l'algorithme proposé dans ce chapitre, passaient plus de temps à chercher la balle, car lors de contacts physique seulement l'adversaire sans l'algorithme tombait. Les parties ont été effectuées contre des équipes de différents niveaux et toujours le même avantage pour l'équipe ayant le module « DynamicControl ».

Plusieurs tests ont été faits pour comprendre l'impact de chaque variable sur le système. De plus, les tests servaient à identifier quelle valeur les variables pouvaient prendre pour offrir le système le plus robuste. Le tableau 5.1 montre quelques résultats récoltés durant les tests à différents moments dans le processus. Dans chacun des tests, l'environnement était une variable contrôlée. Lors des premiers tests avec le robot, les valeurs zéro, donc initiales, empêchaient le robot de marcher, car il tremblait beaucoup trop. Ce phénomène est appelé

« chattering ». Le risque de bris était très grand. Par la suite, avec plusieurs itérations et plus les tests avançaient, plus le système devenait stable et robuste.

Tableau 5.1 Les tests les plus significatifs

Variable	**Initial**	**Test2**	**Test3**	**Test4**	**Test5**	**Test6**	**Test7**	**Test8**	**Test9**	**Test10**
A	0.001			1 10-4	1 10-4	1 10-4	1 10-5	1 10-5	1 10-5	1 10-5
Kp	100			0.1	-0.4	-0.4	5	100	100	100
Kd	5			0.5	0.1	0.1	100	5	5	5
K1	2	10	0.001	1	400	2500	5	2	10	20
K2	500	700	2500	3500	2500	5	2	2	2	2
Beta	1		2	100	10	10	150	10	10	10
Theta0	1 10-6				1 10-5		1 10-5	1 10-5	1 10-5	1 10-5
di	1	1	1	1	1	1	100	0.5	0.1	0.01

Les 10 tests présentés au tableau 5.1 sont un échantillon de l'ensemble des tests effectués. Ils ont été néanmoins les plus pertinents pour identifier l'impact de chaque variable et de voir les effets positifs sur le système. Grâce à ces tests, il a été déterminer que le code était bon mais qu'il comportait des lacunes. Les tests sept à dix ont subi une amélioration dans le code. L'ajout de la variable *di*. À la suite de plusieurs séries de tests, il a été conclu que le test dix offrait une robustesse satisfaisante, car elle permettait de répondre à tous les objectifs de cette recherche. Cette portion des tests a été faite par observation directe avec le comportement du robot, tant en simulation que physiquement. Une partie des tests a été faite par essaie et erreur le temps que les graphiques soient récupérés pour faire des analyses plus précises. Cependant, la couverture de la surface a été couverte, car des bornes de référence ont permis de simplifier les tests.

5.5 Les résultats

Entre deux robots ayant deux différents algorithmes de marche, le premier avec le module populaire et le deuxième avec le nouveau « DynamicControl » qui a été présenté, à vitesse égale soit 310 mm/s selon le standard, il ne semble pas y avoir de grande différence à l'œil humain. Cependant, le second robot avec le module « DynamicControl » a l'avantage, car l'adversaire ne pouvait le faire tomber lors d'impact l'un contre l'autre. La vitesse moyenne des robots durant la compétition est de 310 mm/s, car c'est ce que propose et suggère l'algorithme populaire utilisé par les autres équipes pour garder une stabilité raisonnable. Maintenant, dû au module « DynamicControl » la vitesse de base passe de 310 mm/s à 350 mm/s et la vitesse maximum est de 400 mm/s. Ce changement donne l'impression que le robot court sur le terrain comparativement aux robots de l'autre équipe. De plus, il est important de noter que le plafond à 400 mm/s a été établi en compétition, car le robot v5, lorsqu'il marche plus vite que 400 mm/s, il fait fondre des pièces dans les articulations. Le nouveau Nao v6 est plus résistant et donc il sera pertinent de voir la vitesse maximum possible d'atteindre.

Le premier volet à analyser est en simulation, pour cela, nous regarderons des graphiques avec une vitesse de 310 mm/s pour une marche en ligne droite. Ensuite en gardant le même tapis, le même éclairage, la même balle et sans changer aucun paramètre sauf la vitesse de la marche qui sera de 400 mm/sec, nous reproduirons la marche pour analyser les mêmes graphiques et voir si le robot est en mesure de marcher et de rester stable. Chacune des articulations des jambes sera analysée pour voir leur évolution. Chaque graphique va être un comparatif entre la valeur désirée et la valeur qui est obtenue sur le moteur de l'articulation.

5.5.1 Les résultats en simulation

Les résultats en simulation révèlent une stabilité du système. Le tableau 5.2 possède deux sections, en rouge, il montre la valeur de l'erreur en degré et en bleu, c'est l'erreur en radian pour chacun des joints. Le tableau 5.2 est la comparaison entre la valeur du joint désiré et la

valeur réelle obtenue sur le moteur de l'articulation. La figure 5.7 identifie la valeur désirée par $q_{ctl}(t)$ et la valeur réelle par $q(t)$. Ces résultats proviennent des moteurs dans la simulation.

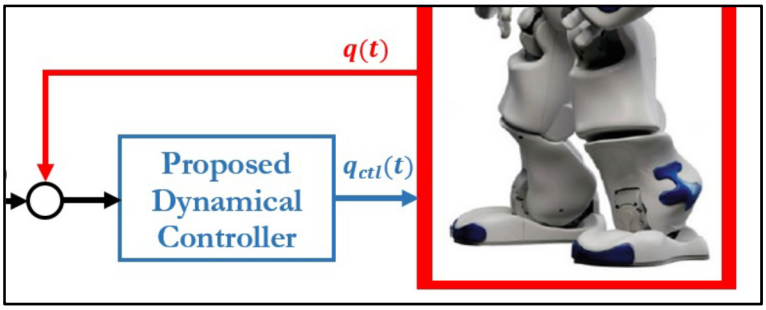

Figure 5.7 Lien entre la valeur désirée et la valeur réelle

Tableau 5.2 Comparatif en simulation de l'erreur de l'angle des moteurs avec 2 vitesses différentes

	Vitesse de 310 mm/sec	**Vitesse de 400 mm/sec**	**Écart entre les vitesses**	**Vitesse de 310 mm/sec**	**Vitesse de 400 mm/sec**	**Écart entre les vitesses**
Joint	Erreur degré			Erreur Radian		
AnklePitch	0.050	0.019	0.030	0.000876	0.0003434	0.0005326
AnkleRow	0.005	0.009	0.003	0.0000979	0.0001584	0.0000605
HipPitch	10.914	9.494	1.420	0.1904935	0.1657018	0.0247917
HipRow	0.000	0.064	0.063	0.0000096	0.0011211	0.0011115
HipYawPitch	0.046	0.448	0.401	0.0008169	0.0078269	0.00701
KneePitch	0.019	0.034	0.014	0.0003435	0.0006013	0.0002578

Pour les deux rectangles, rouge et bleu, la première colonne est pour l'erreur sur la vitesse à 310 mm/s entre la valeur désirée et réelle. La seconde colonne est pour une vitesse de 400 mm/s et la troisième colonne est pour voir l'écart entre les deux vitesses. Comme la différence est petite, cela montre que l'algorithme permet d'augmenter la vitesse du robot et de garder une stabilité équivalente à une vitesse plus basse.

Le tableau 5.2 représente les erreurs entre la valeur désirée et la valeur réelle sur le robot. Il est conclu que l'erreur est pour tous les angles très petits. Cependant, l'erreur du « HipPitch » est nettement supérieure aux autres. La raison est que cet angle a pour point de référence le robot au repos comme le montre la figure 5.8.

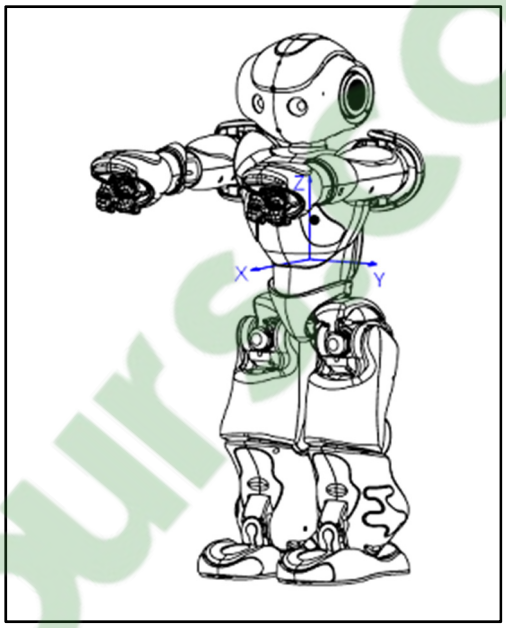

Figure 5.8 Système de coordonnée
et point initial
Tirée de Aldebaran (2013)

La réalité est que pour une meilleure stabilité ce moteur est forcé à avoir un angle qui est entre 10 et 15 degrés ce qui explique cette variation. Pour chaque donnée récoltée, une analyse statistique est faite afin de permettre de produire des graphiques plus faciles à interpréter. La procédure est la même pour chaque tableau, une moyenne de l'erreur est calculée entre la valeur de l'angle désiré $q_{ctl}(t)$ pour l'articulation et la valeur réelle $q(t)$ que le robot utilise à la suite de notre module « DynamicControl ». Il y a un total de 1500 « steps » dans les graphiques. Les « steps » représentent des « frames » et par seconde il y a 100 « steps ». Cela implique donc que les graphiques représentent 15 secondes de marche en ligne droite pour le Nao et le tout sans interférence. En résumé, l'axe x représente le temps et

l'axe y représente la variation en radian. Tous les graphiques se lisent de gauche à droite. La figure 5.9 est le seul graphique pour la simulation. Il sert d'exemple pour nous permettre de passer à la prochaine étape si les résultats sont convenables.

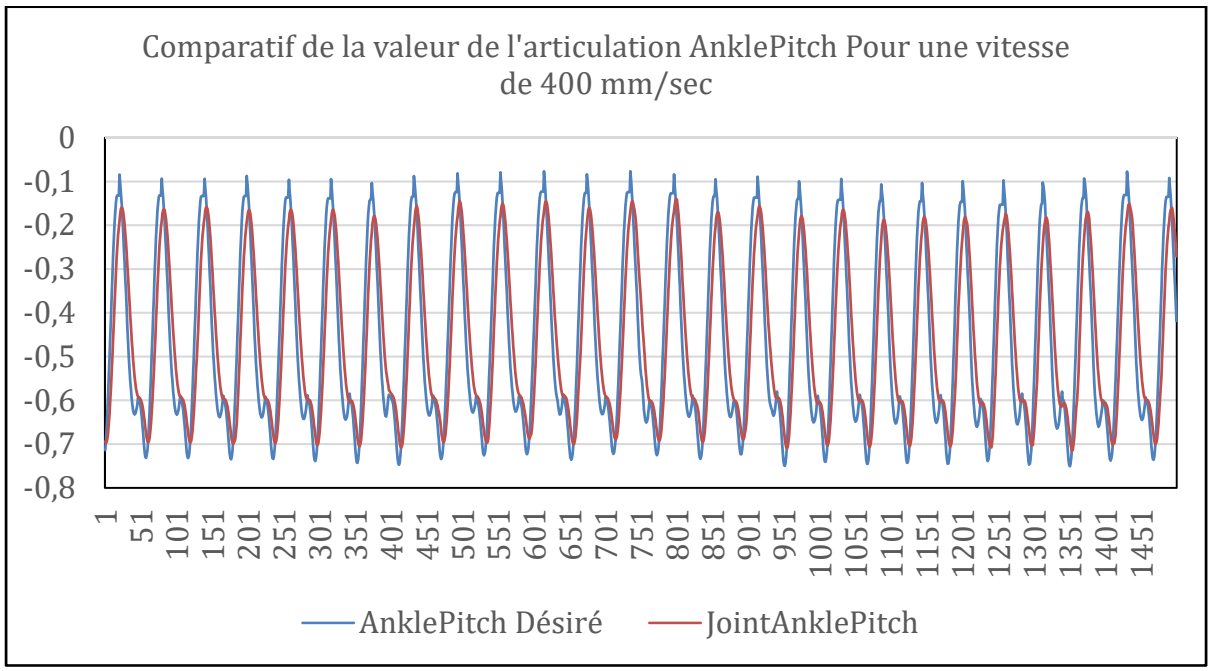

Figure 5.9 Résultats pour l'articulateur AnklePitch (radian)

Les résultats lors de la simulation pour le joint « AnklePitch » montrent une erreur de 0.0003434 radian. Ce moteur possède une erreur très petite. Grâce à cela, Il en est donc convenu qu'il est possible de passer à l'étape sur le vrai robot.

5.5.2 Les résultats avec le robot

Dans cette section, les résultats présentés proviennent directement de la réaction du robot dans son environnement. Cette étape est possible puisque les résultats en simulation, plus haut, sont satisfaisants et que les conditions de sécurité pour le robot sont respectées.

Le tableau 5.3 représente la différence entre la valeur désirée et la vrai valeur prise sur le moteur à une vitesse de 400 mm/s.

Tableau 5.3 Tableaux comparatifs des différentes articulations

Vitesse de 400 mm/s		
Joint	**L'erreur en radian**	**L'erreur en degré**
AnklePitch	0.0270010	1.547
AnkleRow	0.0050970	0.292
HipPitch	0.2368673	13.571
HipRow	0.0135150	0.774
HipYawPitch	0.0194682	1.115
KneePitch	0.0120426	0.689

La figure 5.10, qui suit, représente le parcours du COM sur l'axe des X pour le robot marchant à une vitesse de 400 mm/s. Chacune des valeurs dans le tableau 5.3 est le résultat des analyses de la figure 5.11 à la figure 5.15. Ils représentent chacune des articulations des jambes du robot pendant son action de marcher. Rapidement, il est possible dire que les résultats sont mathématiquement valables et donc offrent un très bon potentiel comme solution à la problématique initial. Ces résultats sont positifs et prometteurs.

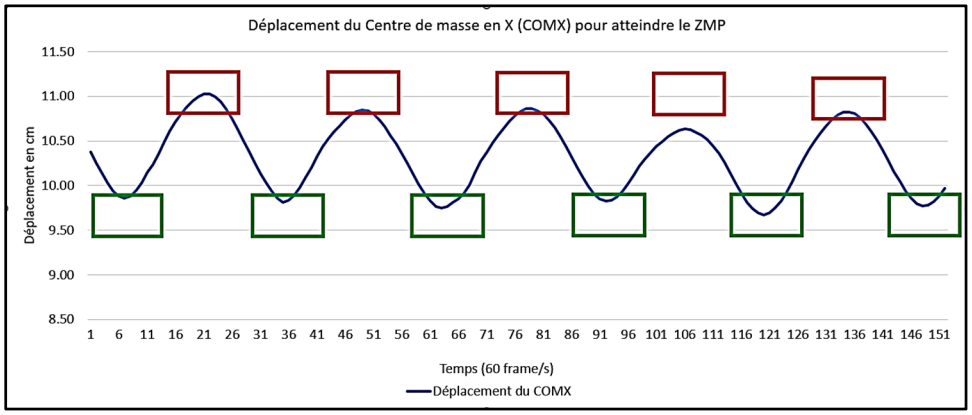

Figure 5.10 Centre de masse du robot Nao avec une vitesse de 400 mm/s

Même en augmentant la vitesse sur le Nao, il est important de noté que le déplacement est encore acceptable. Il n'atteint pas exactement le centre des carrés (représentant le pied du robot), mais le COM arrive assez prêt pour garder un équilibre. Pour respecter les objectifs, il faut comparer l'erreur de la méthode des autres équipes utilisée actuellement avec une vitesse de 310 mm/s avec la méthode proposée, mais à une vitesse de 400 mm/s. On remarque dans le tableau 5.4 que même à plus grande vitesse, soit 400 mm/s au lieu de 310 mm/s, la méthode proposée dans ce mémoire permet de garder une erreur majoritairement plus faible ou similaire. À vitesse égale, il y a un gain sur l'ensemble des articulations.

Tableau 5.4 Comparatifs entre la méthode populaire et la méthode proposée

	Méthode populaire, Vitesse de 310 mm/s.	**Méthode proposée, Vitesse de 310 mm/s.**	**Différence entre les 2 méthodes 310 et 310**	**Méthode proposée, Vitesse de 400 mm/s**	**Différence entre les 2 méthodes 310 et 400**
Articulations	**Variation degré**				
AnklePitch	4.491	0.050	4.440	1.547	2.944
AnkleRoll	0.620	0.005	0.614	0.292	0.328
HipPitch	16.698	10.914	5.783	13.571	3.126
HipRoll	0.602	0.001	0.602	0.774	-0.171
HipYawPitch	0.685	0.046	0.638	1.115	-0.429
KneePitch	5.058	0.019	5.039	0.689	4.368

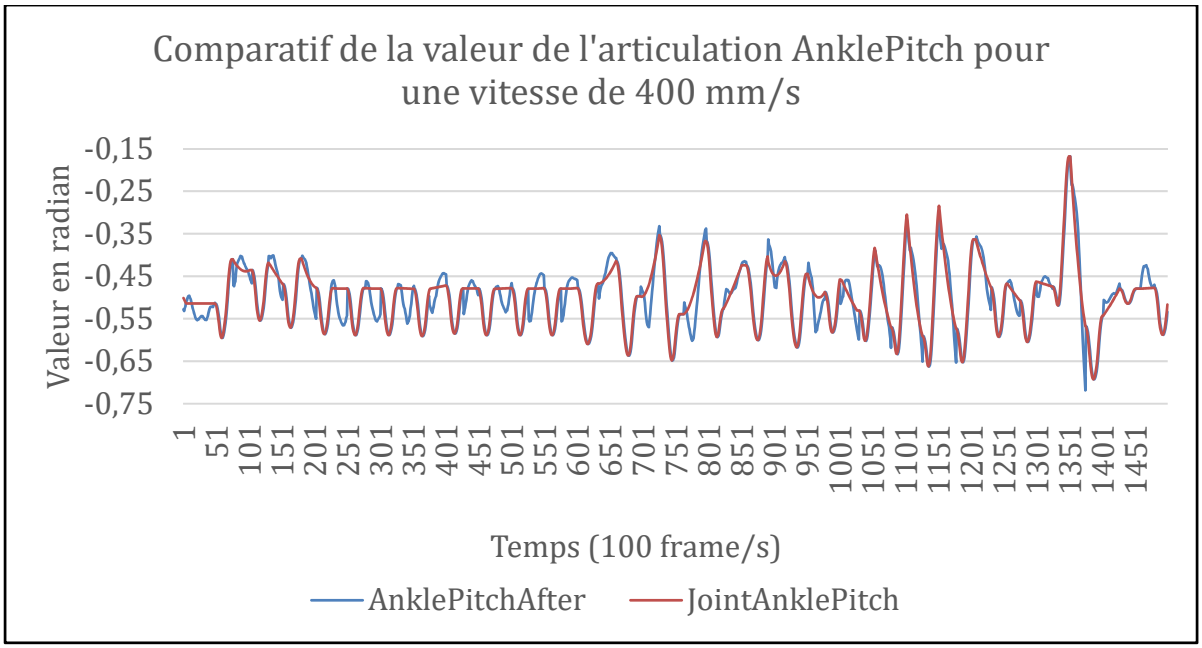

Figure 5.11 Comparatif de la valeur de l'articulation AnklePitch pour une vitesse de 400 mm/s

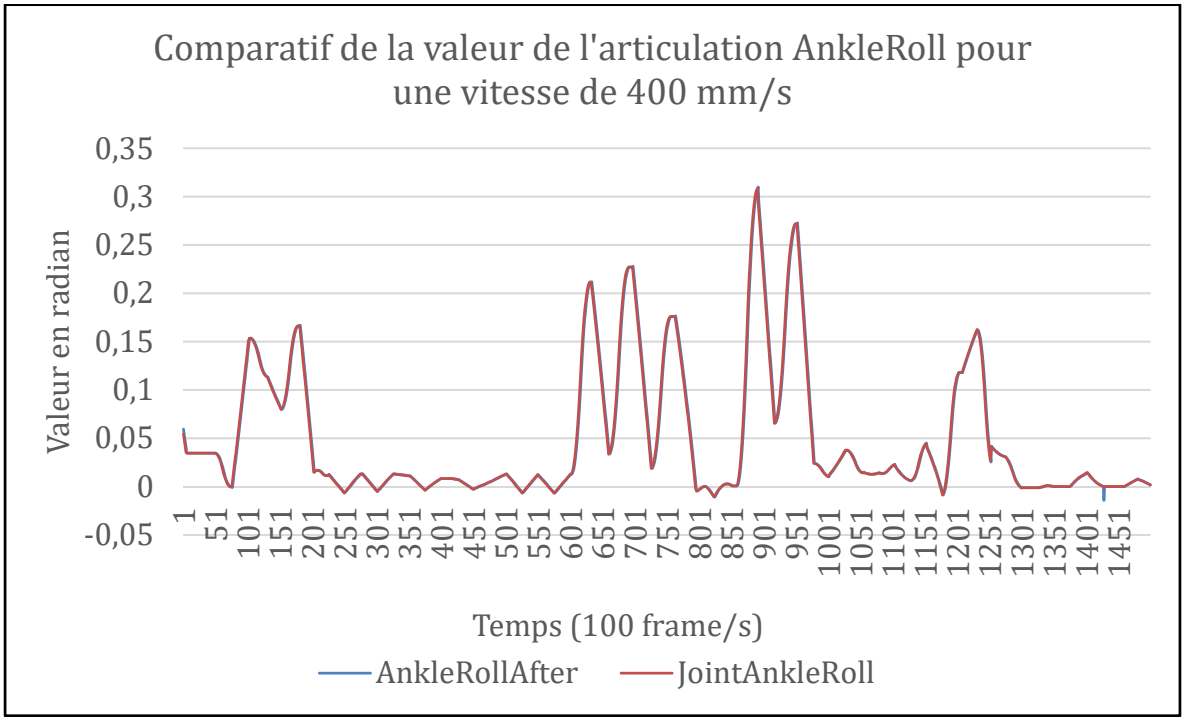

Figure 5.12 Comparatif de la valeur de l'articulation AnkleRoll pour une vitesse de 400 mm/s

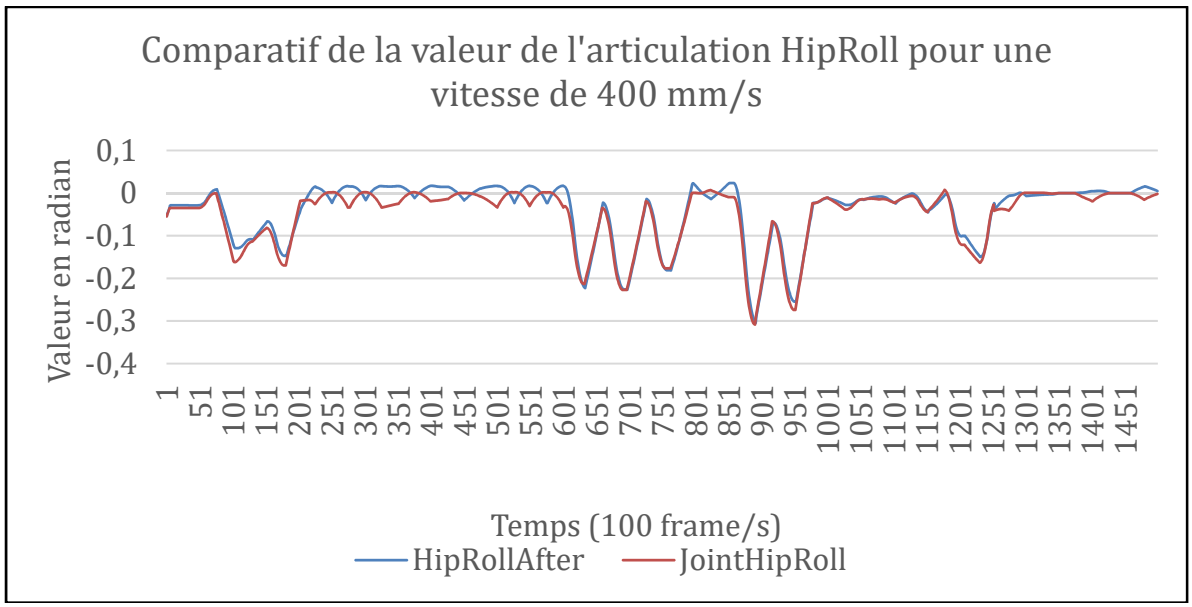

Figure 5.13 Comparatif de la valeur de l'articulation HipRoll pour une vitesse de 400 mm/s

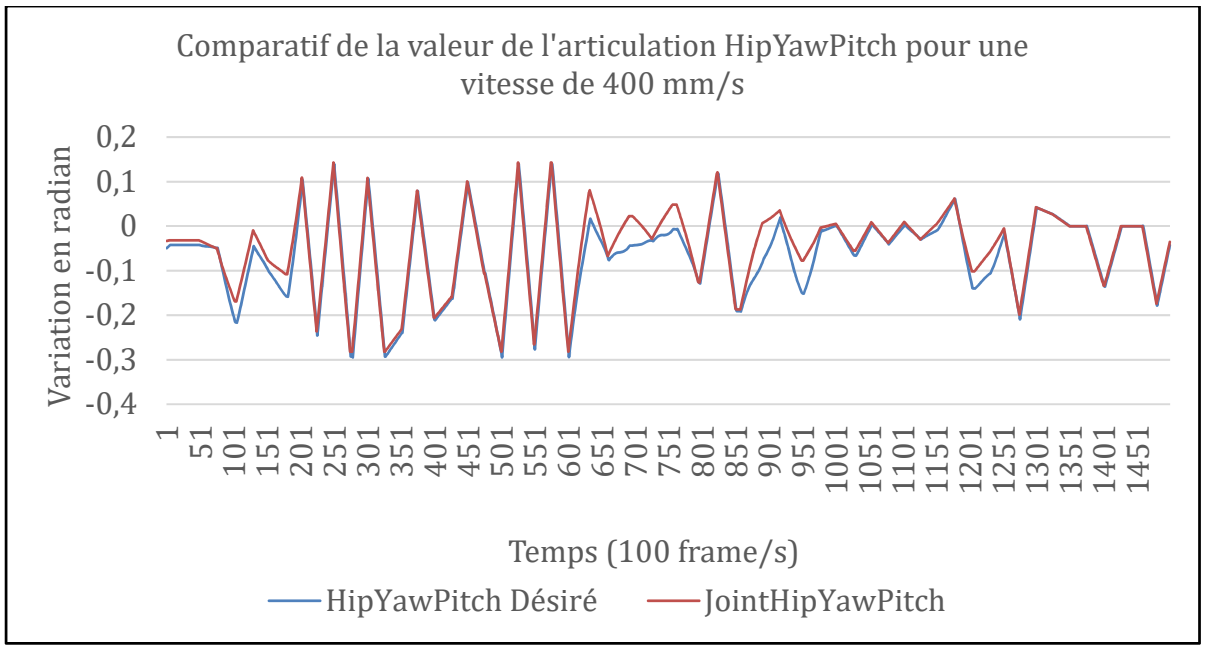

Figure 5.14 Comparatif de la valeur de l'articulation HipYawPitch pour une vitesse de 400 mm/s

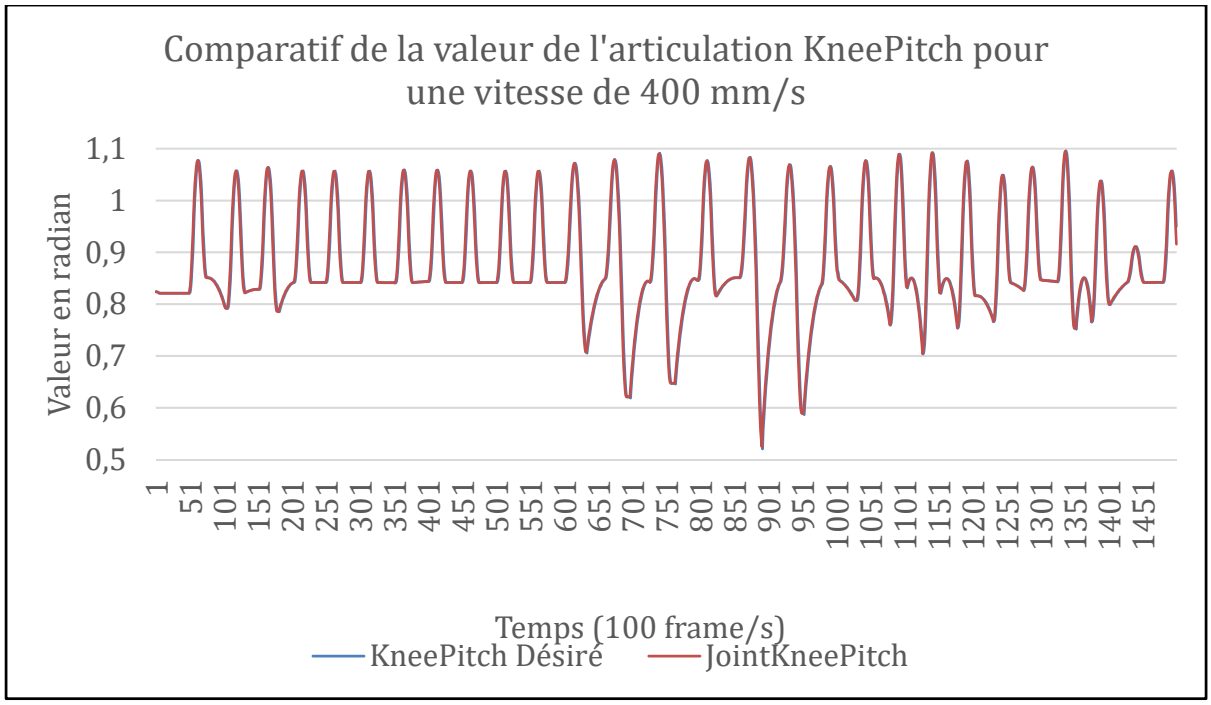

Figure 5.15 Comparatif de la valeur de l'articulation KneePitch pour une vitesse de 400 mm/s

5.6 Conclusion

La méthode utilisée est une méthode qui est très prometteuse pour l'avenir des robots bipèdes avec une petite capacité de calcul. Elle est robuste et permet de travailler à partir de maintenant avec des robots plus rapides et des résultats meilleures que la loi de commande initiale. Actuellement, l'erreur est majoritairement inférieure à un degré. Le robot est capable de rester debout même si le tapis n'est pas 100% plat, et cela avec une plus grande vitesse.

Cette nouvelle approche est donc une confirmation de l'atteinte des objectifs pour cette recherche. Pour plus de validation, il est possible de voir les figures de l'annexe II. Ces figures sont les résultats des tests avec le code de l'équipe BHuman. De plus, l'annexe III présente des tableaux avec des comparatifs en pourcentage sur les résultats obtenus lors de la comparaison entre la méthode de marche populaire et la méthode de marche proposée.

CONCLUSION

En conclusion, il est très complexe de travailler avec des robots humanoïdes qui jouent au soccer. Grâce au travail collaboratif des équipes de la RoboCup depuis plusieurs années il est possible de faire des progrès très rapidement. Il est bénéfique d'avoir une plateforme de départ avec des outils adaptés pour le succès du projet comme une simulation fonctionnel avec le modèle du robot et une plateforme pour contrôler le robot. Cela permet d'avoir des objectifs ambitieux comme il a été présenté pour cette recherche.

Certes nous bénéficions d'une plateforme de départ, cela représente le travail des autres laboratoires de recherche. La première partie de la problématique était qu'il est difficile de développé sur un projet inconnu. Par conséquent, le premier objectif était de valider le modèle mathématique du robot. Pour cela, dans le chapitre 3, il est présenté que le modèle mathématique est valide pour le Nao V5 qui est utilisé pour ce projet. Le robot a été analysé pièce par pièce afin d'en comprendre chaque articulation. Recalculer et comparer chaque détail de chaque chaine du robot a permis de comprendre le modèle. Par la suite, les modules dans le code sont devenus beaucoup plus simples à comprendre et cela facilitait leur utilisation dans nos développements.

La partie suivante de la problématique était la stabilité du robot lors de la marche qui pouvait être améliorée et la vitesse lente que nous voulions augmenter. Pour cela l'algorithme du ZMP a été exploré dans le chapitre 4. Cet algorithme a été un bon ajout, car cela a fait augmenter la vitesse de la marche et améliore la stabilité, mais cela n'a pas eu l'effet permettant de valider les objectifs. En conclusion, l'ajout du ZMP fonctionne sur le Nao, mais seul, cela ne permettait pas d'atteindre nos objectifs ambitieux même si une amélioration était notée. Grâce à cet ajout, une vitesse de 310 mm/s pouvait être atteinte, mais en simulation seulement. De plus, le ZMP étant très populaire, la majorité des équipes l'avait implémenté aussi. Il fallait donc explorer une autre méthode pour complémenter celui-ci.

Le chapitre 5 présente cette solution innovatrice. Cette commande est basée sur un algorithme de contrôle par mode de glissement combiné à une méthode avec une estimation du délai de temps. Cette approche fonctionne sur le CPU du Nao V5 et a permis d'atteindre une vitesse impressionnante de 400 mm/s tout en gardant une stabilité similaire à l'approche des autres équipes qui vont à une vitesse de 310 mm/s. Le tableau 5.4 montre les principaux résultats qui confirment la robustesse de cette approche.

En terminant, bien que les résultats ne soient pas tous égaux ou inférieurs aux valeurs du tableau initial, le tableau 0.1, cet ajout permet d'avoir une approche différente et donne des options pour encore rendre le robot encore plus robuste et développer davantage pour les prochaines recherches.

Il y a énormément de paramètres à prendre en considération et malgré le succès des modifications apportées au projet Naova et que les résultats sont très prometteurs pour le futur, il n'en demeure pas moins que la faiblesse de notre algorithme est que comparativement à l'ancienne méthode pour générer la marche le modèle prenait en considération la tête et donc les caméras. Cela permettait à la tête de ne pas trembler lors d'accélérations ou décélération rapide donc de ne pas perde la balle de vue. Le point faible du module « DynamicControl » est qu'effectivement la marche trouble la vision et fait perdre la balle au robot. En contrepartie, les robots restent beaucoup plus rapides et stables sur le terrain que les autres robots.

TRAVAUX FUTURS

Malgré des résultats très satisfaisants, il reste encore beaucoup à accomplir. Déjà il serait intéressant de voir les résultats avec une correction par intégration en remplacement de la correction par saturation. Cela rendrait théoriquement notre modèle proposé encore plus robuste. Cependant, pour cette recherche le modèle du robot utilisé est le V5, donc très limité en puissance de calcul, c'est ce qui a justifié le choix de la correction qui est moins exigeante pour le processeur. Maintenant que nous avons accès au v6, cette option est envisageable. De plus, actuellement, seulement les jambes sont prises en considération dans le modèle. Cette restriction du modèle a fait ressortir d'autres problématiques, comme le tremblement des caméras sur la tête. Cette problématique engendre une perte de la balle plus fréquente ainsi que la difficulté de l'identifier sur une longue distance. Il faudrait donc considérer dans le prochain modèle cette problématique. Autre que la marche en ligne droite, durant une partie de soccer, il y a des chutes, des accrochages et il faut frapper la balle. Il faudrait donc également inclure un « kick » afin de rendre la balle plus loin, mais être sûr que le robot ne tombe pas après chaque « kick ».

Maintenant que l'intégration du module est faite et que nous avons des résultats prometteurs pour la marche du robot Nao, il va falloir regarder l'option de changer l'estimation de la dynamique et de prendre la vraie dynamique du robot. Cela augmenterait la performance et diminuerait potentiellement l'erreur restante. De plus, avec le nouveau modèle du robot Nao sortie en 2019, il est maintenant possible d'envisager de faire davantage de calcul avec son nouveau CPU. Il va être intéressant de constater l'impact sur la performance.

ANNEXE I

PROGRAMME C++ DU MODULE DYNAMICCONTROL

```
Angle DynamicController::calcAcceleration(Angle dQ0, Angle dQ1, float
delta){
    return (dQ0-dQ1)/delta;
}

Angle DynamicController::calcErrorPosition(Angle Q, Angle Qd){
    return Q - Qd;
}

Angle DynamicController::calcErrorVelocity(Angle dQ, Angle dQd){
    return dQ - dQd;
}

Angle DynamicController::calcSlidingSurface(Angle dQWithError, Angle
QWithError){
    return dQWithError + beta * QWithError;

}


Angle DynamicController::sign(Angle sigma) {
    if(sigma > 0) {
        return 1;
    } else if (sigma < 0) {
        return -1;
    }
    return 0;
}

Angle DynamicController::saturation(Angle sigma){
    float phi = 0.8f;
    float phi1 = 1/ phi;

    if(abs(sigma) > phi1){
        return sign(sigma);
    }
    return sigma*phi1;
}


Angle DynamicController::calcTorque(Angle ddQd, Angle dQWithError, Angle
QWithError, Angle sigma){
    return A * (ddQd- beta* dQWithError- K1*(dQWithError + beta *
QWithError) - K2* calcD(dQWithError, QWithError)*saturation(sigma));
}
```

```
void DynamicController::calcTDE(Angle& torque, Angle torqueT1, Angle
ddQ1){
    Angle torque1 = torqueT1 - A * ddQ1; // torque1 = torque(t-1) -A *
ddQ(t-1)
    torque = torque + torque1;
}

Angle DynamicController::calcTorqueToPosition(Angle Q, Angle dQ, Angle
dQd, Angle torque){
    return Q + Kp * (Kd*(dQ-dQd) - torque);
}

Angle DynamicController::calcAngle(Vector3a desiredAngle0, Vector3a
desiredAngle1, Vector2a angle0, Vector2a angle1,  Vector2a angle2) {
    Angle Qd = desiredAngle0[0];
    Angle dQd = desiredAngle0[1];
    Angle Q = angle0[0];
    Angle dQ = angle0[1];
    Angle ddQ = calcAcceleration(dQ,angle1[1], delta);
    Angle ddQ1 = calcAcceleration(angle1[1],angle2[1], delta);

    Angle QWithError = calcErrorPosition(Q, Qd);
    Angle dQWithError = calcErrorVelocity(dQ, dQd);

    Angle sigma = calcSlidingSurface(dQWithError, QWithError);

    Angle ddQd = calcAcceleration(dQd,desiredAngle1[1], delta); // dQd(t)
- dQd(t-1) / delta

    desiredAngle0[2] = calcTorque(ddQd,dQWithError, QWithError, sigma);
    calcTDE(desiredAngle0[2],desiredAngle1[2], ddQ1); // torque(t) =
desiredAngle0[2] , torque(t-1) = desiredAngle1[2]
    Angle test = Angle(calcTorqueToPosition(Q, dQ,dQd, desiredAngle0[2]));
    // OUTPUT_TEXT("Angle desiré avant : " << desiredAngle0[0] << " Angle
desiré apres : " << test);

    return test;
}

Angle DynamicController::calcD(Angle dQWithError, Angle QWithError) {

        return 1 / (di+(1-di)*(exp(-pow(abs(dQWithError + beta *
QWithError),li))));
}
```

ANNEXE II

FIGURES DES VALEURS DES ARTICULATIONS DE LA MÉTHODE POPULAIRE

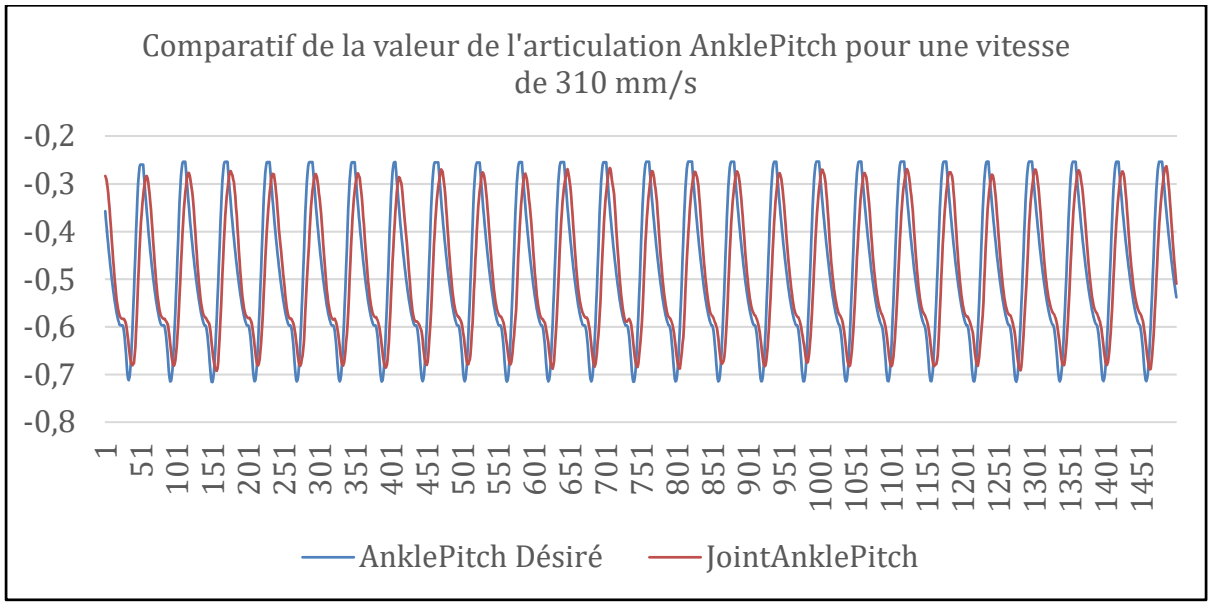

Figure-A II-1 Représentation des valeurs de l'articulation AnklePitch

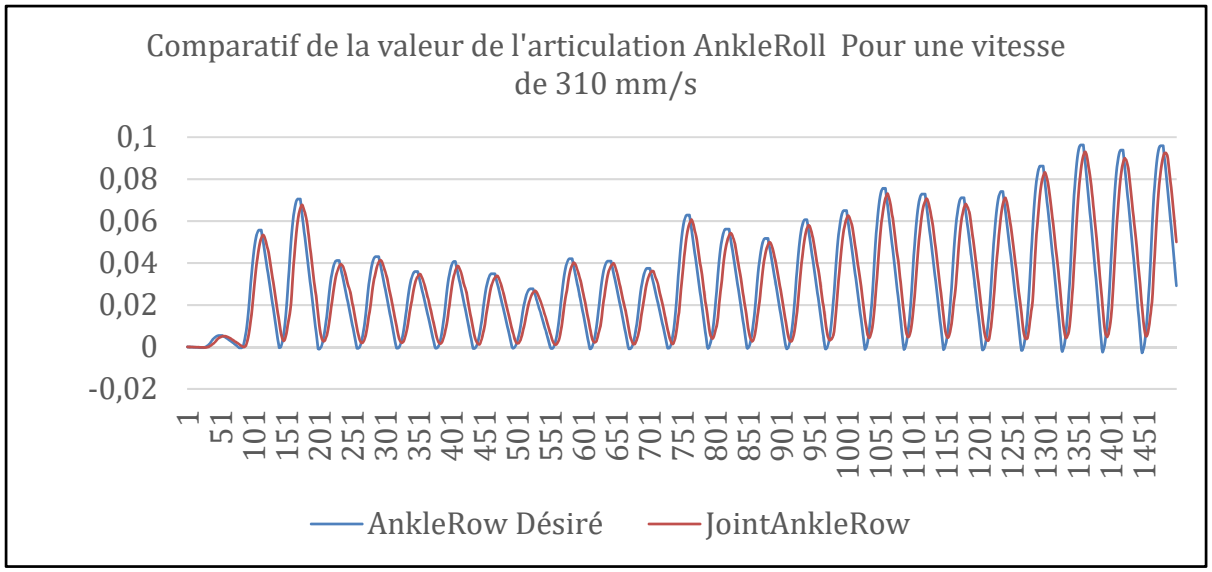

Figure-A II-2 Représentation des valeurs de l'articulation AnkleRoll

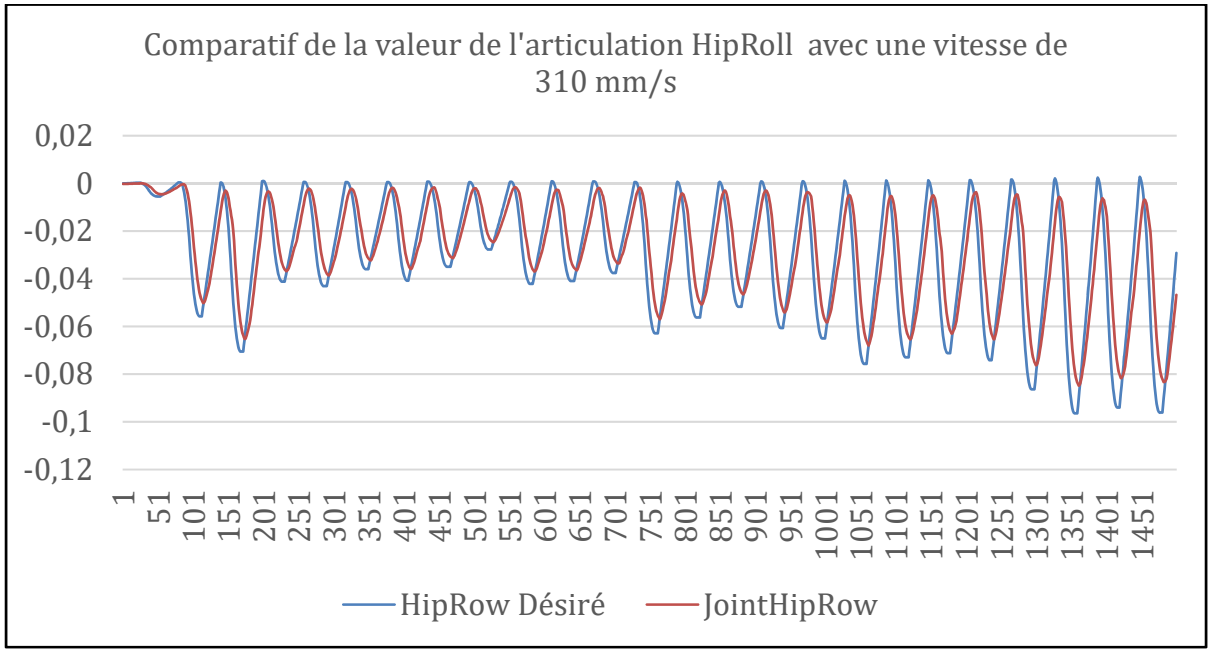

Figure-A II-3 Représentation des valeurs de l'articulation AnkleRoll

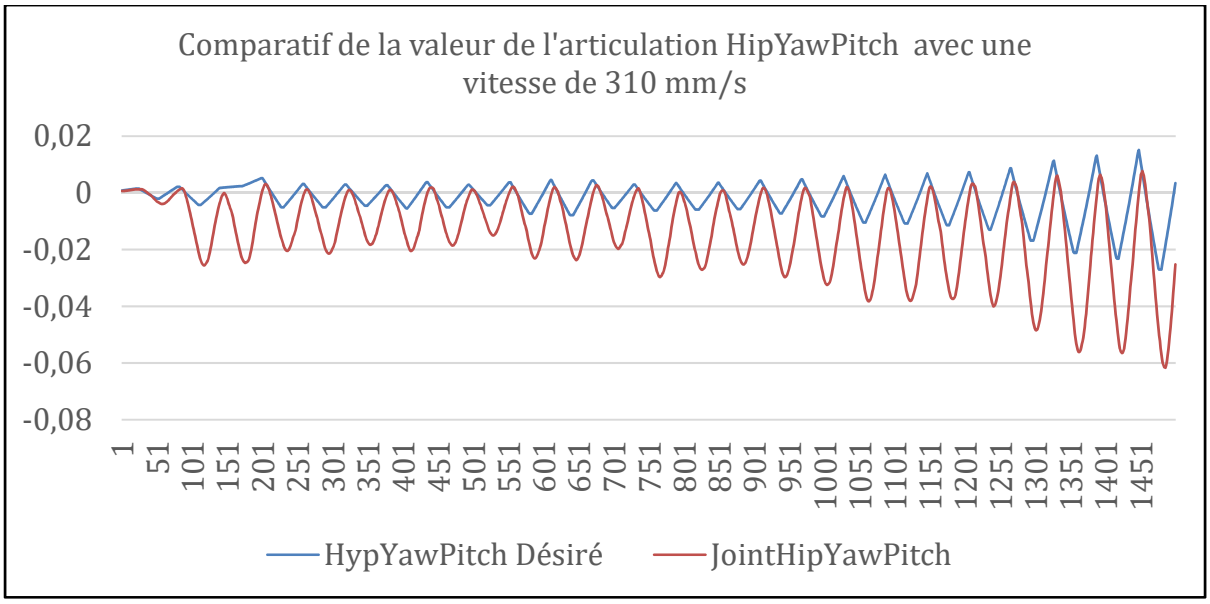

Figure-A II-4 Représentation des valeurs de l'articulation HipYawPitch

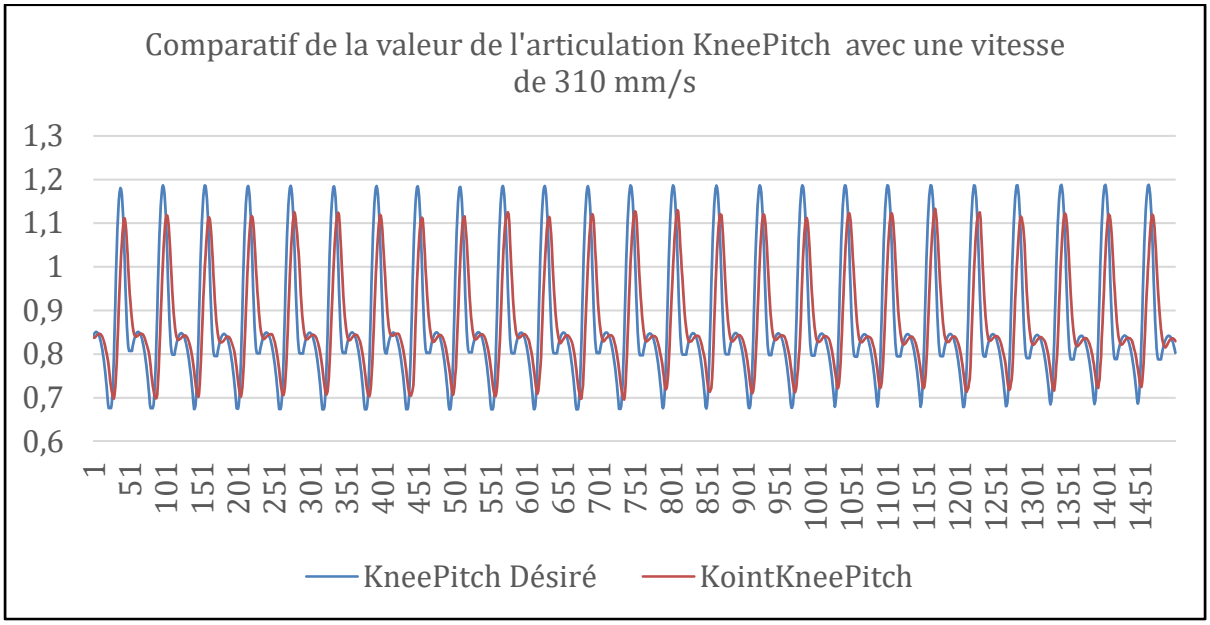

Figure-A II-5 Représentation des valeurs de l'articulation KneePitch

ANNEXE III

TABLEAU COMPARATIF ENTRE LES MÉTHODES PRÉSENTÉES

Les pourcentages dans ce tableau sont un indicateur de l'amélioration du respect de la valeur de l'angle désiré versus la valeur réel de l'angle du joint sur le robot réel. Une valeur positive signifie que l'erreur est plus petite et une valeur négative est que l'erreur est plus grande.

Tableau-A III-1 Comparatifs entre la méthode populaire et la méthode proposée

	Méthode populaire, Vitesse de 310 mm/s.	**Méthode proposée, Vitesse de 310 mm/s.**	**Amélioration en % de la méthode proposée à 300 mm/s**	**Méthode proposée, Vitesse de 400 mm/s**	**Amélioration en % de la méthode proposée à 400 min/s**
Articulations	**Variation degré**				
AnklePitch	4.491	0.05	99%	1.547	66%
AnkleRoll	0.62	0.005	99%	0.292	53%
HipPitch	16.698	10.914	35%	13.571	19%
HipRoll	0.602	0.001	100%	0.774	-29%
HipYawPitch	0.685	0.046	93%	1.115	-63%
KneePitch	5.058	0.019	100%	0.689	86%

LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES

[1] RoboCup Federation, "RoboCup Federation official website," s.d. [Online]. Available: http://www.robocup.org/.

[2] R. Feldhaus Adams, "Backflipping Robot Is A Giant Leap For Robot Kind," *NPR.org*, 17-Nov-2017. [Online]. Available: https://www.npr.org/sections/thetwo-way/2017/11/17/564850067/back-flipping-robot-a-giant-leap-for-robot-kind.

[3] Honda Co.Inc, "Asimo technical information." Sep-2007.

[4] D. E. Knuth, *The art of computer programming*, 3rd ed. Reading, Mass: Addison-Wesley, 1997.

[5] J.-J. E. Slotine and W. Li, *Applied nonlinear control*. Englewood Cliffs, N.J: Prentice Hall, 1991.

[6] J.-L. Lin, K.-S. Hwang, W.-C. Jiang, and Y.-J. Chen, "Gait Balance and Acceleration of a Biped Robot Based on Q-Learning," *IEEE Access*, vol. 4, pp. 2439–2449, 2016.

[7] Honda, "ASIMO Innovations by Honda," 2015. [Online]. Available: https://asimo.honda.com/gallery/.

[8] American Honda Motor, "Walking Technology." 2007.

[9] Boston Dynamics, "Atlas | Boston Dynamics," 2018. [Online]. Available: https://www.bostondynamics.com/atlas.

[10] M. de Waard, M. Inja, and A. Visser, "Analysis of flat terrain for the atlas robot," 2013, pp. 1–6.

[11] J. M. Hsu and S. C. Peters, "Extending Open Dynamics Engine for the DARPA Virtual Robotics Challenge," in *Simulation, Modeling, and Programming for Autonomous Robots*, vol. 8810, D. Brugali, J. F. Broenink, T. Kroeger, and B. A. MacDonald, Eds. Cham: Springer International Publishing, 2014, pp. 37–48.

[12] R. GELIN, "ROMEO Humanoid for Action and Communication," p. 19, 2012.

[13] Aldebaran, "Romeo Documentation," 2013. [Online]. Available: http://doc.aldebaran.com/2-1/home_romeo.html. [Accessed: 24-Jul-2019].

[14] UBTECH Robotics, “Walker – UBTECH Robotics,” 2012. [Online]. Available: https://ubtrobot.com/pages/walker/.

[15] E. Ackerman, “UBTECH Shows Off Massive Upgrades to Walker Humanoid Robot,” *IEEE Spectrum: Technology, Engineering, and Science News*, 08-Jan-2019. [Online].Available:https://spectrum.ieee.org/automaton/robotics/humanoids/ubtech-upgrades-walker-humanoid-robot.

[16] jdō-seigyō-gakkai iKeisoku and IEEE Control Systems Society, Eds., *2011 proceedings of SICE annual conference (SICE 2011)*. Piscataway, NJ: IEEE, 2011.

[17] IEEE Robots, “Your Guide to the World of Robotics,” 2010. [Online]. Available: https://robots.ieee.org/robots/darwin/.

[18] D. Kim, S. Jorgensen, J. Lee, J. Ahn, J. Luo, and L. Sentis, “Dynamic Locomotion For Passive-Ankle Biped Robots And Humanoids Using Whole-Body Locomotion Control,” p. 19, Jan. 2019.

[19] IEEE Robots, “Your Guide to the World of Robotics,” 2018. [Online]. Available: https://robots.ieee.org/robots/mercury/.

[20] N. A. Radford *et al.*, “Valkyrie: NASA’s First Bipedal Humanoid Robot: Learned Stochastic Mobility Prediction for Planning,” *J. Field Robot.*, vol. 32, no. 3, pp. 397–419, May 2015.

[21] M. Young, “Meet Valkyrie, NASA’s Space Robot,” 17-May-2017. [Online]. Available: https://www.skyandtelescope.com/astronomy-news/meet-valkyrie-nasa-space-robot/.

[22] T. Karsenti, J. Bugmann, and E. Frenette, “Un robot humanoïde pour aider les élèves ayant un trouble du spectre de l’autisme,” *Vivre Prim.*, p. P. 34-37, E 2017.

[23] “Setting NAO’s WiFi connection — NAO Software 1.14.5 documentation,” 16-Jul-2018. [Online]. Available: http://doc.aldebaran.com/1-14/nao/nao-connecting.html. [Accessed: 16-Jul-2018].

[24] J. Fortin and thierry Pouplier, “Modelisation du Noa, Projet du cours Modeling and Control.” Nov-2017.

[25] Aldebaran, “Nao Joints documentations,” 2013. [Online]. Available: http://doc.aldebaran.com/2-1/family/nao_h25/joints_h25.html.

[26] Aldebaran, “Nao Links documentations,” 2013. [Online]. Available: http://doc.aldebaran.com/2-1/family/nao_h25/links_h25.html.

[27] Aldebaran, “Nao Masses documentations,” 2013. [Online]. Available: http://doc.aldebaran.com/2-1/family/nao_h25/masses_h25.html.

[28] Nikolaos Kofinas, “Forward and Inverse Kinematics for the NAO Humanoid Robot,” Thesis, Technical University of Crete, Greece, 2012.

[29] M. Vukobratović and J. Stepanenko, “On the stability of anthropomorphic systems,” *Math. Biosci.*, vol. 15, no. 1–2, pp. 1–37, Oct. 1972.

[30] M. H. P. Dekker, “ZERO-MOMENT POINT METHOD FOR STABLE BIPED WALKING,” p. 62, Jul. 2009.

[31] Jinsu Liu and M. Veloso, “Online ZMP sampling search for biped walking planning,” 2008, pp. 185–190.

[32] X. Zang, Y. Liu, W. Li, Z. Lin, and J. Zhao, “Design and Experimental Development of a Pneumatic Stiffness Adjustable Foot System for Biped Robots Adaptable to Bumps on the Ground,” *Appl. Sci.*, vol. 7, no. 10, p. 1005, Sep. 2017.

[33] H.-Y. Liu, W.-J. Wang, and R.-J. Wang, “A Course in Simulation and Demonstration of Humanoid Robot Motion,” *IEEE Trans. Educ.*, vol. 54, no. 2, pp. 255–262, May 2011.

[34] E. Hashemi and A. Khajepour, “Kinematic and three-dimensional dynamic modeling of a biped robot,” *Proc. Inst. Mech. Eng. Part K J. Multi-Body Dyn.*, vol. 231, no. 1, pp. 57–73, Mar. 2017.

[35] C. Graf and T. Röfer, “A Center of Mass Observing 3D-LIPM Gait for the RoboCup Standard Platform League Humanoid,” in *RoboCup 2011: Robot Soccer World Cup XV*, vol. 7416, T. Röfer, N. M. Mayer, J. Savage, and U. Saranlı, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 102–113.

[36] K. Youcef-Toumi and O. Ito, “A Time Delay Controller for Systems With Unknown Dynamics,” *J. Dyn. Syst. Meas. Control*, vol. 112, no. 1, pp. 133–142, Mar. 1990.

[37] Y. Kali, M. Saad, K. Benjelloun, and M. Benbrahim, “Sliding Mode with Time Delay Control for MIMO nonlinear Systems With unknown dynamics,” in *2015 International Workshop on Recent Advances in Sliding Modes (RASM)*, Istanbul, Turkey, 2015, pp. 1–6.